CONSULTOR SINUHE ARROYO GÓMEZ

TFC-WEB SEMANTICA MEMORIA FINAL

ALUMNO SINESIO DAVID CARVAJAL TABASCO

INGENIERIA TECNICA INFORMÁTICA DE GESTIÓN

INDICE

1 INTRODUCCION

Este documento de entrega contiene la memoria final del Trabajo Fin de Carrera. En él se describe los pasos realizados de acuerdo al estudio e investigación de la web semántica y a los requerimientos de entrega del proyecto. Estos requerimientos son:

- Ontología sobre un dominio
- Parser para poblar la ontología creada
- Prototipo para visualizar y navegar por dicha ontología

En el documento se describen los avances realizados hasta la fecha los cuales se detallan a continuación:

- Creación de la ontología sobre el dominio elegido, detallando:
  - Conceptos y componentes relativos a la ontología
  - Descripción del dominio elegido
  - Su proceso de diseño, modelado y desarrollo
  - Tecnologías utilizadas

- Análisis funcional del parser que rellene dicha ontología de ontología sobre el dominio de Cloud Computing
- Explicación de Construcción y programación del prototipo del Parser

El documento de memoria final se irá completando en los apartados que aluden a los requerimientos todavía no abordados y que detallamos a continuación:

- Creación de Interfaz Gráfica(modelo SWING java) del prototipo de parser que crea, y puebla la ontología
- Creación del visualizador de la ontología vía navegador, con explicación de la fuente de recogida de información
- Video explicativo del software entregable del proyecto así como de documento final.

2. DESCRIPCION DEL PROYECTO Y OBJETIVOS

2.1OBJETIVO GENERAL

El objetivo general de este proyecto es realizar un estudio e investigación de la web semántica a través de un dominio concreto. La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida **(1).** Para realizar el análisis de la web semántica el dominio elegido es el **del Cloud Computing.** Se ha seleccionado este dominio porque se considera una temática con un gran auge en la actualidad pero sobre la que existe un gran desconocimiento. Utilizando un lenguaje informal "se trata de un término del que todo el mundo ha oído hablar pero que posteriormente no sabe situarlo en el contexto adecuado". Por este motivo y dado que uno de las características de la web semántica, es proporcionar a los usuarios de internet respuestas a sus preguntas de una forma más rápida, se considera que el estudio de este dominio a través de las tecnologías de la web semántica ayudará a proporcionar y trasmitir mayor conocimiento sobre el mismo.
Dentro de este estudio, se analizará una de las tecnologías principales de este tipo de web: **la ontología. Las ontologías** son el medio principal para lograr el objetivo de la web semántica, al facilitar la definición formal de las entidades y conceptos presentes en los diferentes dominios, la jerarquía que les sustenta y las diferentes relaciones que los unen entre sí. **(2)** Para entender su funcionamiento se realizará el diseño y creación de una ontología sobre un dominio del Cloud Computing. Asimismo, para aprender a tratar y modelar dicha ontología, se creará un código de software que permita poblar la misma y un sencillo prototipo que posibilite visualizar y navegar por sus conceptos y relaciones.

2.2OBJETIVOS ESPECIFICOS O SECUNDARIOS

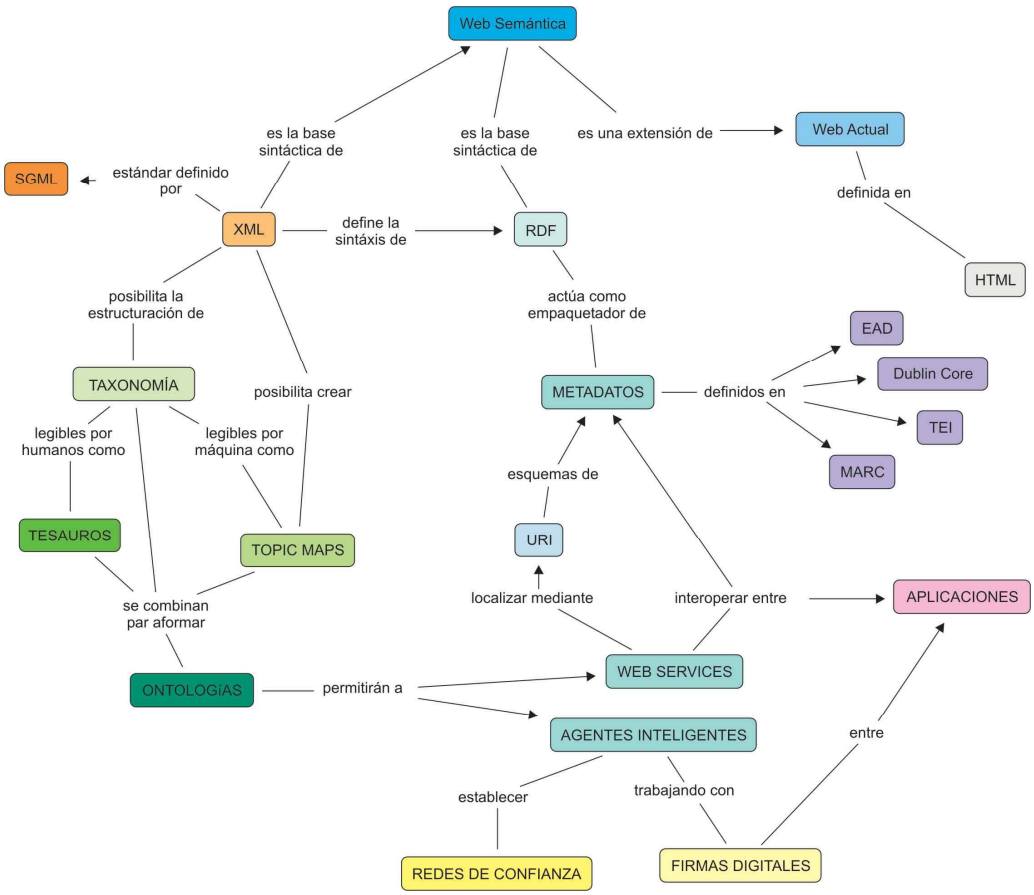
- Conocer y familiarizarse con los **conceptos básicos de la Web Semántica**.
- **Características principales** de la web semántica
- Comparativa **Web Semántica–Web Actual**
- **Tecnologías** web semántica
- Conocer y familiarizarse con el **concepto de ontología.**
- Modelado de una ontología en un dominio a elegir (en este caso Cloud Computing)
- Desarrollar un **parser o analizador sintáctico** que permita conectándose a la Wikipedia y/o DBpedia rellenar con instancias la ontología creada.
- Desarrollar un **pequeño prototipo de un software** que permita navegar por la ontología creada junto con las instancias añadidas de una forma visual, a través de un navegador de Internet.

3. LA WEB SEMANTICA

3.1INTRODUCCION

La web semántica(3) **[Berners-Lee 2001]** propone superar las limitaciones de la web actual mediante la introducción de descripciones explícitas del significado, la estructura interna y la estructura global de los contenidos y servicios disponibles en la WWW.
La Web Semántica como infraestructura basada en metadatos aporta un camino para **razonar** en la Web, extendiendo así sus capacidades. Se trata de aprovechar las capacidades que pueda tener un computador para resolver problemas específicos, a través de operaciones previamente establecidas que se llevarán a cabo sobre datos existentes pero **muy bien definidos**.

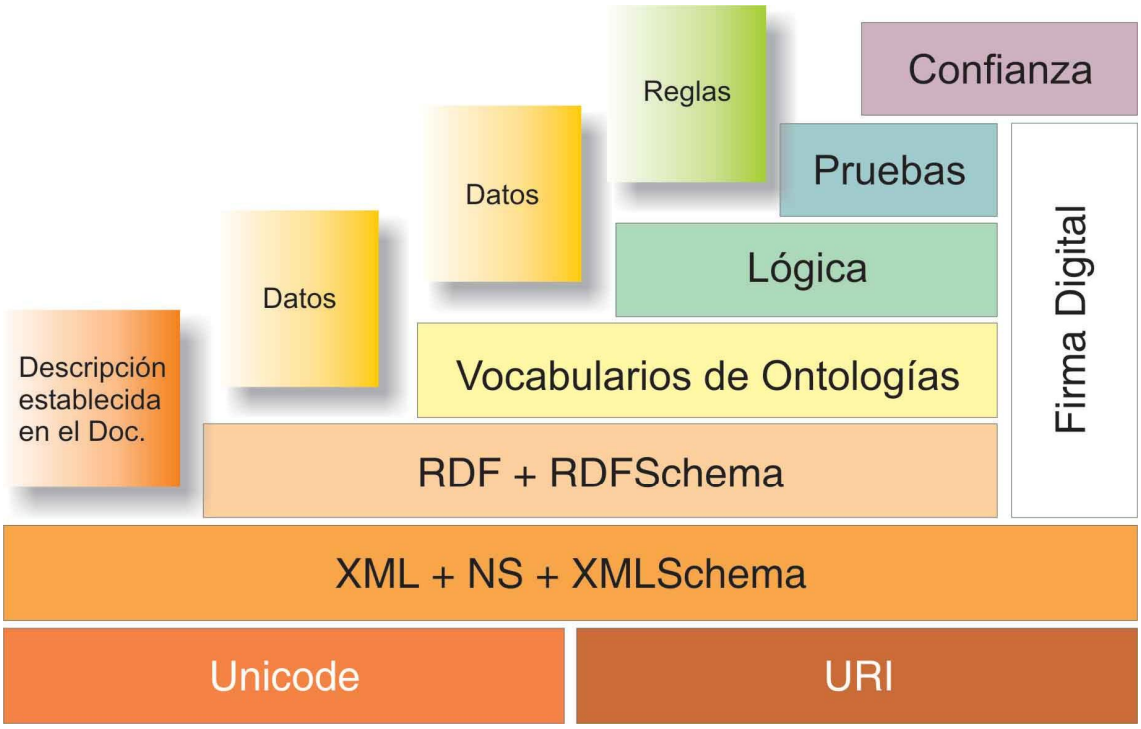
Se trata de área pujante en la que convergen la Inteligencia Artificial y las tecnologías web que propone introducir descripciones explícitas sobre el significado de los recursos, para permitir que las propias máquinas tengan un nivel de comprensión de la web suficiente como para procesar un trabajo que actualmente realizan manualmente los usuarios que navegan por la web. Frente a la semántica implícita, el desorden de los recursos, y la ausencia de una organización definida de la web actual, la web semántica propone clasificar, dotar de estructura y anotar los recursos con semántica explícita procesable por un ordenador. El siguiente mapa conceptual describe los componentes y tecnologías que integran el modelo de la web semántica. El diseño se realizó con el software IMHC Camp Tool, un kit de herramientas orientadas al diseño, mantenimiento e intercambio de mapas conceptuales, desarrollado por el Institute for Human and Machine Cognition (IHMC) de la Florida.



3.2 CARACTERÍSTICAS WEB SEMANTICA

- Estructura mejor la información con el fin de poder localizar y encontrarla de forma más sencilla
- Se basa en metadatos que permiten razonar en la web
- Utiliza un lenguaje universal que permite el intercambio con otros
- Posee herramientas capaces de procesar la información de manera sencilla
- Es estándar y permite que los contenidos puedan ser utilizados y entendidos por cualquier software.
- Los navegadores distinguen más que páginas HTML , con lenguajes como XMTL Y RDF
- Posee orden y flexibilidad.

En la siguiente imagen se muestra una taxonomía de los capas de la web semántica:



3.3COMPARATIVA WEB ACTUAL – WEB SEMANTICA

Para diferenciar el funcionamiento de la web actual con la web semántica enumeramos en primer lugar las características de ambas:

Web actual:

- **Biblioteca Digital con hipertexto**: Enorme biblioteca con documentos (llamados *páginas W*eb) conectados entre sí mediante enlaces
- **Una base de datos (o plataforma común de aplicaciones)**:Un portal común de aplicaciones accesibles a través de páginas Web y que muestran sus resultados como páginas Web
- **Una plataforma para multimedia**: Una nueva forma de transmitir programas de radio, TV y vídeos
- **Un esquema de nombres:** Identidad única para los documentos[(4)]

La Web semántica

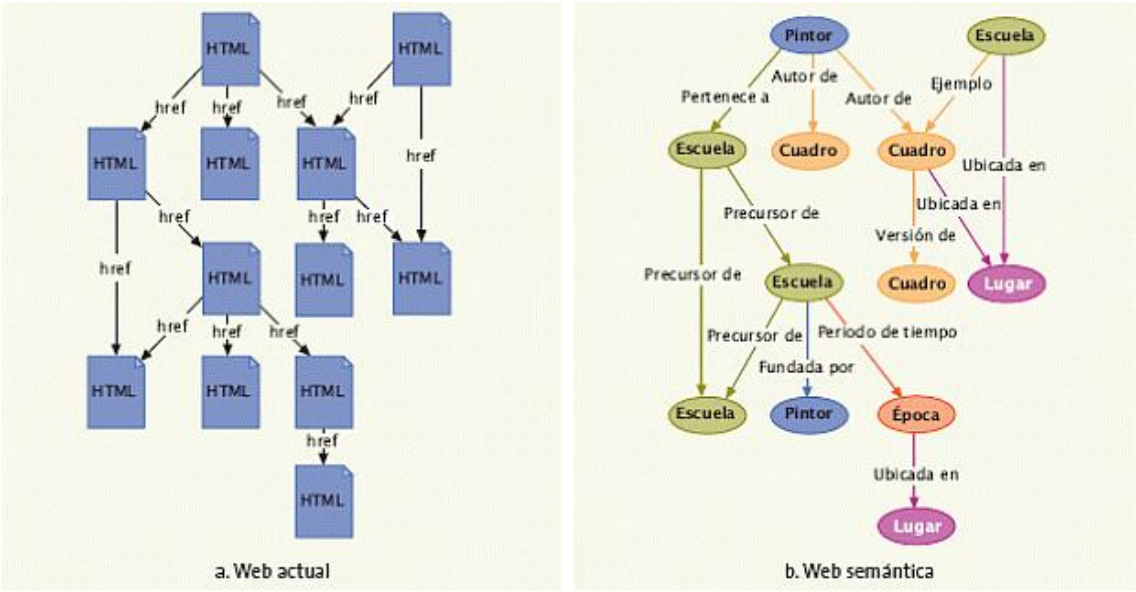
- La información es **procesable por programas**
- La información está clasificada y estructurada.
- La semántica es explícita (metadatos, procesable por máquinas) con vocabularios: ontologías consensuadas y posee orden y flexibilidad.[(5)]

A través de la siguiente tabla podemos comparar las diferencias entre web semántica y web actual.

Característica	Web Actual	Web Semántica
Lenguaje principal de uso	HTML	XML
Forma y Estructura	Documentos no estructurados	Documentos estructurados y siguiendo las pautas XML
Semántica Usada	Semántica implícita	Etiquetado explícito (*metadatos*, Web Semántica)
Relaciones entre Contenido y Forma	HTML = fusión de forma y contenido	Estructura en capas de forma y contenido: XML + transformación (p.e., *XSL*) a HTML, WML, PDF, u otros formatos
Editabilidad	Documentos estáticos	Documentos dinámicos
Interactividad	Medio de difusión unidireccional	*Web editable*, bidireccional
Público al que se dirige	Humanos	Humanos y computadores

Si estudiamos la web actual se asemeja a un grafo formado por nodos del mismo tipo, e hiperenlaces entre ellos igualmente indiferenciados. Por ejemplo, no se hace distinción entre la un blog profesional de una temática concreta y el portal de una tienda on-line, como tampoco se distinguen explícitamente los enlaces de publicidad externa de la tienda con los de los productos concretos. Por el contrario en la web semántica cada nodo (recurso) tiene un tipo (profesor, tienda, pintor, libro), y los arcos representan relaciones explícitamente diferenciadas (pintor – obra, profesor – departamento, libro – editorial).

En la siguiente figura podemos ver un ejemplo[6]



3.4 TECNOLOGIAS UTILIZADAS EN LA WEB SEMANTICA

El primer lenguaje para la construcción de la web semántica fue SHOE15, creado por Jim Hendler en la Universidad de Maryland en 1997.A partir de entonces han aparecido muchos otros lenguajes y tecnologías que detallamos a continuación:

- **UNICODE** Es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas técnicas además de textos clásicos de lenguas muertas. El término Unicode proviene de los tres objetivos perseguidos: universalidad, uniformidad y unicidad.(7)
- **URI** Cadena de caracteres compacta que interactúa y localiza recursos y nombres de cualquier red. El URI se diferencia de URL en que permite incluir en la dirección una subdirección, determinada por el "fragmento". Esto se comprende mejor analizando la estructura de un URI.

  Un URI consta de las siguientes partes:

  - **Esquema**: nombre que se refiere a una especificación para asignar los identificadores, e.g. urn:, tag:, cid:. En algunos casos también identifica el protocolo de acceso al recurso, por ejemplo http:, mailto:, ftp:.
  - **Autoridad:** elemento jerárquico que identifica la autoridad de nombres (por ejemplo //es.wikipedia.org).
  - **Ruta:** Información usualmente organizada en forma jerárquica, que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres (e.g. /wiki/Uniform_Resource_Identifier).
  - **Consulta:** Información con estructura no jerárquica (usualmente pares "clave=valor") que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres. El comienzo de este componente se indica mediante el carácter '?'.
  - **Fragmento:** Permite identificar una parte del recurso principal, o vista de una representación del mismo.(8)

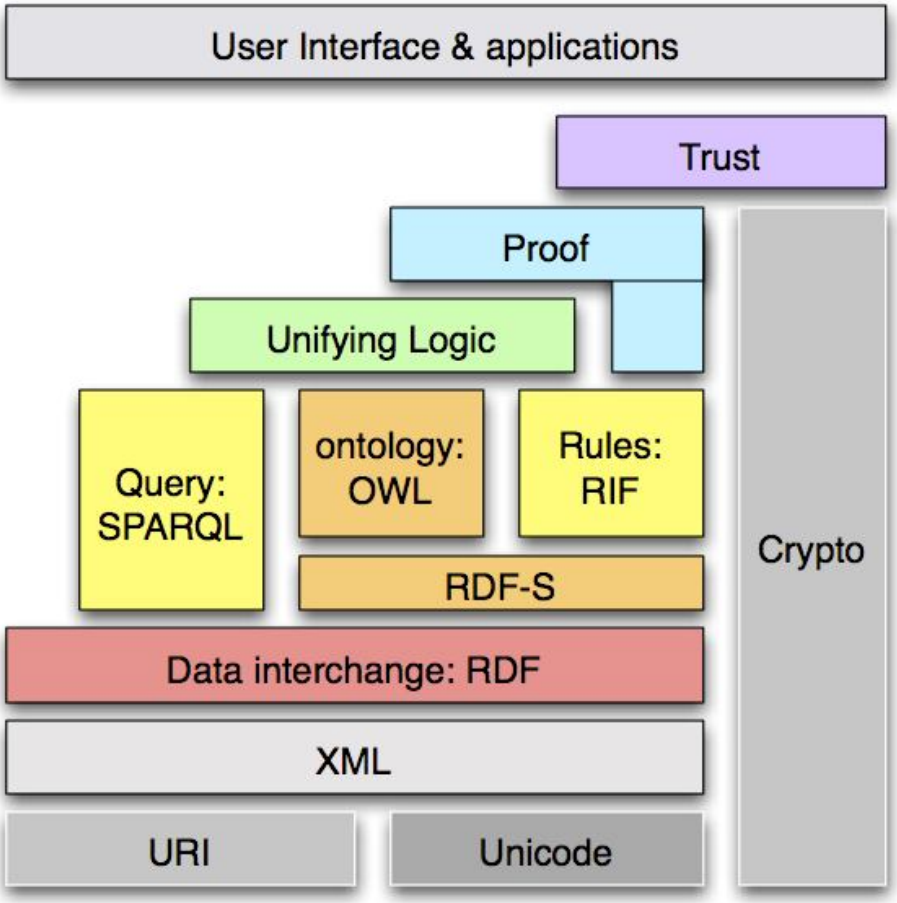
- **XML +NS+xmlschema** Cualifica elementos y atributos de nombres usados en XML asociándolos con los espacios de nombres. XML es un metalenguaje desarrollado en 1998 bajo los auspicios del W3C. Se trata de un lenguaje de marcas, un subgrupo de SGML, específicamente pensado para ser utilizado en entorno web. Es tal la importancia que está adquiriendo este lenguaje que se dice que "XML es el futuro de Internet".XML ofrece información sobre la estructura de los contenidos, lo que permite intercambiar datos estructurados. El otro conocidísimo lenguaje de etiquetas para crear páginas web, HTML, tiene una gran desventaja, y es que sólo tiene capacidad para describir la apariencia de los contenidos en Internet (colores, tamaños, enlaces, etc.).XML, por su parte, ofrece una capacidad limitada para expresar semántica. Por eso se entiende que RDF es a la Semántica lo que XML es a la Sintaxis. El modelo de datos XML consiste en un árbol que no distingue entre objetos y relaciones, ni tiene noción de jerarquías de clases. En cambio RDF cuenta con clases y subclases que definen "esquemas". Por tanto, gracias a RDF se expresan afirmaciones y gracias a su lenguaje de base XML se define la estructura de esas afirmaciones.XML responde a la necesidad de contar con una sintaxis que fuera capaz de representar el modelo planteado por RDF en archivos legibles por ordenador.

La otra recomendación del W3C relacionada con la WS e íntimamente ligada a RDF es OWL (Ontology Web Language).

- **ONTOLOGIA** La web semántica rescata la noción de ontología del campo de la Inteligencia Artificial como vehículo para cumplir este objetivo. Gruber define ontología como "a *formal explicit specification of a shared conceptualization*" [Gruber 1993]. Una ontología es una jerarquía de conceptos con atributos y relaciones, que define una terminología consensuada para definir redes semánticas de unidades de información interrelacionadas. Una ontología proporciona un vocabulario de clases y relaciones para describir un dominio, poniendo el acento en la compartición del conocimiento y el consenso en la representación de éste.
- **RDF+ rdfschema** fue creado en 1998 y recomendado por W3C en 1999. Es acrónimo de *Resource Description Framework* y es un lenguaje para la representación de la información sobre los recursos en la web (autor de una página web, licencia, etc.), particularmente dirigido para la representación de los *metadatos*. Es decir, define la sintaxis y modelos de datos para la representación semántica de los datos. RDF se basa en los estándares de URIs y Unicode además de que se puede presentar en XML (por lo que se le considera como una de sus aplicaciones).

- **OWL** Es un lenguaje de ontologías. OWL puede ser usado para representar explícitamente el significado de términos en vocabularios y las relaciones entre esos términos. Esta representación de términos y sus interrelaciones se denomina ontología. OWL está pensado para ser usado cuando la información contenida en los documentos necesita ser procesada por las aplicaciones, al contrario que en las situaciones donde el contenido sólo necesita ser presentado a los humanos.. OWL tiene mayor capacidad para expresar significado y semántica que XML, RDF, y RDF-S, y, de este modo, OWL va más allá de estos lenguajes en su capacidad para representar contenido interpretable por un ordenador en la Web.
- **SPARQL** es un lenguaje de recuperación basado en RDF; su nombre es un acrónimo recursivo del inglés *SPARQL Protocol and RDF Query Language*. Se trata de una recomendación para crear un lenguaje de consulta dentro de la Web semántica que está ya implementada en muchos lenguajes y bases de datos. Con **SPARQL** los desarrolladores y usuarios finales pueden representar y utilizar los resultados obtenidos en las búsquedas a través de una gran variedad de información como son datos personales, redes sociales y metadatos sobre recursos digitales como música e imágenes. Es de utilidad para la **recuperación y organización de información**

A continuación se muestra un esquema de las diferentes tecnologías que se utilizan en la web semántica:



4. CREACION DE LA ONTOLOGIA

4.1DEFINCION DE ONTOLOGIA

- WordNet] *"The metaphysical study of being and existence"*
- [Webster's] *"That department of the science of metaphysics which investigates and explains the nature and essential properties and relations of all beings, as such, or the principles and causes of being"*
- [Merriam-Webster] *"A branch of metaphysics concerned with the nature and relations of being; a particular theory about the nature of being or the kinds of existents"*
- [Gruber 93] *"A formal explicit specification of a shared conceptualization"*

La definición más utilizada para el término ontología es la de Gruber , ya citada en este documento , "se trata de la especificación explícita de una conceptualización". Para la inteligencia Artificial lo que existe es aquello que puede ser representado conceptualmente. Por ello cuando el conocimiento de un dominio es implementado en un formalismo declarativo, el conjunto de objetos que pueden ser representados se denomina le universo del discurso. Este conjunto de objetos se refleja en el vocabulario que representa el conocimiento. Borst modifico ligeramente la definición de Gruber de la forma siguiente: "Las ontologías se definen como la especificación formal de una

conceptualización compartida". Studer especifico cada uno de los términos de las definiciones de Gruber y Borst:

- **Conceptualización** implica que toda ontología desarrolla un modelo abstracto del dominio o fenómeno del mundo que representa. Dicho modelo abstracto se basa esencialmente en el empleo de conceptos, atributos, valores y relaciones.
- **Explícita** se refiere a que los conceptos usados y las definiciones para su uso se definen explícitamente
- **Formal** La ontología debe ser interpretable por un ordenador
- **Compartida** La ontología va a tratar sobre conocimiento aceptado públicamente y consensuado, no el conocimiento propuesto de forma individual

La Ontología en Informática(no confundir con el término filosófico) hace referencia al intento de formular un **exhaustivo y riguroso esquema conceptual dentro de un dominio dado**, con la finalidad de facilitar la comunicación y la compartición de la información entre distintos sistemas.**(9)**

Las ontologías son, por tanto, **herramientas para la representación del conocimiento** y constituyen un vehículo para acercarnos a la WS.

Una ontología contiene un **vocabulario de conceptos** así como las **relaciones entre estos conceptos**. Las ontologías **definen de forma estándar los términos y sus relaciones** dentro de **un determinado dominio** (área del conocimiento), formando redes jerárquicas semánticas

Las ontologías facilitan **reglas lógicas y restricciones** para hacer comprender a las máquinas los conceptos que se manejan en ese campo. (por ejemplo en una ontología de arte, establecemos que todos los escultores son artistas pero no todos los artistas son escultores).

Las ontologías están pensadas para actuar como **referencia común entre sistemas distintos** que utilizan conceptos similares (control del vocabulario). Se busca lograr el consenso sobre cómo representar el conocimiento para poder así compartirlo y facilitar la **interoperabilidad**. Estas herramientas codifican el conocimiento de un dominio y también el conocimiento que se expande a través de varios dominios. La adopción de **ontologías comunes** es clave para que todos los que participan en la WS puedan trabajar de forma autónoma con la garantía de que "hablan el mismo idioma".

Las ontologías pueden hacer uso de cualquier de los lenguajes de representación del conocimiento existentes, normalmente basados en **XML**, pero el más común es **RDF**.

La ontología contiene los siguientes elementos:

- **Clases**: conceptos generales de un dominio determinado. (Ejemplo.: en una ontología de deportes, cada clase sería un deporte. Fútbol, Baloncesto, Ciclismo, cada uno conformaría una clase).
- **Relaciones** entre clases e instancias (jerarquías)
- **Instancias**: instancias particulares del concepto (subclases)
- **Propiedades** de las clases y las instancias (características)
- **Restricciones y reglas de inferencia**: aplican la lógica (Si A tiene relación con B, B tiene relación con A)

Las ontologías se empiezan a utilizar a finales de los 80 en el campo de la inteligencia artificial como medio para la compartición y reusabilidad de conocimiento. En la segunda mitad de los 90 se empiezan a aplicar a la web para la inclusión de descripciones semánticas explícitas de recursos (contenidos y servicios).Hoy son un eje fundamental en las nuevas tecnologías para la web semántica.

Para profundizar más en el estudio de la tecnología de las ontologías en la web semántica se procede a diseñar y modelar una sobre un dominio concreto: en este caso el dominio del Cloud Computing. Para ello y con el fin de proporcionar un mejor entendimiento al conocimiento formalizado en la ontología, comprendiendo sus términos a nivel de clases, propiedades, instancias y relaciones se define el dominio, describiendo sus características y enumerando sus ventajas, inconvenientes y aplicaciones.

4.2DESCRIPCION DEL DOMINIO ELEGIDO PARA LA CREACION DE LA ONTOLOGIA

Cloud Computing es un nuevo modelo de prestación de servicios tecnológicos a través de la plataforma de internet(10). Mediante el Cloud Computing se genera la capacidad de consumir servicios IT de forma ágil y flexible. El término es una tendencia que responde a múltiples características integradas. Se habla de "nube" por la ubicación de los servicios de computación dentro de la red de internet. El avance más importante que ofrece la computación en nube es que permite aumentar el número de servicios basados en la red. Esto genera beneficios tanto para los proveedores, que pueden ofrecer, de forma más rápida y eficiente, un mayor número de servicios, como para los usuarios que tienen la posibilidad de acceder a ellos de forma rápida y trasparente disfrutando de un modelo de pago por consumo.

Dentro de las nubes de computación se pueden distinguir las **nubes públicas** si el propietario de la nube es un proveedor que la mantiene por la empresa, la cual paga por el uso y disfrute del recurso a través de internet, y puede ser **privada** si la nube se mantiene dentro de las instalaciones de la propia empresa.

4.3TECNOLOGIAS UTILIZADAS POR EL CLOUD COMPUTING

Las tecnologías en que suele apoyarse el cloud computing son virtualización, infraestructura multi-tenant, escalabilidad, :

4.3.1LA VIRTUALIZACION

Se trata de una de la tecnologías más utilizadas y sobre las que más se apoya el Cloud Computing. Muchos de los servicios ofertados a través del Cloud Computing son ofrecidos a través de la virtualización de aplicaciones o de virtualización de servidores o escritorios.

Igualmente se produce virtualización de redes y almacenamiento para la oferta de estos servicios mediante la "nube".

4.3.2 INFRAESTRUCTURA MULTI-TENANT

Las arquitecturas **multi-tenant** (multi-propietario) son cada vez más utilizadas entre los proveedores de **SaaS** (Software as a Service). En un entorno **multi-tenant**, todos los clientes y sus usuarios consumen el servicio desde la misma plataforma tecnológica, el

intercambio de todos los componentes de la tecnología incluyendo el modelo de datos, servidores y las capas de base de datos.
Las arquitecturas **Multi-Tenant** se refiere a un principio en la arquitectura de software donde una única instancia del software se ejecuta en un servidor y al servicio a múltiples clientes (los arrendatarios).
Multi-Tenant contrasta con una arquitectura **multi-instancia** en la cual son instancias independientes de software (o sistemas de hardware) que establecen para las organización de distintos clientes. Con una arquitectura **multi-tenant**, una aplicación de software está diseñada para la partición sus datos y la configuración de manera que cada cliente trabaja con una instancia de la aplicación virtual personalizada
Este tipo de arquitecturas se denominan **multi-tenant** o **multipropietario**, en las que sobre un único recurso operan múltiples usuarios que son dueños, por así decirlo, del mismo.

La distribución de recursos a través de técnicas cloud utiliza este tipo de de diseño e implementación. Para lograr una distribución de aplicaciones multi-tenant hay que centrarse principalmente en tres aspectos fundamentales:

- **El mecanismo para la distribución de recursos** para reducir el costo del hardware, del software y de la administración de cada tenant
- **El mecanismo de aislamiento de seguridad** para evitar posibles accesos no válidos, conflictos e interferencias entre los tenants
- **El mecanismo de personalización** para darle soporte al modelo de UI por tenant, de control de acceso, de proceso y de datos a través de los enfoques de configuración

Una arquitectura multit-tenancy tiene que basarse en cuatro pilares:

- **Disponibilidad.** Si una infraestructura compartida falla, podría afectar a un alto número de clientes a los que se ofrezca estos servicios. Por ello, la infraestructura debe proporcionar una redundancia integrada de forma que los recursos necesarios de informática, red y almacenamiento sigan estando disponibles en caso de posibles fallos
- **Separación segura.** Dado que se trata de una arquitectura compartida , cada cliente debe estar separado de forma segura.
- **Garantía de servicios.** Debe aislarse y garantizarse el rendimiento del equipo, de la red y del almacenamiento durante el funcionamiento normal, si se producen fallos o si determinados servicios generan cargas excepcionales. Esta solución coloca la clase de servicio lo más cerca posible de la aplicación, asigna ese valor a una definición de normativa y garantiza que la normativa se aplique uniformemente en todas las capas de acuerdo con las características únicas de cada una de ellas.
- **Gestión.** La capacidad de aprovisionar, gestionar y supervisar rápidamente todos los recursos debe ser una cuestión fundamental. Los clientes deben acceder a la gestión de sus servicios de manera inmediata y trasparente.

4.3.3.ESCALABILIDAD

Si se define la escalabilidad en el ámbito de la informática como la propiedad que cualquier sistema debería poseer para añadir nuevos componentes y así dar cobertura a un crecimiento de tu demanda. Una de las ventajas más importantes del **cloud computing** en el nivel de infraestructura (**iaas**) es la facilidad y rapidez para poder escalar los

sistemas en función de tus necesidades y tan importante es esta propiedad como la posibilidad de "**desescalarlos**" que también provee el cloud computing.

En el nivel del **paas(plataforma como servicios)** y **saas (software como servicio)**, la escalabilidad corre a cargo del proveedor formando parte del conjunto de servicios que ofrecen sus soluciones, es decir, el usuario de las **paas** y el **saas** no se preocupa de este término. Por último, en el nivel del **saas** la escalabilidad también puede referirse a la posibilidad de aumentar el número de usuarios que pueden acceder a la aplicación y esto al igual que en el nivel de infraestructura es una propiedad destacable ya que se puede realizar con facilidad y rapidez.

4.4 LOS TRES NIVELES DEL CLOUD COMPUTING

El Cloud Computing se puede dividir en tres niveles en función de los servicios que se ofrezcan a los clientes. Desde el más interno hasta el más externo podemos citar: IaaS (Infraestructura como Servicio), Paas (Plataforma como Servicio) y SaaS(Software como servicio).

En la siguiente imagen se muestra los niveles indicados de Cloud Computing y la forma en que se relacionan:

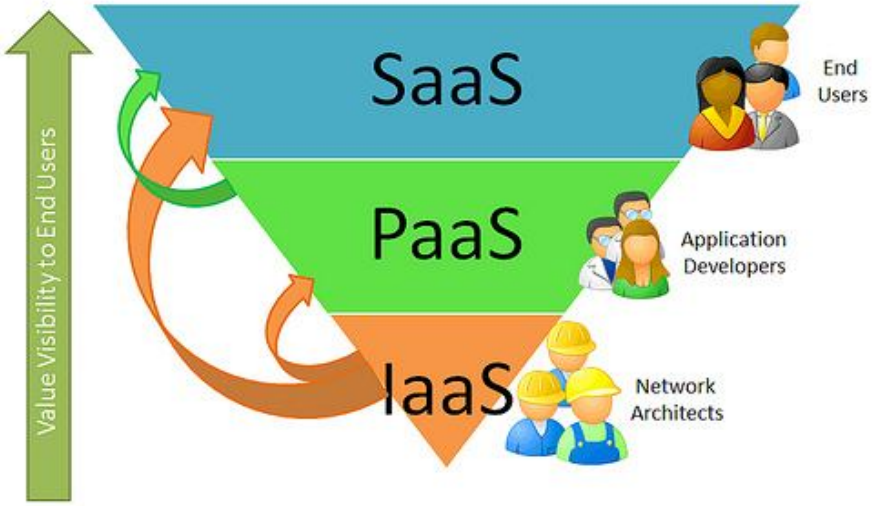


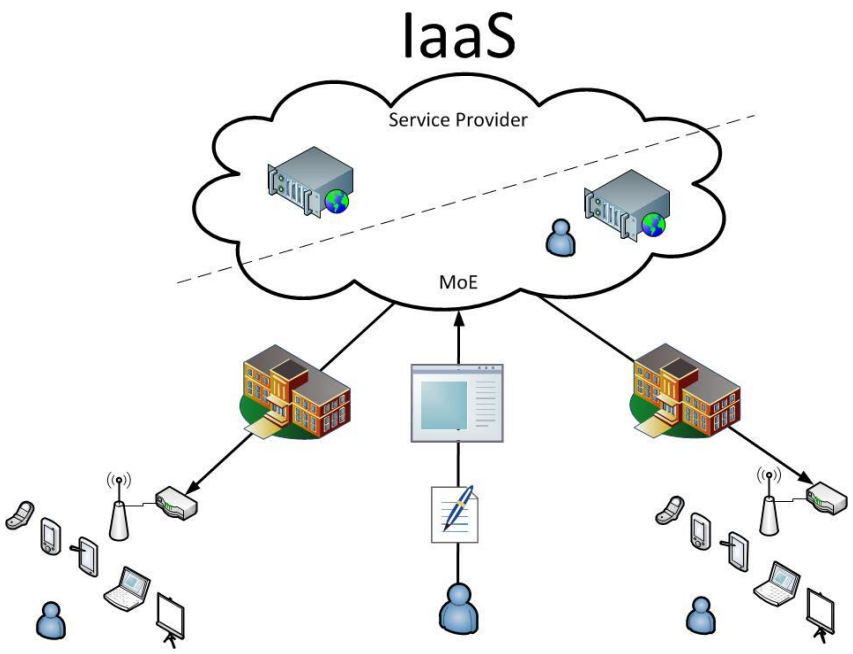
IMAGEN DE http://www.technoreeze.com/2011/07/15/cloud-computing-v-infraestructura-como-servicio-iaas/

4.4.1IaaS (INFRAESTRUCTURA COMO SERVICIO)

Se puede incluir en este nivel de Cloud Computing la computación y almacenamiento, es decir, CPU y disco. La idea básica es la de externalización de servidores para espacio en disco, base de datos y/o tiempo de computación, en lugar de tener un control completo de los mismos con el datacenter dentro de la empresa u optar por un centro de datos y sólo administrarlo. Con una **Infraestructura como servicio** (Iaas) lo que se tiene es una solución basada en virtualización en la que se paga por consumo de recursos: espacio en

disco utilizado, tiempo de CPU, espacio en base de datos, transferencia de datos. El ejemplo comercial más conocido son los servicios **EC2 de Amazon** que apoyándose en la virtualización ofrecen máquinas virtuales con un diseño específico. En este nivel incluimos lo que serian los servicios de almacenamiento no relacionado, disco, y también los servicios de almacenamiento relacionado, es decir, las bases de datos.

La ventaja más inmediata de elegir este tipo de soluciones es la desplazar una serie de problemas al proveedor relacionados con la gestión de las máquinas. A continuación habría que situar el ahorro de costes al pagar sólo por lo consumido y aprovechar las economías de escala que tienen empresas de gran tamaño como Amazon. Además tenemos que las **Infraestructura como servicio** pueden permitir una escalabilidad automática o semiautomática, de forma que podamos contratar más recursos según los vayamos necesitando



Muchos expertos dividen los servicios IaaS en dos niveles más DaaS(almacenamiento como servicio y CaaS(Comunicaciones como servicio).

4.4.1.1Caas(COMUNICACIONES COMO SERVICIO)

Las comunicaciones como servicio nacen de la tecnología VoIP. La VoIP, que puede ser virtualizable, ha pasado a convertirse en un candidato más a ser hosteado. Dicho de otra manera, a ser ofrecido como un servicio más.

De ahí nace el concepto **CaaS (Communication as a Service).** De esta manera, el usuario final ya no se ve obligado a realizar una costosa inversión inicial en licencias software y equipamientos dedicados, sino que pasa a pagar mes a mes, en función de los distintos servicios que quiera contratar (Voz, UM, UC, CC, etc…). Como la necesidad de una garantía de **calidad de servicio** (**QoS**) para la comunicación de la red crece para los sistemas de Cloud Computing, la comunicación se convierte en un componente vital de dicha infraestructura. En consecuencia, los sistemas de **Cloud Computing** están obligados a proporcionar cierta capacidad de comunicación orientada al servicio, configurable, programables, predecibles y fiables. Con este objetivo, el concepto de **Comunicación**

como Servicio (**CaaS**) surge en apoyo de tales requisitos, así como seguridad de redes, aprovisionamiento dinámico de superposiciones virtuales para el aislamiento de tráfico o ancho de banda dedicado, el cifrado de comunicaciones y monitoreo de redes

4.4.1.2DaaS(DATAWAREHOUSE COMO SERVICIO)

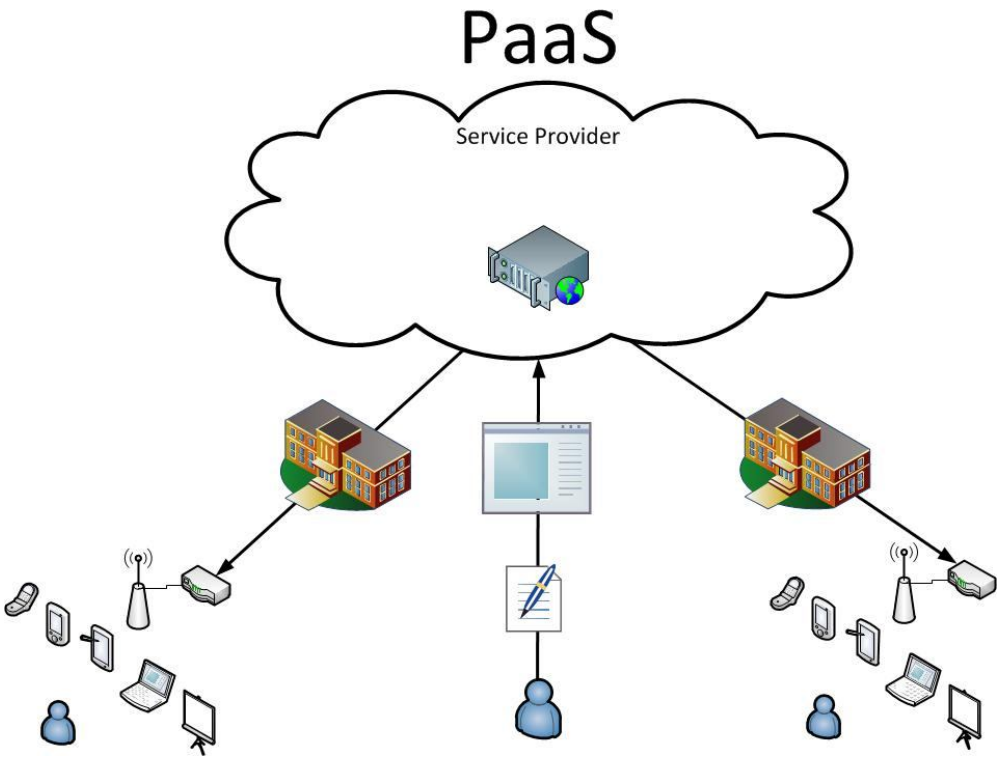
Se trata de ofrecer almacenamiento a través de la red. Con el almacenamiento se ofrecen servicios de replicación y backup. DaaS se basa en el concepto de que el producto, los datos en este caso, se puede proporcionar "on demand" ,bajo demanda, para el usuario, independientemente de la separación geográfica o de organización del proveedor y de los consumidores. Además, la aparición de la arquitectura orientada a servicios (SOA) ha hecho que la plataforma actual en el que residen los datos también resulte irrelevante. Este desarrollo ha permitido la reciente aparición del concepto relativamente nuevo de DaaS.

Uno de los resultados de este paradigma es la agrupación de los datos y el software necesario para ofrecerlo en un solo paquete, que se vende como un producto de consumo. A medida que el número de paquetes de software / datos de paquetes proliferan y se requiere la interacción entre unos y otros, se necesita otra capa de interfaz. Estas interfaces son conocidas colectivamente como la integración de aplicaciones empresariales (EAI).
La existencia de esta situación contribuye al atractivo de DaaS hacia los consumidores de datos, ya que permite la separación de los costes y uso de datos de la de un software específico o plataforma.

4.4.2PaaS (PLATAFORMA COMO SERVICIO)

Se trata del conjunto de plataformas compuestas por uno o varios servidores de aplicaciones y una base de datos (aunque no todas la plataformas incluyen la posibilidad de tener la BBDD) que ofrecen la posibilidad de **ejecutar aplicaciones** (escritas en los lenguajes que la plataforma soporte) encargándose el proveedor de escalar los recursos en caso de que la aplicación lo requiera. Además el proveedor velará por el rendimiento óptimo de la plataforma, actualizaciones de software, seguridad de acceso, etc. Paas (Platform as a Service o Plataforma como Servicio) es el resultado de la aplicación al desarrollo de Software del modelo SaaS (Software como servicio). El modelo PaaS abarca el ciclo completo para desarrollar e implantar aplicaciones desde Internet.

PaaS incluye todas las facilidades al programador para realizar prototipos, analizar, desarrollar, testear, documentar y poner en marcha aplicaciones todo en un sólo proceso. Paas da servicio de integración de la base de datos, seguridad, escalabilidad, almacenaje, copias de seguridad, versioning, y facilidad para colaborar en la comunidad.



Todos estos servicios son ofrecidos e integrados en una sola solución PaaS a través de Internet.

Características de PaaS. Los diferentes Servicios PaaS ofrecen diferentes combinaciones de servicios y soporte de aplicación para el ciclo de desarrollo.

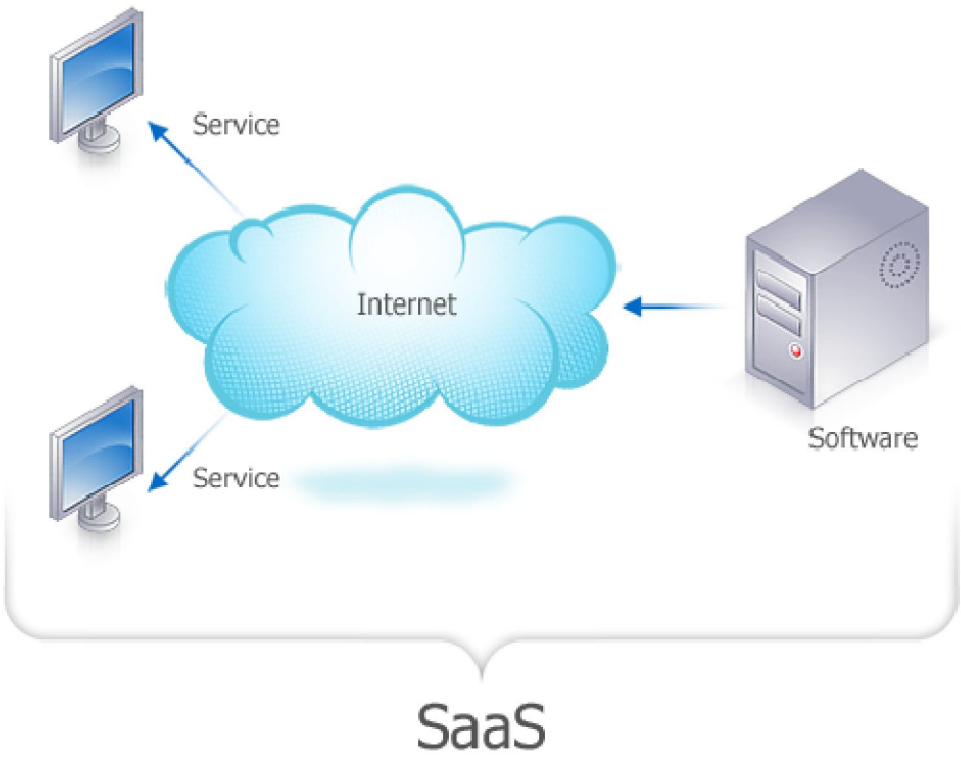
- Arquitectura Multi-Usuario
- Escalabilidad del sistema al desarrollador. Debe de incluir facilidades para que el desarrollador pueda tener cuantos usuarios necesiten sus aplicaciones, manteniendo la seguridad y escalabilidad del sistema.
- Soporte para desarrollo Colaborativo
- La capacidad para desarrollar y compartir código fuente con diferentes desarrolladores, que pueden estar ubicados en diferentes emplazamientos geográficos. PaaS mejora la productividad de los equipo de desarrollo.

4.4.3SaaS(SOFTWARE COMO SERVICIO)

Es el más conocido de los tres niveles del **cloud computing** y el que suele tener como target al cliente final que utiliza el software para ayudar, mejorar o cubrir algunos de los procesos de su empresa. El **saas** es aquella aplicación "consumida" a través de Internet, casi siempre a través del navegador , cuyo pago está condicionado al uso de la misma y donde la lógica de la aplicación como los datos residen en la plataforma del proveedor. En contadas ocasiones es necesario instalar algo en el pc del cliente y si se necesita suele ser alguna plugin o pequeña aplicación a modo de interface para que el usuario pueda interactuar con el sistema. La flexibilidad o escalabilidad de este parte del cloud computing se suele refleja en la facilidad para añadir o quitar usuarios que hacen uso de la aplicación.

Las características del Software como Servicio son las siguientes:

- ***Acceso a través de internet***: Es una característica principal y definitoria "el software puede ser consultado en cualquier computador, esté presente en la empresa o no". Como consecuencia de esta característica, no se requiere instalación de programas en local, y permite el uso multiplataforma.
- ***Alojado en servidores que gestiona la compañía proveedora***. Para poder cumplir "servicio de mantenimiento, operación diaria, y soporte del software usado por el cliente", tal y como lo dice la wikipedia "Se deduce que la información, el procesamiento, los insumos y los resultados de la lógica de negocio del software está hospedado en la compañía de IT".
- Pago por uso. Pago mensual de una cantidad fija por empleado, que incluye todos los conceptos relacionados con el software (**licencia de uso, hosting, soporte y mantenimiento**).



4.5 VENTAJAS DEL CLOUD COMPUTING

En primer lugar una de las ventajas más repetidas es la de reducción de costes.

- Por ello se puede decir que hay 0€ en inversión hardware => 0€ en mantenimiento
- Posibilidad de aumentar o disminuir el consumo de los recursos hardware o software inmediatamente y en algunos casos automáticamente.
- Pago en función de la demanda y por tanto permitiendo un control más eficiente de los gastos.
- Acceso inmediato a la mejoras del recurso propuesta (hardware y software) y correcciones de Bugs.

- Disfrutar de los procedimientos de seguridad, disponibilidad y performance más avanzados de los proveedores con experiencia y conocimientos en este tipo de servicios.
- Acceso a los recursos desde cualquier ubicación

4.6 DESVENTAJAS DE CLOUD COMPUTING

- Percepción de inseguridad. Datos y lógica de negocio fuera de la empresa
- Integración. Dificultad para integrar los recursos cloud con los sistemas propios
- Disponibilidad Sujeto a paradas por mantenimiento programadas por el proveedor y no por el cliente.
- Fallos Dos puntos de fallo muy críticos en la propia infraestructura: Proveedor de servicios cloud y proveedor de Internet.
- Aseguramiento de protección de datos y aplicación de la LOPD

4.7 PROVEEDORES DE CLOUD COMPUTING

A continuación se detalla proveedores de Cloud Computing, diferenciados por los distintos niveles de servicios de la nube, describiendo en cada uno de ellos su infraestructura, características y servicios:

- **Proveedores de IaaS (Infraestructura como Servicio)**
  - **Amazon web services** Proporciona infraestructura de servicios elástica donde alojar computación , almacenamiento o sistemas empresariales. Ofrece servicios de servidores bajo demanda, bases de datos , almacenamiento de datos, servicios web y recuperación de datos.
  - **RackSpace:** Servicios de servidores, almacenamiento, balanceo de carga
  - **TerreMark:** Servicios de redes y conectividad

- **Proveedores de PaaS (Plataforma como Servicio)**
  - **Google App Engine** Proporciona hosting, procesamiento y base de datos. Dentro de sus componentes se puede destacar plataformas de desarrollo como Python , bases de datos como SimpleBD.
  - **SalesForce:** Con la plataforma de desarrollo Force.com de código Apex el IDE VisualForce
  - **Microsoft Azure:** ofrece Windows Azure como un nuevo sistema operativo pensado para 'La Nube' y un conjunto de servicios para desarrolladores que pueden usarse de manera conjunta. Este conjunto de servicios ofrecen un rico conjunto de APIs para la gestión de la infraestructura, y otro conjunto de aplicaciones ofrecidas como servicios a los clientes -Windows Live, Microsoft Dynamics, Exchange Online y Sharepoint.
- **Proveedores de SaaS (Software como Servicio)**
  - **Google Apps** Formato OpenSource en aplicaciones como Google Gmail, Google Calendar , Google Docs, Google Video y Google Sites.
  - **SalesForce:** CRM:Sales.Cloud, Correo: AppExchange, Conversación:Chatter.

- Zendesk: Servicio de tención al cliente

4.8 DESCRIPCION DE LA ONTOLOGIA

Una vez descrito que se va a modelar por la ontología, a continuación detallamos los aspectos que van a componer la ontología del dominio Cloud Computing. La ontología a desarrollar tiene que cubrir todos los aspectos descritos sobre el dominio del Cloud Computing:

- Descripción General de Cloud Computing
- Características del Cloud Computing
- Tecnologías de Cloud Computing
- Niveles de Cloud Computing
- Ventajas de la nube
- Desventajas
- Proveedores y servicios

Con el uso de esta ontología se especificará el significado de las anotaciones y el contexto del dominio Cloud Computing. Esta ontología deberá permitir:

- Proveer un vocabulario de términos
- Combinar los términos existentes provocando la creación de otros nuevos
- Especificar formalmente el significado de cada término
- Especificar las relaciones entre términos

Para conocer si se cumple correctamente con la conceptualización del dominio seleccionado, se plantean de inicio una serie de preguntas que tienen que ser respondidas por la ontología creada:

- ¿Qué es el cloud computing?
- ¿Para qué sirve el cloud computing?
- ¿Cómo se puedo utilizar?
- ¿Beneficios del cloud computing?
- ¿Riesgos del cloud Computing?
- ¿Proveedores actuales de Cloud Computing?
- ¿Qué servicios ofrece el cloud computing en la actualidad?

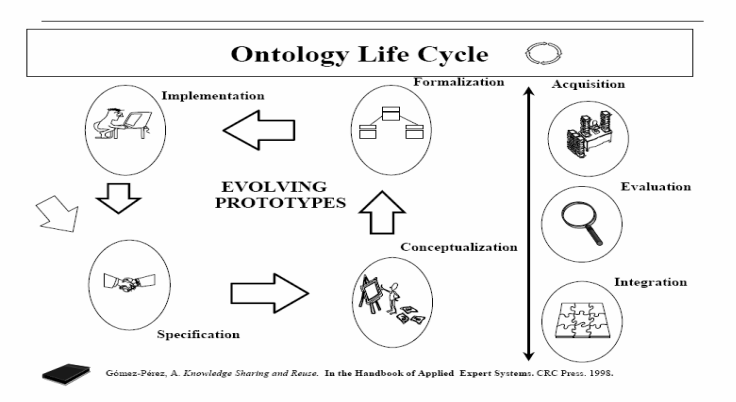
Mediante estas preguntas se puede determinar el alcance del modelo de ontología Juzgando a partir de esta lista de preguntas, la ontología incluirá la información de la definición de la nube, las características del Cloud Computing, ventajas, desventajas, proveedores y servicios.

Para el desarrollo de la ontología se utilizará la metodología **Methontology.** Se trata de una metodología creada en el Laboratorio de Inteligencia Artificial de la Universidad Técnica de Madrid. La creación de la ontología puede empezar desde cero o en base a la reutilización de otras existentes. Methontology incluye la identificación del proceso de desarrollo de la ontología (calendario, control, aseguramiento de calidad, adquisición de conocimiento), un ciclo de vida basado en la evolución de prototipos, para lo cual se sigue - los pasos definidos en el estándar IEEE 1074 de desarrollo de software (2) : que son:

- **Especificación.-** Definir el alcance y granularidad de la ontología.
- **Conceptualización.-** Permite organizar y estructurar el conocimiento adquirido mediante tablas, lenguaje UML, jerarquías etc.

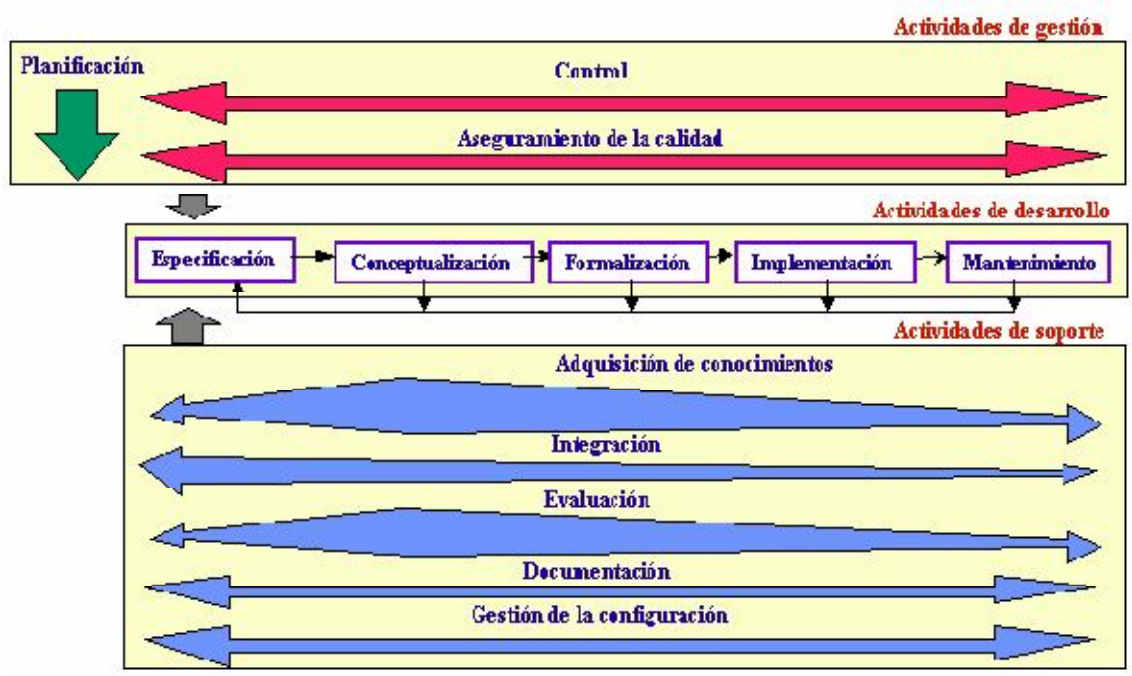
- **Implementación**.- Representa la formalización de la ontología; es decir pasar la conceptualización de la ontología a un lenguaje como RDF, OWL, etc.
- **Evaluación.-** Comprobar el funcionamiento de la ontología

El proceso de desarrollo de una ontología de conformidad con lo establecido en esta metodología se identifica en la siguiente figura:



4.9 DISEÑO DE LA ONTOLOGÍA

El diseño de la ontología tiene que llevar a cabo todas las actividades que conforman cada prototipo del ciclo de vida de la ontología.



Un primer paso es el estudio de ontologías existentes sobre el dominio y verificar si es posible refinar y extender recursos existentes para nuestro dominio y tarea particular. Hay bibliotecas de ontologías reusables en la Web, para verificar la posibilidad de reutilizar una ontología ya realizada sobre el dominio elegido del Cloud Computing se han analizado las siguientes:

- **Ontolingua** (http://www.ksl.stanford.edu/software/ontolingua/)

- **DAML** (http://www.daml.org/ontologies/).
- **UNSPSC** (www.unspsc.org),
- **RosettaNet** (www.rosettanet.org),
- **DMOZ** (www.dmoz.org)

En ninguna de ellas se ha encontrado una ontología del dominio a modelar que pudiera ser reutilizable en este estudio. Por ello se realiza la ontología sobre este dominio desde el principio. No obstante, se utilizará como base de conceptualización de dominio la propuesta de Ontología de la Universidad Santa Bárbara de California, que se muestra en la siguiente imagen.

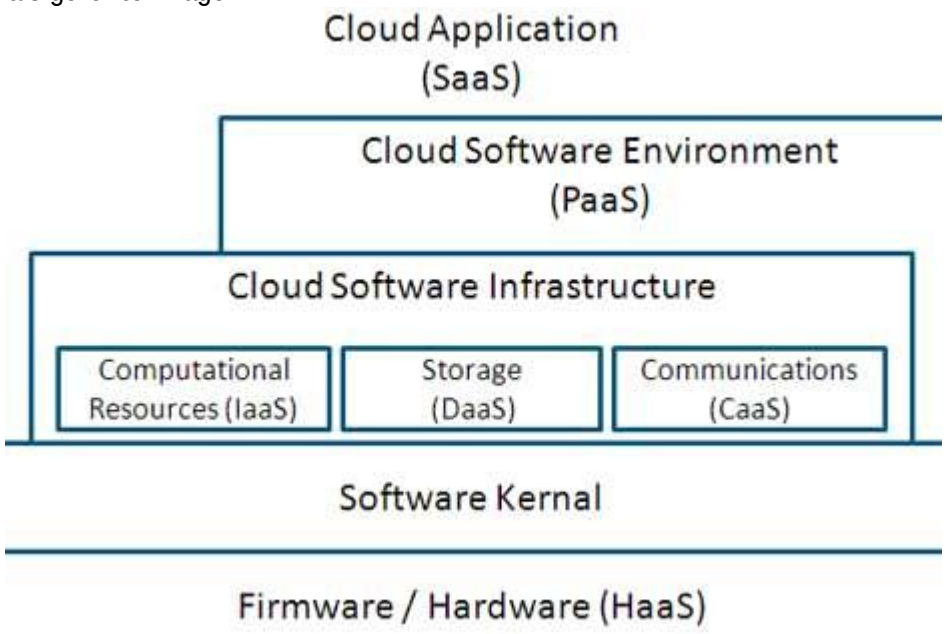
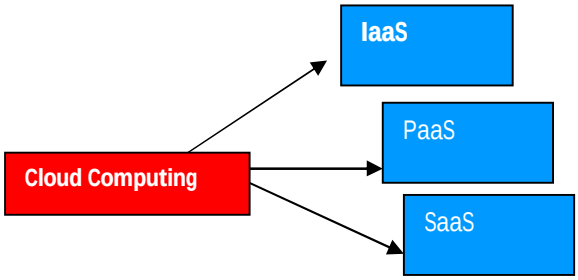


Imagen de ontología propuesta por Lamia Youseff University of California, Santa Barbara Santa Barbara, CA 93106

4.10PREPARACION DEL DISEÑO DE LA ONTOLOGIA

Como primer paso para la construcción de esta nueva ontología y siguiendo los criterios de la metodología Methontology es necesario conceptualizar usando un conjunto de representaciones intermedias tabulares y gráficas, las cuales permiten modelar los principales componentes. Estos componentes son:

- **Conceptos:** Son objetos o entidades, considerados desde un punto de vista amplio. Ejemplo, en el dominio cloud computing: Cloud Computing , IaaS, PaaS, Saas, Proveedores, Servicios
  Los conceptos de una ontología se organizan en taxonomías en las cuales se pueden aplicar mecanismos de herencia.
- **Relaciones:** Representan un tipo de asociación entre conceptos del dominio.



Concepto	Relación Binaria	Concepto
PaaS	Nivel de	Cloud Computing
IaaS	Nivel de	Cloud Computing
SaaS	Nivel de	Cloud Computing

Esta relación se podría concretar como que IaaS, PaaS, SaaS son niveles de Cloud Computing.

- **Instancias:** Representar individuos en la ontología.

Concepto	Instancias
Proveedor Cloud	Amazon, Microsoft

- **Atributos:** Describen propiedades. Se pueden distinguir dos tipos de atributos:
  - Atributos de instancia: Describen propiedades de las instancias de los conceptos, en las cuales toman su(s) valor(es). Estos atributos se definen en un concepto y se heredan a sus subconceptos e instancias.
  - Atributos de clase Describen conceptos y toman su(s) valor(es) en el concepto en el cual se definen. Estos atributos no se heredan ni a los subconceptos ni a las instancias.

- **Axiomas:** Expresiones lógicas que son siempre verdaderas y son utilizadas normalmente para especificar restricciones en la ontología.
  Ejemplo: Es diferente el nivel de servicios IaaS de Cloud Computing del nivel de servicios PaaS.
- **Reglas:** Son usadas generalmente para inferir conocimiento en la ontología, tales como valores de atributos, instancias de relaciones, etc.
  Ejemplo: A un servicio de Cloud se accede a través de Internet

4.11TAREAS DE DISEÑO Y MODELADO DE LA ONTOLOGIA

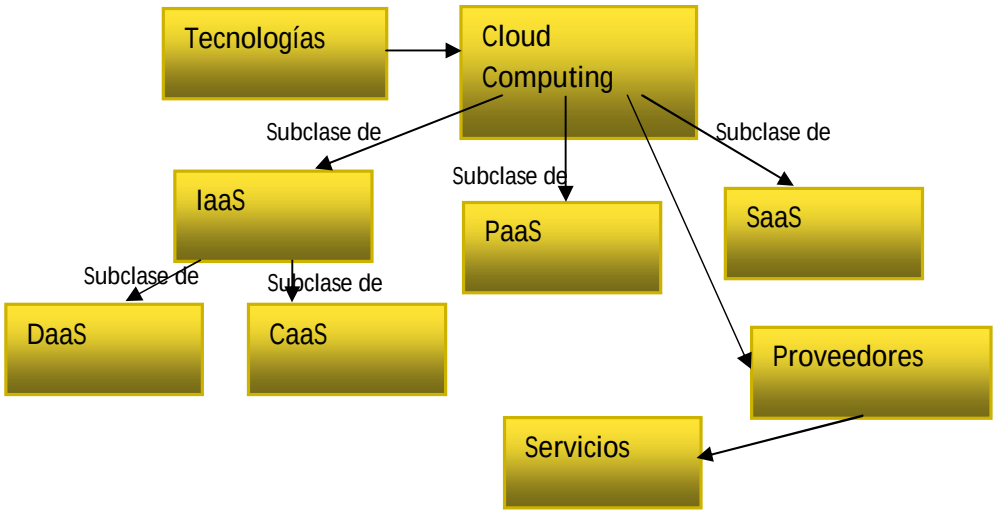
A. Elaborar un glosario de términos importantes para la ontología

Por ejemplo términos importantes relativo al dominio de Cloud Computing son: cloud, niveles, características, IaaS, PaaS, SaaS, ventajas, desventajas, proveedores, servicios, tecnologías, virtualización, escalabilidad, "on-demand", Amazon, SalesForce, disminución de costes...

B. Construir una taxonomía de conceptos

Seleccionamos del glosario aquellos términos que son conceptos. A partir de ellos construimos la taxonomía que define la jerarquía entre los conceptos del dominio. Para ello realizaremos un proceso de **desarrollo top-down** comienza con la definición de los

conceptos más generales en el dominio la subsecuente especialización de los conceptos. De esta manera comenzamos creando clases para el concepto más general de Cloud Computing. Luego especializamos las clases de Cloud: IaaS, PaaS, SaaS. Dentro de IaaS las subclases de DaaS , CaaS. Finalmente implementamos los conceptos del dominio: Tecnologías , Proveedores y Servicios.



C.DESCRIBIR LAS RELACIONES BINARIAS Y DE ENTIDADES

Nombre de la relación	Concepto origen	Cardinalidad máxima	Concepto destino	Relación inversa
Nivel_cloud	Cloud	N	IaaS	Es nivel de
Nivel_cloud	Cloud	N	PaaS	Es nivel de
Nivel_cloud	Cloud	N	SaaS	Es nivel de
Tipo_IaaS	Iaas	N	CaaS	Es un tipo de
Tipo_IaaS	Iaas	N	DaaS	Es un tipo de
Tecnologia_Cloud	Cloud	N	Tecnología	Es una tecnología
Proveedor	Cloud	N	Proveedor	Es un proveedor
Servicio	Proveedor	N	Servicio	Es un servicio de

D.DEFINIR LAS PROPIEDADES DE LAS CLASES:SLOTS

Las clases aisladas no proporcionan información a las preguntas de competencia del que hemos descrito en el apartado anterior a las que tienen que responder esta ontología. Una vez que hemos definido algunas de las clases, debemos describir
la estructura interna de los conceptos. Los términos resultan propiedades de las clases definidas. Así para definir las propiedades que caracterizan cada uno de los conceptos que se contienen en cada clase tenemos:

CLOUD COMPUTING	
Slot	**Tipo**
Denominacion	Texto
Descripcion	Texto
Caracteristicas	Texto
Ventajas	Texto
Inconvenientes	Texto
Modelos	{Publica,Privada,Mixta}

IaaS(Subclase de Cloud)	
Slot	**Tipo**
Hereda todos los slots de Cloud	Texto
Proveedor_IaaS	Instancia de Proveedores

DaaS(Subclase de IaaS)	
Slot	**Tipo**
Hereda todos los slots de IaaS	Texto

CaaS(Subclase de IaaS)	
Slot	**Tipo**
Hereda todos los slots de IaaS	Texto

PaaS(Subclase de Cloud)	
Slot	**Tipo**
Hereda todos los slots de Cloud	Texto
Proveedor_PaaS	Instancia de Proveedores

SaaS(Subclase de Cloud)	
Slot	**Tipo**
Hereda todos los slots de Cloud	Texto
Proveedor_IaaS	Instancia de

	Proveedores

TECNOLOGIA	
Slot	**Tipo**
Denominacion	Texto
Descripcion	Texto
Uso	Texto

PROVEEDORES	
Slot	**Tipo**
Denominacion	Texto
Descripcion	Texto
Pais	Texto
Fundacion	Año
Servicios_Ofertados	Instancia de Servicios

SERVICIOS	
Slot	**Tipo**
Denominacion	Texto
Descripcion	Texto
Tecnlogia_Servicios	Instancia Tecnologia

E.Definir las facetas de los slots

Los slots puedes tener diferentes facetas que describen el tipo de valor, valores admitidos, el número de los valores (cardinalidad), y otras características de los valores que los slots pueden tomar.

CLOUD COMPUTING			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Denominacion	Texto	....	(1,1)
Descripcion	Texto	...	(1,1)
Caracteristicas	Texto	...	(1,1)
Ventajas	Texto	...	(1,1)
Inconvenientes	Texto	...	(1,1)
Modelos	{Publica,Privada,Mixta}	{Publica,Privada,Mixta}	(1,3)

IaaS(Subclase de Cloud)			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Hereda todos los slots de Cloud	Texto	....	(1,1)
Proveedor_IaaS	Instancia de Proveedores	...	(1,*)

DaaS(Subclase de IaaS)			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Hereda todos los slots de IaaS	...	....	...

CaaS(Subclase de IaaS)			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Hereda todos los slots de IaaS	Texto	....	...

PaaS(Subclase de Cloud)			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Hereda todos los slots de Cloud	Texto	....	...
Proveedor_PaaS	Instancia de Proveedores	...	(1,*)

SaaS(Subclase de Cloud)			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Hereda todos los slots de Cloud	Texto	...	....
Proveedor_IaaS	Instancia de Proveedores	...	(1,*)

TECNOLOGIA			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Denominacion	Texto	...	(1,1)
Descripcion	Texto	...	(1,1)
Uso	Texto	...	(1,1)

PROVEEDORES			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Denominacion	Texto	...	...
Descripcion	Texto	...	...
Pais	Texto	...	...
Tamano	{Grande,Mediano,Pequeño}	{Grande,Mediano,Pequeño}	{1,3}
Servicios_Ofertados	Instancia de Servicios	....	{1,*}

SERVICIOS			
Slot	**Tipo**	**Rango de valores**	**Cardinalidad**
Denominacion	Texto	....	...
Descripcion	Texto	...	..
Tecnlogia_Servicios	Instancia Tecnolo	....	{1,*}

F.DEFINIR AXIOMAS FORMALES

A continuación se identifican los axiomas formales que nos sirven para determinar restricciones en la ontología y que resultan imprescindibles para determinarlos de manera precisa. Los axiomas son expresiones lógicas que son siempre verdaderas y su determinación nos ayudara a crear las restricciones de los conceptos de la ontología.

Nombre del axioma	Descripción	Expresión	Conceptos	Relaciones	Variables
Incompatibilidad entre servicios de Proveedores Cloud Computing	Un servicio prestado por un proveedor sólo puede ser o IaaS o Paas o SaaS	no (existe(?X,?Y) (proveedor (?X) y servicio(?Y) y proveedor_IaaS(?Y,?X) y Proveedor_PaaS(?Y,?X) Proveedor_SaaS(?Y,?X)))	Proveedor Servicio IaaS, PaaS, SaaS	IaaS, PaaS SaaS Proveedor Servicio	?X ?Y

G. DEFINIR REGLAS

En este paso de diseño de la ontología se debe identificar qué reglas (usadas generalmente para inferir conocimiento en la ontología) se necesitan en la ontología, y posteriormente definir y describir esas reglas en la tabla de reglas.

METHONTOLOGY propone especificar las expresiones de las reglas utilizando el formato *si <condiciones> entonces <consecuencias o acciones>*.

Es esta fase del proyecto y dado que todavía no hemos procedido a poblar la ontología no encontramos reglas que apliquen al dominio elegido. Este hecho no es óbice para que en el futuro cuando se proceda a poblar la ontología se desarrollen reglas que ayuden a inferir conocimiento en esa ontología.

H. CREAR INSTANCIAS

Este es el último paso que consiste en crear instancias individuales de clases en la jerarquía. La definición de una instancia individual de una clase requiere elegir una clase, crear una instancia individual de la clase y rellenar los valores del slot. Por ejemplo, podemos crear una instancia individual con los valores para representar un proveedor concreto de SaaS que sería de la siguiente manera:

Proveedor

- **Denominación_Proveedor:** SalesForce
- **Pais:** EEUU
- **Tamaño**: Grande
- **Servicios_Ofertado:** Instancia de Servicios

Servicio

- **Denominación_Servicio**: SalesCloud

- **Descripcion:_Servicio:** Herramienta de ventas que ofrece a los representantes, gestores y ejecutivos todo lo que necesitan para conectar con los clientes y centrarse en lo que les sea más importante: más ventas y menos administración. Pemite:

    - Un rápido cierre de contratos
    - Una visibilidad en tiempo real de las ventas
    - Conexión con los clientes sociales de la actualidad

- **Tecnología_Servicio:** Instancia de tecnología

Tecnología

- **Denominación_Tecnología:** Cloud_Services_Apps
- **Descripción_Tecnología**: Servicio de aplicaciones software desde internet.
- **Uso**:Se distribuyen aplicaciones de negocio desde internet:CRM,Correo,Ofimática, aplicaciones Colaborativas.

4.12IMPLEMENTACION DE LA ONTOLOGÍA

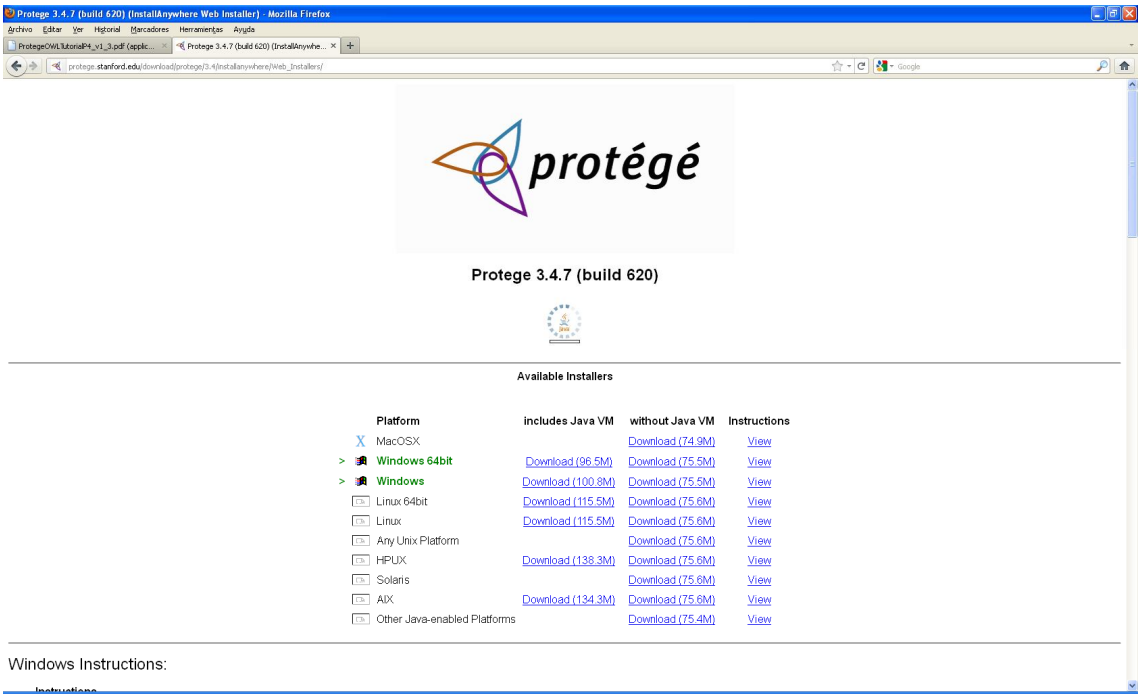
4.12.1 DESCRIPCION DEL EDITOR PROTEGE

Para la construcción de la ontología se utiliza el editor de Ontologías Protegé.Se trata de un Editor de ontologías y bases de conocimiento gratis y abierto. Entre sus características podemos destacar las siguientes:

- Basado en Java
- Soporta Frames, XML Schema, RDF y OWL
- Cuenta con un ambiente “plug-and-play”

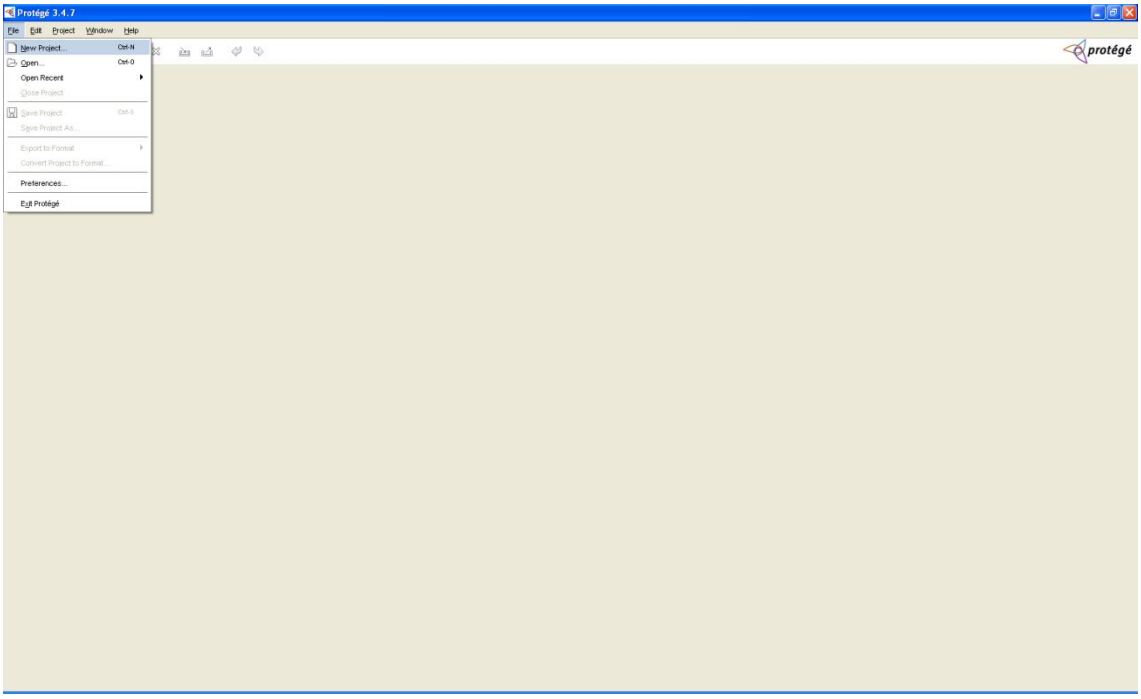
Para proceder a la instalación de Protegé nos descrgamos la última versión estable desde la web del editor: http://protege.stanford.edu/

La versión a descargar es la 3.4.7 como observamos en la siguiente imagen:



4.12.2CREACION DE CLASES EN PROTEGE

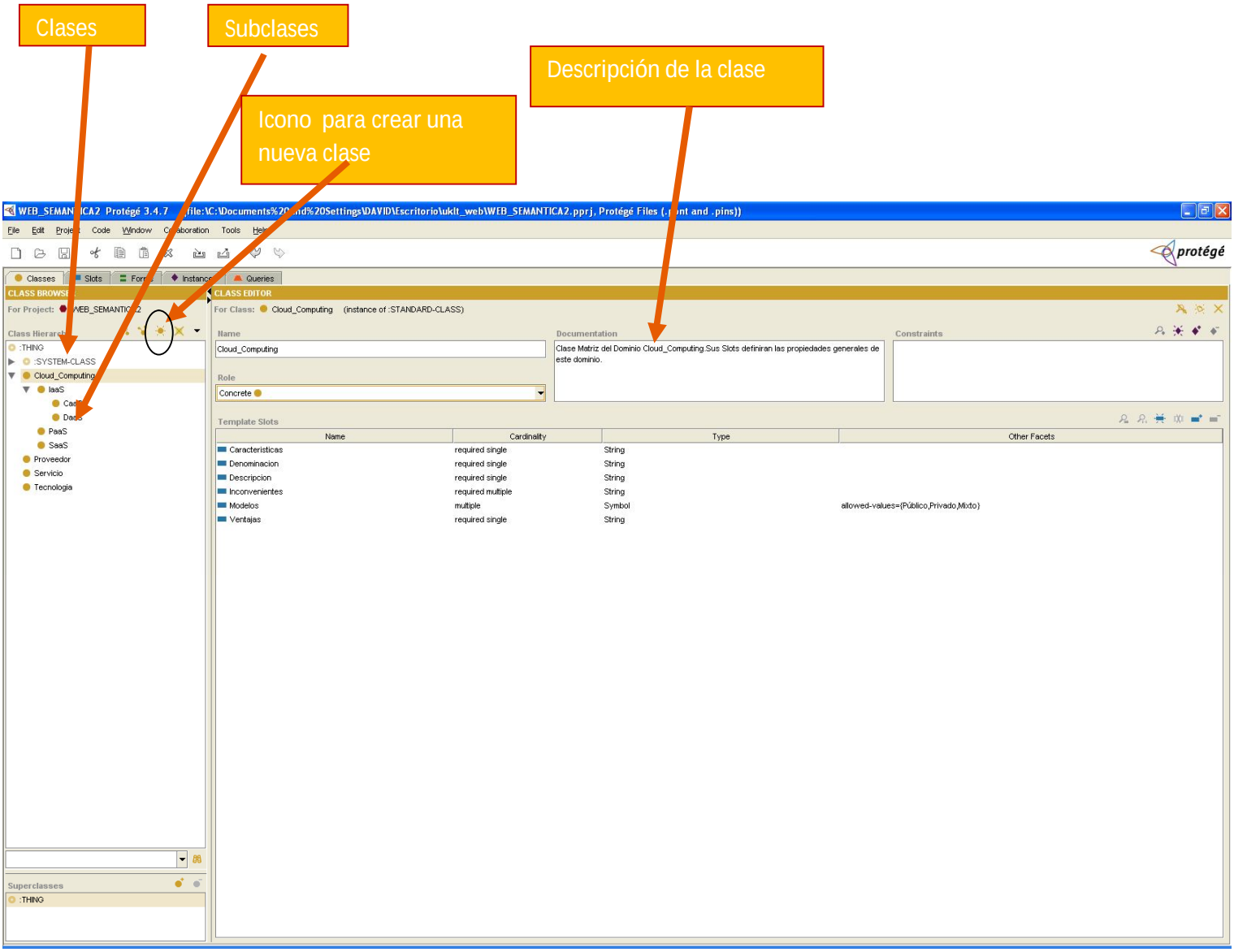
En primer lugar se crea un proyecto de Protegé desde la ventana de inicio o desde el menú File---New Project. Tenemos que darle un nombre a ese proyecto, en nuestro caso lo hemos denominado Web Semántica.



La creación de clases en Protege se realiza desde la pestaña de clases, pulsando en la opción ***create class***.Al crear la clase se tiene que dar el nombre de la misma, su rol(abstracta o concreta), poner la descripción y documentación de la clase. Asimismo al crear la clase se puede crear como clase principal o como subclase de , según se seleccione

previamente desde el nivel superior que es el de la clase THING o desde el nivel de la clase que va a ser padre(ejemplo se selecciona la clase IaaS al crear las clases CaaS y DaaS)

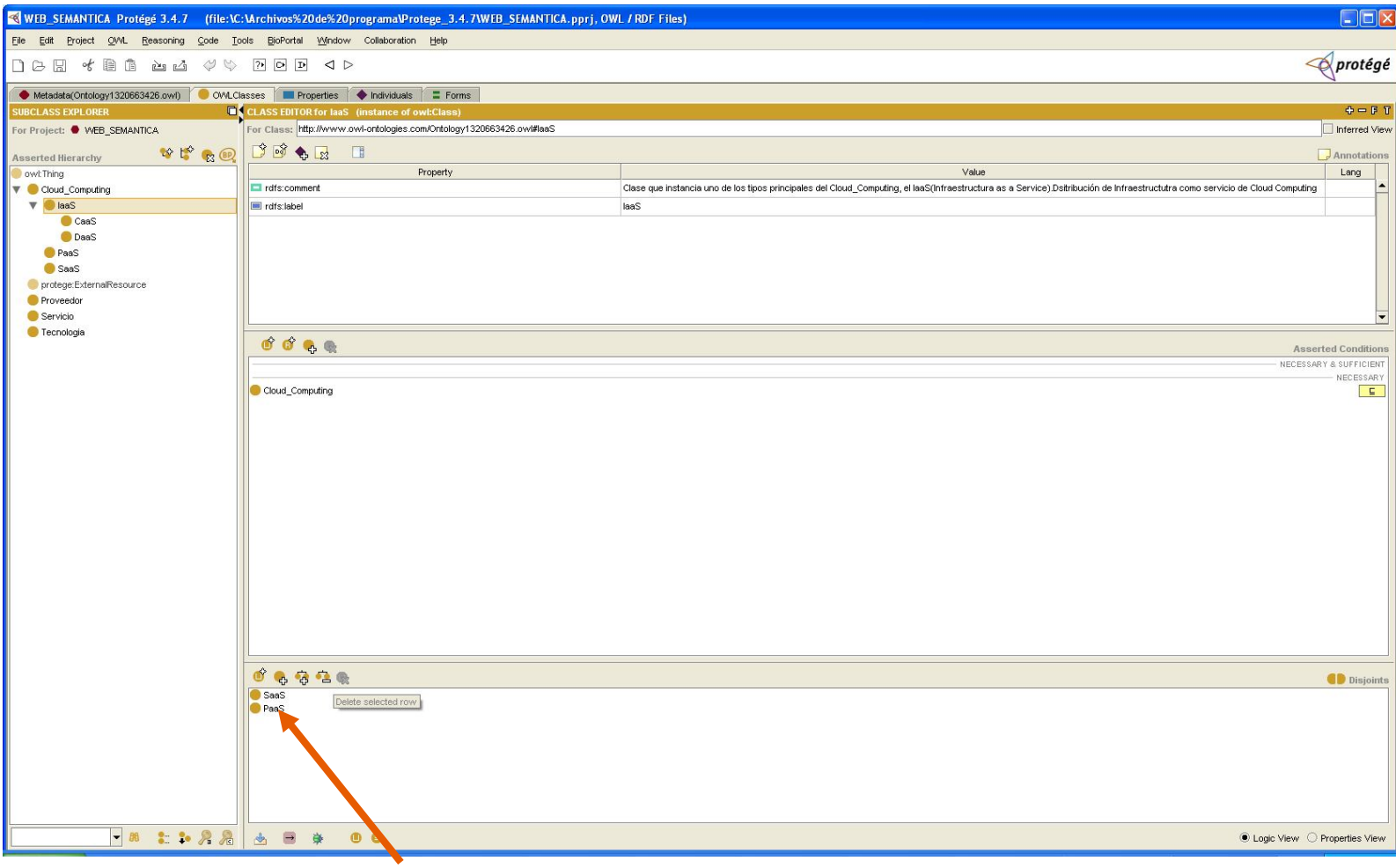
En la siguiente imagen se muestra la creación en Protege de las clases que componen la ontología del dominio Cloud Computing , destacando cada uno de sus componentes.



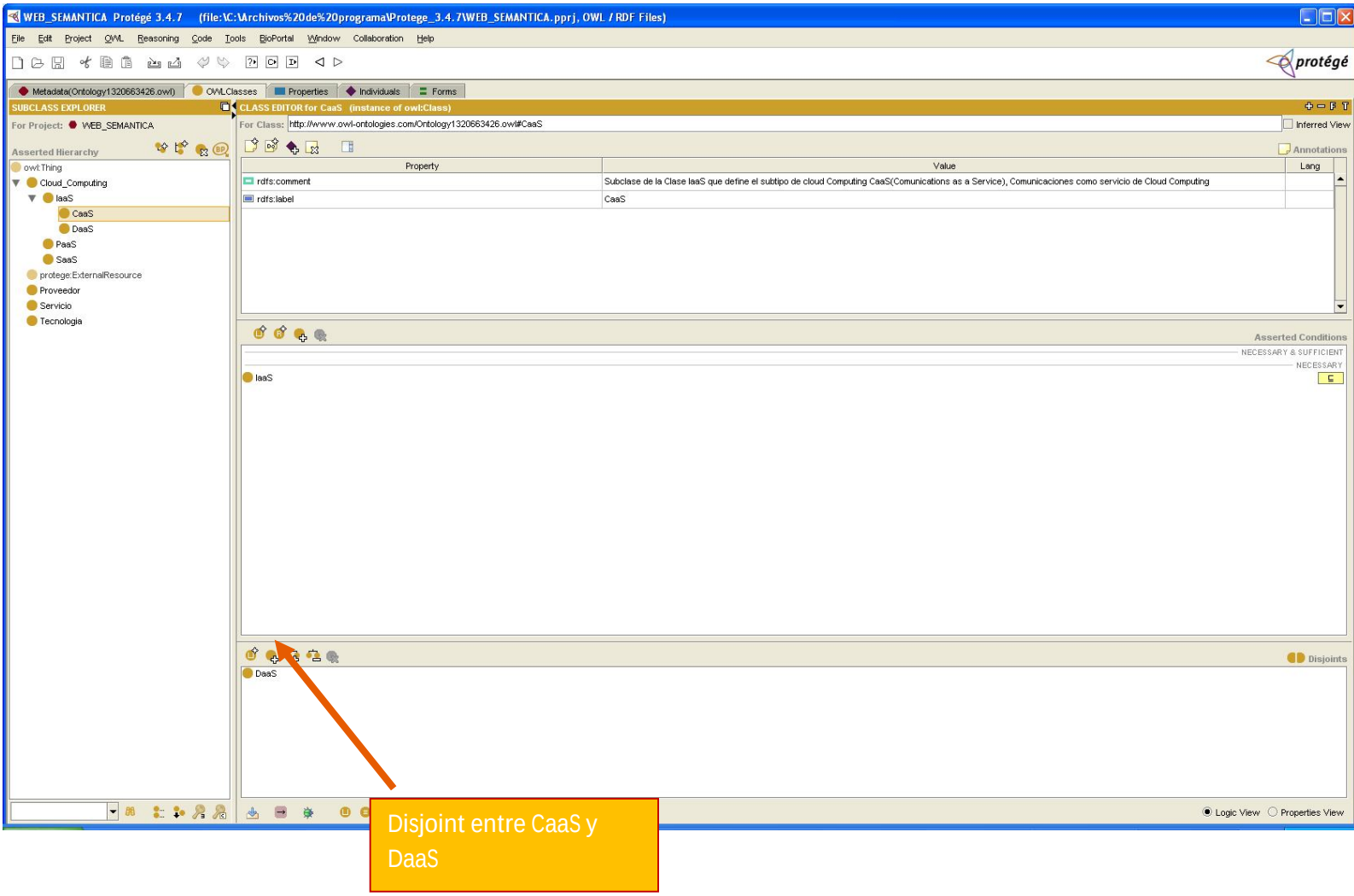
4.12.3CREACION DE DISJOINT EN LAS CLASES EN PROTEGE

Ningún individuo (objeto) puede ser un caso de más de una de un tipo de clases. 'Add sibblings'.En nuestro caso por ejemplo: entre PaaS, IaaS, SaaS y a nivel de IaaS entre CaaS y DaaS.

En nuestro caso se pulsa el botón de 'Add sibblings' una vez hemos seleccionado IaaS y en la ventana de disjoint aparecerán las otras clases PaaS y SaaS. Si se selecciona PaaS aparecerán en disjoint IaaS y Saas y si selecciona SaaS aparecerá IaaS y Paas como clases disjoint.



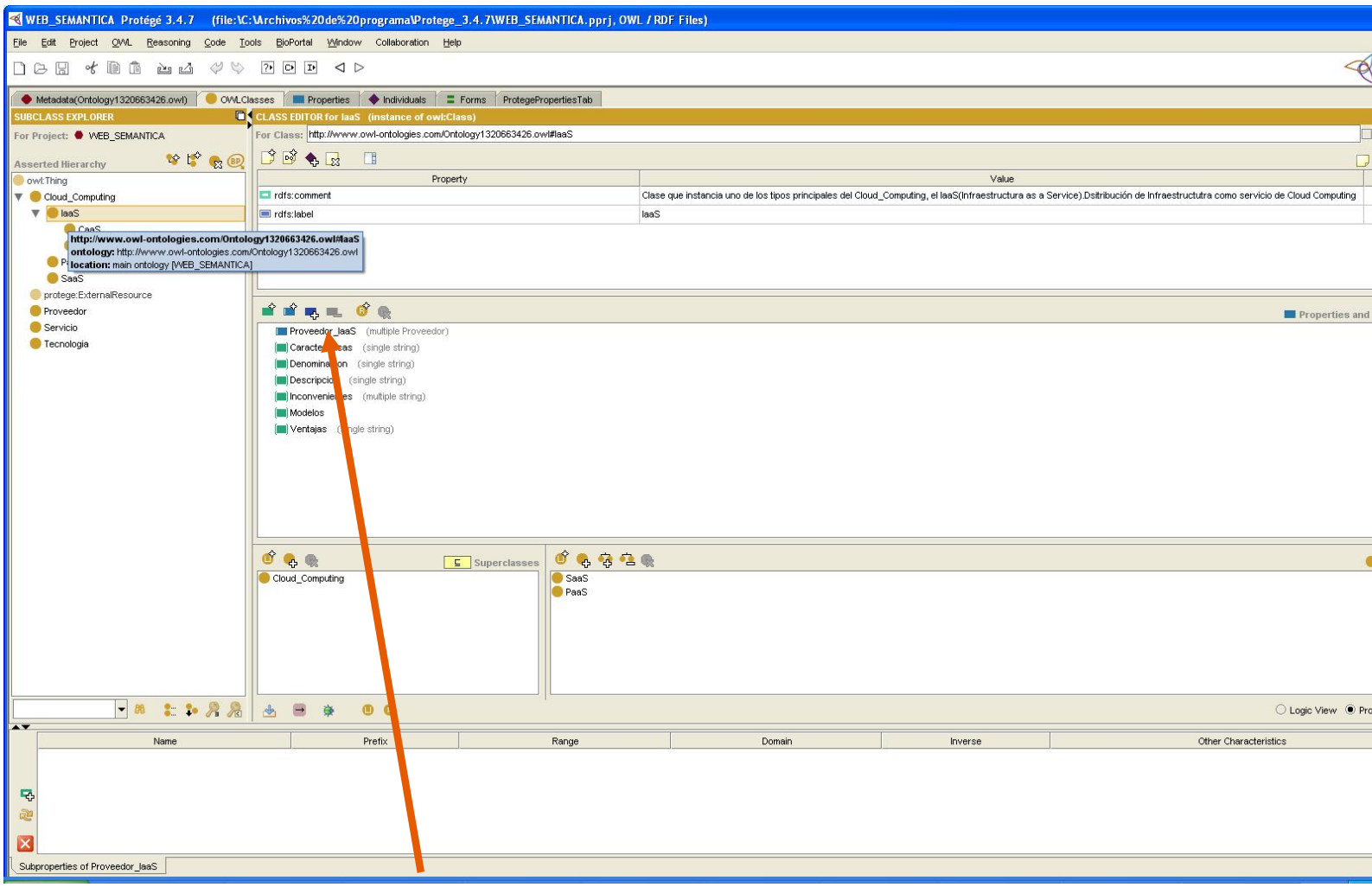
Disjoint entre IaaS, Paas y SaaS



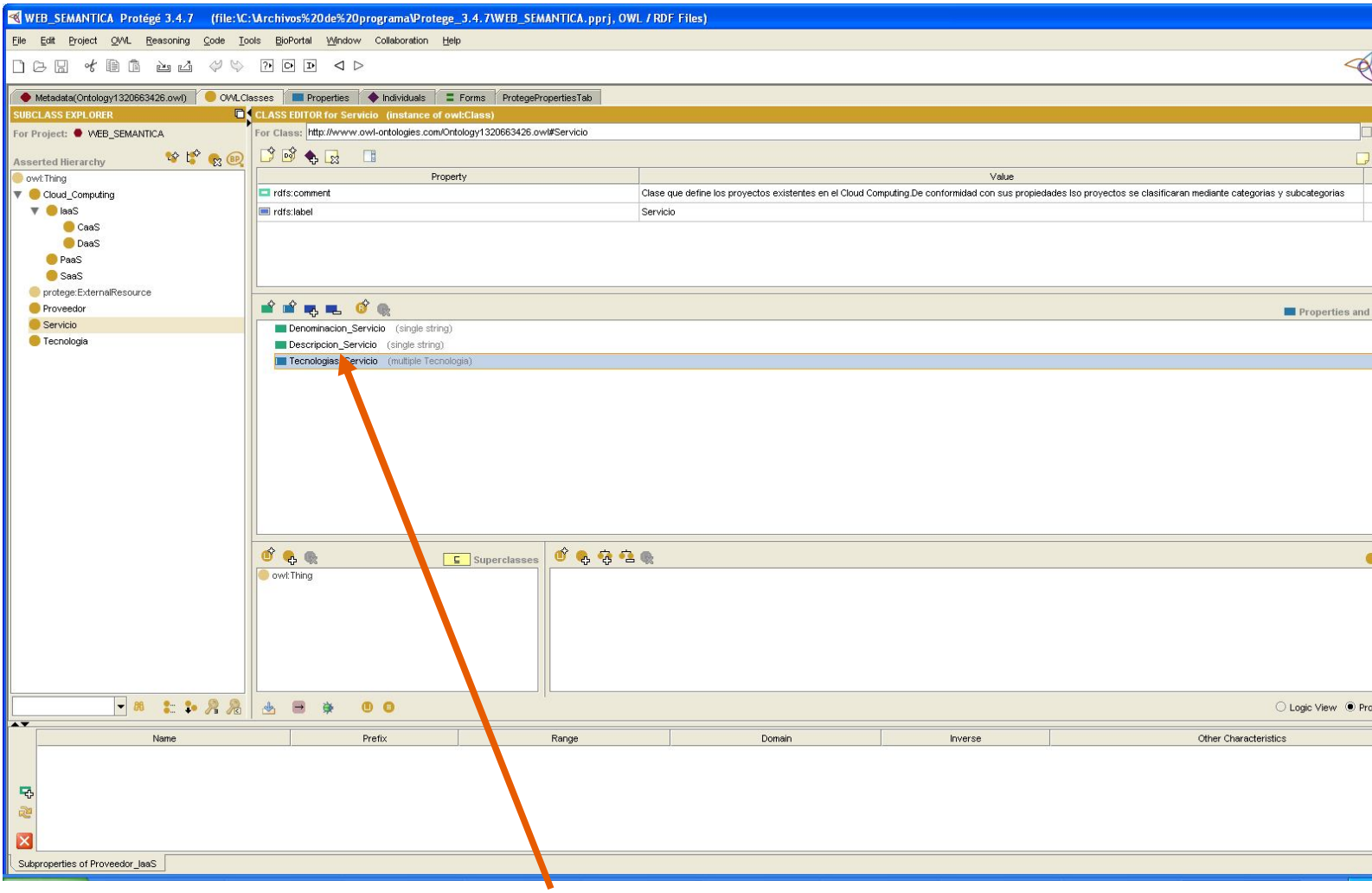
4.12.4CREACIÓN DE PROPIEDADES OWL

Son relaciones entre dos objetos ligando un objeto a un valor de tipo de dato XML Schema o literal RDF. Son usadas para agregar información (metadatosdatos acerca de datos) a las clases, individuos y propiedades de objeto o tipo de dato) Se realiza desde la pestaña properties y cada propiedad puede tener a su vez una propiedad inversa. Las propiedades que vamos a añadir en esta ontología son:

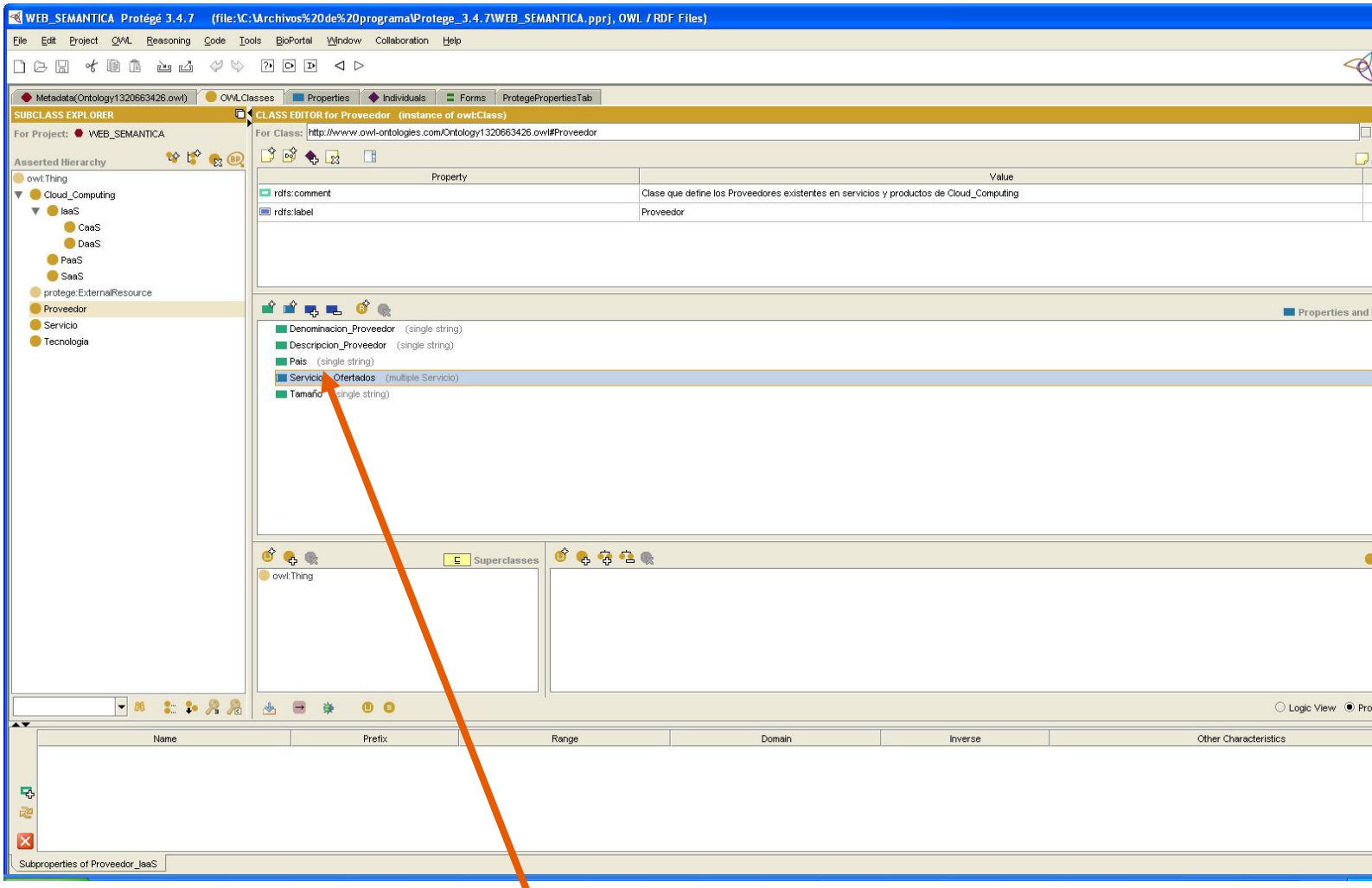
- Es proveedor de
- Es un servicio de
- Es una tecnología de



Propiedad es Proveedor de



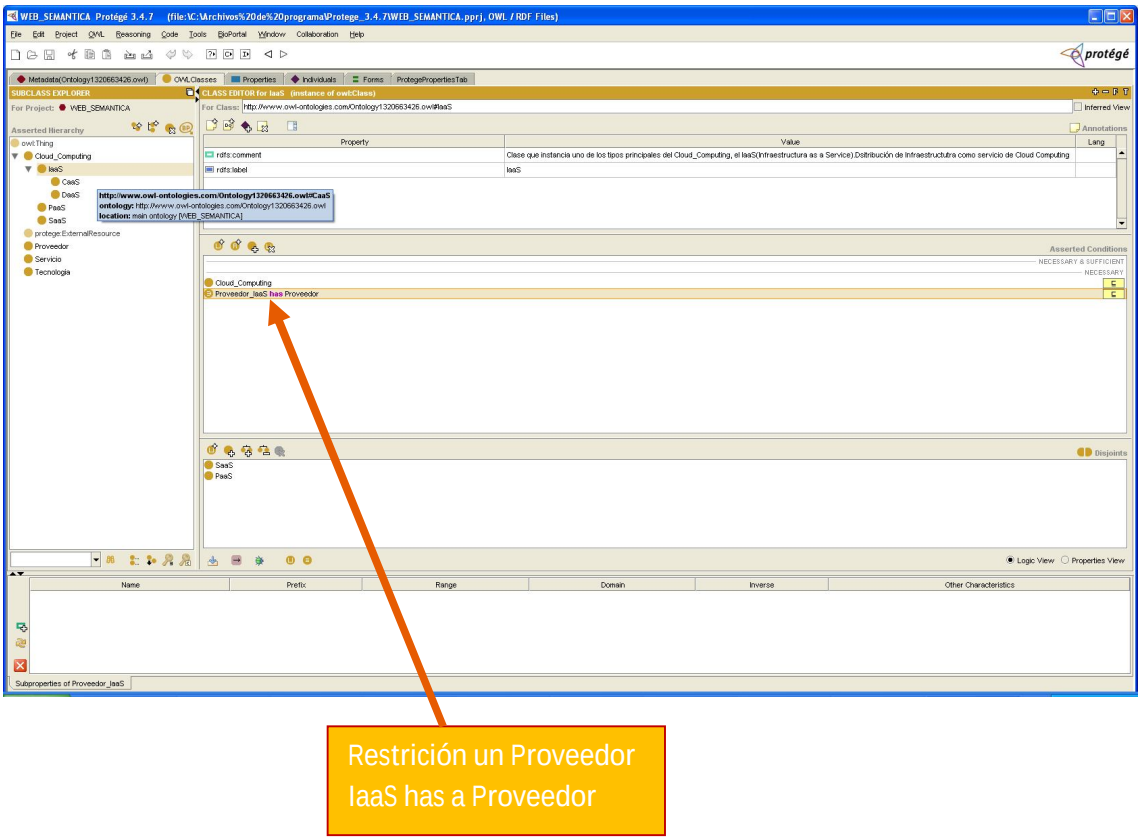
Propiedad es tecnología de



4.12.5 CREACION DE RESTRICCIONES

Se establece desde las ventanas restricciones. Podemos distinguir entre restricciones existenciales(alguno) y restricciones universales(todos).Como restricciones para el conjunto de conceptos del dominio, exponemos las siguientes:

- Un objeto IaaS tiene que tener algún proveedor IaaS
- Un objeto PaaS tiene que tener algún proveedor PaaS
- Un objeto SaaS tiene que tener algún proveedor SaaS
- Un proveedor tiene que tener algún servicio
- Un servicio tiene que tener alguna tecnología



5. CREACION DE PARSER PARA POBLAR LA ONTOLOGÍA

5.1DESCRIPCIÓN Y FUNCIONALIDADES DEL PARSER

Como uno de los requisitos de entrega de este proyecto de investigación de la web semántica se solicita la creación de un **parser o analizador sintáctico** que mediante la conexión con Wikipedia o DBPedia rellene con instancias la ontología creada anteriormente. Implementación de este parser que posibilite poblar con instancias la ontología de Cloud Computing utilizando como base los conceptos y atributos propios de la ontología. Una descripción más exhaustiva de las funcionalidades de este parser serían:

- Conectarse a un documento web de Wikipedia o DBPedia sobre el dominio elegido
- Extraer el conocimiento de ese documento web
- Rellenar con ese conocimiento a la estructura de datos de la ontología creada y modelada con anterioridad
- Explotar esa Ontología

5.2TECNOLOGIAS A UTILIZAR

Para el desarrollo de este parser se utilizaran las siguientes tecnologías:

5.2.1 RDF

Ya descrito en este documento RDF es la abreviatura de *Resource Description Framework que* es una DTD (definición del tipo de documento) de XML .Una aplicación de metadatos que utiliza XML a fin de proporcionar un marco estándar para la interoperabilidad en la

descripción de contenidos web. Se puede definir como un modelo de datos para objetos (recursos) y las relaciones entre ellos. **(11)**
El RDF surge en agosto de 1997 el seno del Consorcio Web W3C, cuya actividad en relación con los metadatos está apoyada por protagonistas muy influyentes en la escena industrial, tales como creadores de navegadores —Netscape, Microsoft— y motores de búsqueda. Se nutre de los trabajos de varios colectivos como otras iniciativas del W3C —PICS para el control de contenidos o P3P destinado a salvaguardar la privacidad en la web— y por supuesto, de los trabajos de la comunidad bibliotecaria en torno al Dublin Core (DC) que es uno de los modelos de metainformación que primero ha adoptado la sintaxis del RDF. En la fecha de realización de esta comunicación, y desde febrero de 1999, la especificación del modelo y la sintaxis de RDF —tras muchos borradores de trabajo1 — es ya una recomendación del Consorcio Web [W3C-RDF-R], y su esquema es, desde marzo de este mismo año, una propuesta de recomendación [W3C-RDFS-PR]. **(12)**

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

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 pues, un modelo formal para la representación de las propiedades y los valores de esas propiedades .Este modelo lo podemos describir a través de la siguiente figura:

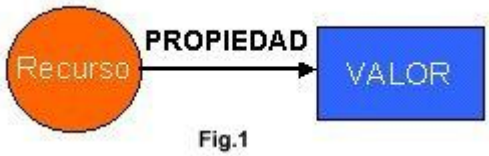


Fig.1

Según esto, el modelo de datos que propone RDF consiste en tres tipos de objetos [W3C-RDF-R]:

- **Recursos***:* cualquier objeto web identificable unívocamente por un URI, es decir, un identificador uniforme de recursos como un URL. Un recurso puede ser un documento HTML; una parte de una página web como por ejemplo un elemento HTML o XML dentro de un documento fuente, una colección de páginas, un sitio web completo; y en síntesis, cualquier recurso entendido como objeto de información.
- ***Propiedades****:* son aspectos específicos, características, atributos o relaciones utilizadas para describir recursos. Cada tipo de propiedad tiene sus valores específicos, define los valores permitidos, los tipos de recursos que puede describir y las relaciones que existen entre las distintas propiedades.
- ***Descripciones:*** Son el conjunto de un recurso, un nombre de propiedad y el valor de esa propiedad —sujeto, predicado y objeto, respectivamente— (la figura 1 representa un *RDF statement* o descripción RDF).

Profundizando más sobre el esquema descrito detallamos los recursos siguientes que son las **clases principales** que se definen como parte del vocabulario del esquema RDF. Cada modelo RDF que se traza sobre el namespace del Esquema RDF los incluye implícitamente:

- **rdfs:Resource:** todas las cosas que se describan por expresiones RDF se denominan recursos *(resources)*, y se consideran como *instances* (objetos específicos de la categoría) de la clase **rdfs:Resource**. La clase RDF **rdfs:Resource**

representa el conjunto denominado 'Resources' en el modelo formal para RDF presentado en la sección 5 de las especificación del modelo y la sintaxis

- **rdf:Property:** representa el subconjunto de recursos RDF que son propiedades, es decir, todos los elementos del conjunto presentados como 'Propiedades' en la sección 5 de la especificación del modelo y la sintaxis.
- **rdfs:Class:** corresponde con el concepto genérico de un tipo *(Type)* o categoría *(Category)*, semejante a la noción de Clase en los lenguajes de programación orientados a objetos tales como Java. Cuando un esquema define una nueva clase, el recurso que representa esa clase debe tener una propiedad **rdf:type** cuyo valor es el recurso **rdfs:Class**. Las clases RDF pueden definirse para representar cualquier cosa, como páginas *web*, personas, tipos de documentos, bases de datos o conceptos abstractos.

El esquema RDF es, básicamente, un conjunto de declaraciones que definen clases y propiedades. Así unos ejemplos de las clases de objetos que pueden ser dichas con una mezcla de RDF y vocabularios de RDF Schema son las siguientes:

- La URI puede considerarse (rdf:type) una clase (rdfs:Class) de una propiedad (rdf:Property).
- Se puede indicar una etiqueta legible por humanos (rdfs:label) o comentario (rdfs:comment). Hay muchas formas de visualizar RDF de una forma "amigable" para los seres humanos
- La URI se define por (rdfs:isDefininedBy)
- Esta clase es una subclase de esta otra (rdfs:subClassOf)
- Esta propiedad es una subpropiedad de esta otra (dfs:subPropertyOf)
- Esta propiedad conecta esta clase de sujetos (rdfs:domain) con esta clase de objetos (rdfs:range)

El *schema* o vocabulario empleado por RDF se puede resumir de la siguiente forma:

Clases RDF:

Nombre de a Clase	Comentario
rdfs:Resource	La clase de recurso, cada uno.
rdfs:Literal	La clase del valor literal, por ejemplo, cadenas de texto y números enteros.
rdf:XMLLiteral	La clase de los valores literales de los valores literales XML.
rdfs:Class	La clase de las clases.
rdf:Property	La clase de las propiedades RDF.
rdfs:Datatype	La clase de los tipos de datos RDF.
rdf:Statement	La clase de las declaraciones RDF.
rdf:Bag	La clase de los contenedores desordenados.
rdf:Seq	La clase de los contenedores ordenados.
rdf:Alt	La clase de los contenedores de alternativas.
rdfs:Container	La clase de los contenedores RDF.
rdfs:ContainerMembershipProperty	La clase de las propiedades de los miembros contenedores, rdf:_1, rdf:_2, ..., todas ellas son subpropiedades de 'miembro'.
rdf:List	La clase de las listas RDF.

Propiedades RDF:

Nombre de la Propiedad	Comentario	Domain (Dominio)	Range (Rango)
rdf:type	El sujeto es una instancia de una clase.	rdfs:Resource	rdfs:Class
rdfs:subClassOf	El sujeto es una subclase de una clase.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	El sujeto es una subpropiedad de una propiedad.	rdf:Property	rdf:Property
rdfs:domain	Un dominio de la propiedad del sujeto.	rdf:Property	rdfs:Class
rdfs:range	Un rango de la propiedad del sujeto.	rdf:Property	rdfs:Class
rdfs:label	Un nombre para el sujeto legible por seres humanos.	rdfs:Resource	rdfs:Literal
rdfs:comment	Una descripción del recurso sujeto.	rdfs:Resource	rdfs:Literal
rdfs:member	Un miembro del recurso sujeto.	rdfs:Resource	rdfs:Resource
rdf:first	El primer item en la lista RDF del sujeto.	rdf:List	rdfs:Resource
rdf:rest	El resto de la lista RDF del sujeto después del primer item.	rdf:List	rdf:List
rdfs:seeAlso	Más information sobre el recurso sujeto.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	La definición del recurso sujeto.	rdfs:Resource	rdfs:Resource
rdf:value	Propiedad idiomática usada para valores estructurados .	rdfs:Resource	rdfs:Resource
rdf:subject	El sujeto de la declaración RDF del sujeto.	rdf:Statement	rdfs:Resource
rdf:predícate	El predicado de la declaración RDF del sujeto.	rdf:Statement	rdfs:Resource
rdf:object	El objeto de la declaración RDF del sujeto.	rdf:Statement	rdfs:Resource

A continuación se presenta un ejemplo de documento RDF con HTML 4.0(sintáxis abreviada)

Ejemplo: RDF en un documento HTML4.0 (sintaxis abreviada)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs = "http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
      xmlns:dc = "http://purl.org/dc/elements/1.0/">
<HEAD>
<TITLE>Resource Description Framework (RDF) Schema Specification</TITLE>
<STYLE TYPE="text/css"> .EXAMPLE { margin-left: 1em } </STYLE>
<LINK rel="stylesheet" type="text/css" media="screen" href="/StyleSheets/TR/W3C-
PR">

<rdf:RDF>
<rdf:Description about=""

      xmlns:ddc="http://purl.org/net/ddc#"
      dc:Title="Resource Description Framework (RDF) Schema Specification"
      dc:Description="The Resource Description Framework (RDF) is a foundation for
      processing metadata; it provides interoperability between applications that
      exchange machine-understandable information on the Web.  RDF emphasizes
      facilities to enable automated processing of Web resources."
```

```
        dc:Publisher="World Wide Web Consortium"
        dc:Date="1999-03-03"
        dc:Format="text/html"
        dc:Type="technical specification"
        dc:Language="en">
        <dc:Subject resource="http://purl.org/net/ddc/025.30285"/>
        <dc:Subject resource="http://purl.org/net/ddc/025.316"/>
        <dc:Subject ddc:Class="025.302855741"
        ddc:Heading="Applications of computer file organization and access
        methods"/>
        <dc:Creator>
            <rdf:Bag rdf:_1="Dan Brickley"
                rdf:_2="R.V. Guha" /></dc:Creator>
        <rdfs:seeAlso  rdf:resource="http://www.w3.org/1999/.status/PR-rdf-schema-
        19990303/status"/>
</rdf:Description>
</rdf:RDF>
</HEAD>
```

5.2.2 DBPedia.

DBpedia es un proyecto para la extracción de datos de Wikipedia para proponer una versión Web semántica. Este proyecto está realizado por la Universidad de Leipzig, Universidad Libre de Berlín y la compañía OpenLink Software. Ultima versión DBpedia 3.7. 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 (ya que el contenido se basa en la Wikipedia).
La información se almacena con el Resource Description Framework, podemos hacer consultas a la base de datos a través de SPARQL.

El motor de extracción de datos se realiza con Scala un software libre publicado bajo el GNU General Public License. Su código fuente se distribuye: se alberga en sourceforge y disponible a través de Subversion.

5.2.2.1 El Dataset de DBpedia

Los artículos de Wikipedia consisten sobre todo en el texto libre, pero también contienen diversos tipos de información estructurada, por ejemplo plantillas del infobox, la información de la clasificación, imágenes, geo-coordina y se liga a los Web pages externos. Esta información estructurada se puede extraer de Wikipedia y puede servir como base para permitir preguntas sofisticadas contra el contenido de Wikipedia.

El dataset de DBpedia describe 1.950.000 “las cosas”, incluyendo por lo menos a 80.000 personas, 70.000 lugares, 35.000 álbumes de la música, 12.000 películas. Contiene 657.000 acoplamientos a las imágenes, 1.600.000 acoplamientos a los Web pages externos relevantes, 180.000 acoplamientos externos en otros datasets de RDF, 207.000 categorías de Wikipedia y 75.000 YAGO categorías.

El proyecto de DBpedia utiliza Marco de la descripción del recurso como un modelo flexible de los datos para representar la información extraída y para publicarla en el Web. En el día septiembre de 2007, el dataset de DBpedia consiste en alrededor 103 millones de triples de RDF, que se han extraído de las versiones inglesas, alemanas, francesas, españolas, italianas, portuguéas, polacas, suecas, holandesas, japonesas, chinas, rusas, finlandesas y noruegas de Wikipedia.

El dataset de DBpedia está disponible de conformidad con Licencia libre de la documentación del GNU.

El dataset de DBpedia se liga en nivel de RDF con otro Abra los datos datasets en el Web. Esto permite a usos enriquecer los datos de DBpedia con datos de estos datasets. En el día junio de 2007, DBpedia se liga con los datasets siguientes: GeoNames, Musicbrainz, Libro del hecho del mundo de la Cia, DBLP, Proyecto Gutenberg, DBtune Jamendo y La EUROSTAT así como Censo de los E.E.U.U. datos. Vea Web site de DBpedia y W3C SWEO que liga proyecto abierto de la comunidad de los datos para los detalles sobre datasets ligados.

SPARQL *SPARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage*. SPARQL es el estándar en el que está trabajando actualmente W3C, un lenguaje que nos permite obtener información de grafos RDF. Se adapta totalmente a las necesidades de RDF. Algunas de sus características nos permiten:

- Extraer información de URIs, literales y nodos vacíos.
- Obtener subgrafos RDF y Construir nuevos grafos RDF basados que información de grafos que devuelve una query.

Se trata de un lenguaje estandarizado para la consulta de grafos RDF, normalizado por el RDF Data Access Working Group (DAWG) del Word Wide Web Consortium (W3C).
En el lenguaje SPARQL, 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.

Es de utilidad para la **recuperación y organización de información** disponiendo de funciones para la recuperación sentencias RDF cuyas características hemos visto anteriormente. Sin embargo, algunas propuestas también incluyen operaciones para el mantenimiento (creación, modificación y borrado) de datos. De esta manera con **SPARQL** los desarrolladores y usuarios finales pueden representar y utilizar los resultados obtenidos en las búsquedas a través de una gran variedad de información como son datos personales, redes sociales y metadatos sobre recursos digitales como por ejemplo música e imágenes.

El **lenguaje de recuperación SPARQL** ha sido diseñado para un uso a escala de la Web, así permite hacer consultas sobre orígenes de datos distribuidos, independientemente del formato. A la hora de recuperar información la creación de una sola consulta a través de diferentes almacenes es mejor que múltiples consultas, además de tener un coste menor y de ofrecer unos resultados mejores.

El **lenguaje SPARQL** de posee tres componentes importantes: URIs, literales y variables procedentes del lenguaje RDF.

- URIs sirven para especificar las URLs su especificaciónn es ta en

la RFC 3987.
• Literales se describen como una cadena de caracteres encerradas entre " "
• Variables estas variables son globales, además debe de ser prefijadas por "?" o "$" no formando parte del nombre de la variable.

La sintaxis de **SPARQL** es similar a la de RQL, añadiendo algunas modificaciones para facilitar el análisis sintáctico (parsing) del lenguaje.

El **lenguaje SPARQL** tiene cuatro tipos de consultas, estas con sus respectivas descripciones:

SELECT identifica las variables que aparecen en el resultado de la Query. SELECT : Devuelve todo, o un conjunto de, las variables que coinciden con el patrón de búsqueda.

Para evitar tener que incluir todo el nombre de la URI cada vez que se necesita SPARQL nos permite definir QNames o prefijos, variables que guarda la URI para identificarla a lo largo de la query. En el siguiente ejemplo puede verse la definición de una variable *genero* para traer la **descripción** de la obra El aleph, además se utilizan el operador *ORDER BY* para ordenar en sentido descendente y las cláusulas *LIMIT* para limitar el nº de registros que se obtendrán y *OFFSET* que nos trae los resultados a partir del 5º registro de salida.

```
PREFIX music: < 'http://www.semanticaudio.org/ontology/0.1#'>
SELECT DISTINCT ?Description
WHERE
{
?Description genero:name “El_Aleph” .
}
ORDER BY DESC (?Description )
LIMIT 2
OFFSET 5
```

Para querys más complejas podemos utilizar otros operadores que describimos a continuación y que posibilitan filtrar los resultados:

- **Lógicos.**
  - A||B, A&&B, ¡A, (A)
- *de comparación.* =, !=, <, >, <=, >=
- **Aritméticos.**
  - *Unarios* +A, -A
  - *Binarios (A op B)*
- **Operadores y funciones RDF.**
  - *Boleanos.* BOUND(A), isURI(A), isBLANK(A), isLITERAL(A)
  - *String.* STR(A), LANG(A), DATATYPE(A)
- **Operadores para relacionar Strings (paracido al LIKE de SQL).**
  - REGEX (String expression, pattern expression, [flags expresion])

Soporta además tipos de datos RDF boolean, string, double, float, decimal, integer y dateTime.

Para traer todos los grupos de música de la ciudad de Oslo que tengan una "z" o "Z" en su nombre y que pertenezcan a una década posterior a 1980 podemos hacer la siguiente query:

```
PREFIX music: < http://www.semanticaudio.org/ontology/0.1#'>
SELECT ?name
WHERE
{
?city music:city Oslo . FILTER (?decade > 1980)
        FILTER regex(?name, "z","i")
```

La claúsula "i" indica que no sea sensitivo a mayúsculas (case- insensitive).

SPARQL permite la definición de querys con la directive **UNION:**

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { { ?book dc10:title ?title } UNION { ?book dc11:title
    ?title } }
```

La cláusula **UNION** es un operador binario que permite combinar dos grafos.

Hemos visto ejemplos que utilizan la cláusula SELECT, SPARQL permite utilizar la cláusula **ASK** simplemente para devolver YES/NO.

También se puede utilizar **DESCRIBE** que devuelve un grafo con la información relacionada con los nodos del grafo o bien todos los metadatos asociados a un recurso (URI).

```
PREFIX music: <http://www.semanticaudio.org/ontology/0.1#>
DESCRIBE music:music0
```

Para ordenar resultados de las querys SPARQL ofrece la misma directiva que SQL: **ORDER BY**.Así por ejemplo si realizamos una consulta a DBPedia sobre lo que escribió Bart en la pizarra ordenado por temporadas la sentencia sería de la siguiente manera:

```
select  ?temporada, ?episodio, ?pizarra WHERE {
?episodio skos:subject  ?temporada.
?temporada rdfs:label ?titulo_temporada.
?episodio dbpedia2.blackboard ?pizarra.
FILTER (regex(?titulo_temporada, "The simpsons episodes, season")).
ORDER BY  ? temporada
```

SPARQL tiene además una característica que es especialmente interesante **CONSTRUCT,** que permite construir un grafo de respuesta en vez de devolvernos la típica tabla de variables y valores de SQL. En CONSTRUCT, pueden especificarse los tripletes a ser devueltos.

Un ejemplo de la sintáxis de sentencia CONSTRUCT sería:

```
PREFIX foaf: <http://lenguajerecuperacionsparql/foaf/0.1/>
```

```
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT {
<http://lenguajerecuperacionsparql.org/person#Alice> vcard:FN
?name }
```

Finalmente presentamos una tabla comparativa de lenguajes RDF donde se puede realizar un análisis de sus características y comparar ellenguaje SPARQL que acabamos de analizar con otros lenguajes RDF.La comparativa se hace a nivel de tipos de datos que trabajan y de las propiedades y funcionalidades que soportan.

	RDQL	**RDQL**	**SeRQL**	**RDFQL**	**SquishQL**	**Triple**	***Versa***
Tipos de datos	Strings,integers ,reals, dates, URI, enumerate			Strings, integers, reals, dates and URI types	Strings, integers,	Strings, numbers, URIS, lists, sets, booelans	Strings, numbers, URIS, lists, sets, booelans
Path Expression	*Si*	*Si*	*Si*	*Si*	*No*	*Si*	*Si*
Optional Path	*Restringido*	*No*	*Si*	*Si*		*No*	*Si*
Union	*Si*	*No*	*Si*	*Si*		*Si*	*Si*
Diffrence	*Si*	*No*	*Si*	*Si*		*No*	*Restringido*
Quantification	*Si*	*No*	*Si*	*Si*		*Restringido*	*No*
Aggregation	*Si*	*No*	*Si*	*Si*	*No*	*No*	*Sí*
Recursion	*No*	*No*	*No*	*Si*		*Sí*	*Sí*
Reification	*Restringido*	*Restringido*	Si	*Si*		Restringido	Restringido
Collection and containers	*Restringido*	Restringido	Restringido	Restringido	Sí	*Restringido*	*Restringido*
Namespace	*Si*	Restringido	Si	Si	Si	*No*	No
Language	*No*	No	Si	Si		No	No
Lexical Space	*Si*	Si	Si	Si		Sí	Sí
Value Space	*Si*	Restringido	Si	Si		No	No
Entailment	*Si*	Restringido	Si	Si		Restringido	No

Ahora pasamos a describir las tecnologías que permiten desarrollar aplicaciones basadas en RDF, OWL o lenguajes similares.Para construir este tipos de aplicaciones se precisan librerías para leer y procesar ontologías definidas en estos lenguajes. En primer lugar cabe destacar el framework **Jena** que permite leer, recorrer y modificar grafos tanto RDF como OWL desde un programa java. Jena permite además guardar las ontologías tanto en RDF textual como en formato de base de datos, lo que es importante para grafos muy grandes.

Otra librería conocida de similares características para RDF y OWL es **Sesame**, desarrollado en el proyecto europeo Ontoknowledge y actualmente distribuido por Administrator. Se trata de un API Java para la web semántica que permite el almacenamiento, consulta y razonamiento con RDF y RDF Schema. Casi todas las operaciones en Sesame están relacionadas con un repositorio. ¿Qué es el repositorio? Repositorio = contenedor de almacenamiento para RDF.

A continuación se analizan las características y detalles de cada uno de ellos.

5.2.3 SESAME

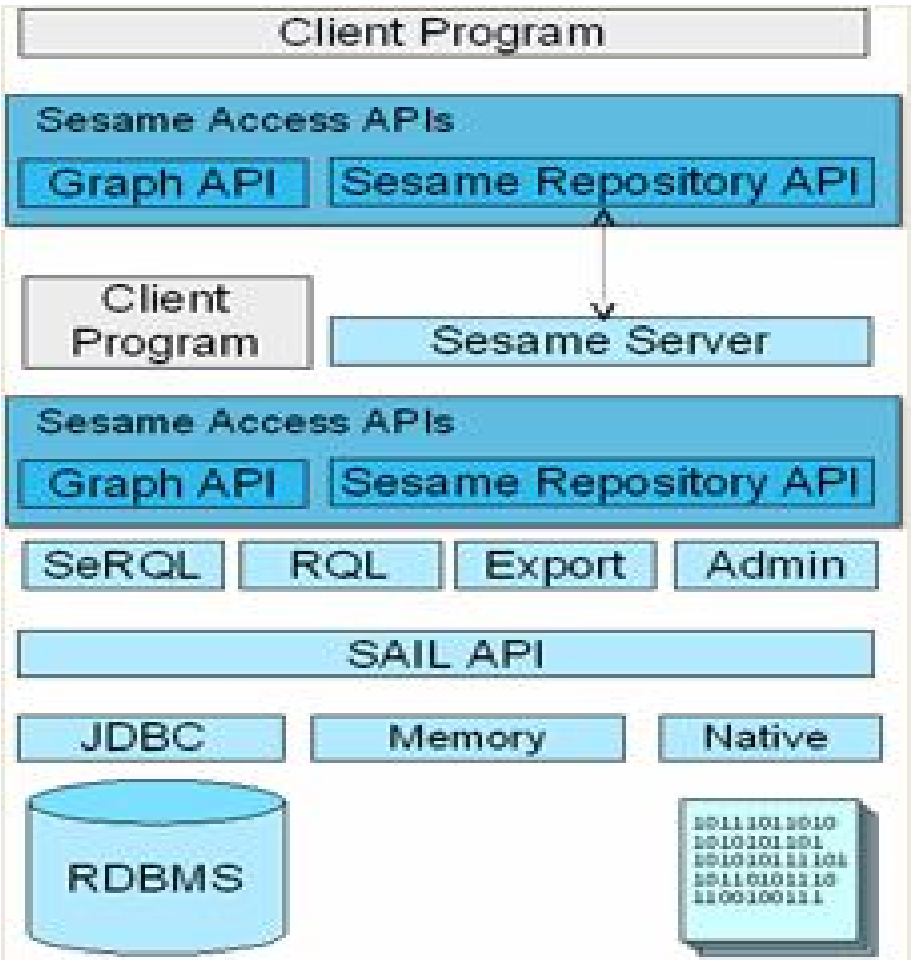
Sesame es un framework RDF, software libre desarrollado en Java, que soporta inferencia y búsqueda sobre RDF Schema y OWL. Fue diseñado para ser flexible, pudiendo trabajar sobre varios sistemas de almacenamiento (bases de datos relacionales, en memoria, en sistema de archivos, etc...) con soporte transaccional, as como ofrecer una serie de herramientas para facilitar a los desarrolladores el aprovechamiento de los beneficios de RDF y RDFS, como una API de acceso a los datos que soporta comunicación local y remota (a través de REST HTTP), varios lenguajes de consulta (de los cuales SeRQL es, según los creadores de Sesame, el más potente) y diversos formatos de entrada y salida RDF. Puede ser utilizado como servicio como componente de otros sistemas.[13]

Originariamente, Sesame fue desarrollado por Aduna Software como prototipo para el proyecto de la Union Europea On-To-Knowledge Actualmente es mantenido por Aduna en cooperación con NLnet Foundation y una comunidad de voluntarios, pertenecientes en su mayoría al foro OpenRDF. Sus desarrolladores principales son Arjohn Kampman y Jeen Broekstra.

Existen dos demostraciones on-line del producto del producto: Museum y vCard. La versión actual de Sesame 2.x se ha desarrollado con Java 5, requiriéndose varios sistemas para su funcionamiento. De los indicados a continuación, los dos primeros son para el uso de las bibliotecas de programación, mientras que el ultimo solo es necesario que Sesame trabaje como servidor.

- JRE (Java Runtime Environment) 1.5 o superior.
- SLF4J (Simple Logging Facade for Java), como fachada del sistema de autenticación.
- Un contenedor de Java Servlet que soporte Java Servlet API 2.4 y JSP
- (Java Server Pages) 2.0. Se recomienda el uso de una versión estable de Apache Tomcat; a fecha de desarrollo de la documentación del producto, versión 2.1, la versión aconsejada es 5.5.x o 6.x.

Un esquema de la arquitectura de Sesame la podemos ver en la siguiente imagen:



A partir de la parte inferior, el almacenamiento y la capa de inferencia, o API SAIL, es una API interna De Sesame que los resúmenes del formato de almacenamiento utilizado (es decir, si los datos se almacenan en un RDBMS, en la memoria, o en los archivos, por ejemplo), y proporciona apoyo razonamiento. SAIL implementaciones también pueden apilarse uno encima del otro, para ofrecer mayor funcionalidad, como caché o la manipulación de acceso concurrente. Cada repositorio De Sesame tiene su propio objeto de SAIL para que la represente.

En la parte superior de SAIL, nos encontramos con módulos funcionales De Sesame, como la SeRQL, RQL y motores RDQL consulta, el módulo de administración, y la exportación de hornos. El acceso a estos módulos funcionales está disponible a través de Access Sesame API, que consta de dos partes separadas: la API del repositorio y al API de gráficos. El repositorio ofrece acceso al API de alto nivel a los repositorios Sesame, como la consulta, el almacenamiento de archivos RDF, extracción RDF, etc. El API gráfico proporciona un apoyo más de grano fino para la manipulación de RDF, como la adición y la eliminación de las declaraciones individuales, y la creación de pequeños modelos RDF directamente desde el código. Los dos APIs se complementan entre sí en la funcionalidad, y en la práctica a menudo se utilizan juntos.

Las APIs de acceso proporcionan, acceso directo a los módulos funcionales Sesame, ya sea a un programa cliente (por ejemplo, una aplicación de escritorio que utiliza Sesame como una biblioteca), o para el siguiente componente de la arquitectura Sesame, el servidor Sesame. Este es un componente que proporciona acceso basado en HTTP a las API de Sesame. Luego, en el lado cliente remoto HTTP, volvemos a encontrar las API de acceso,

que de nuevo se puede utilizar para comunicarse con Sesame, esta vez no como una biblioteca, sino como un servidor que ejecuta en una ubicación remota.

Extensibilidad La posibilidad de ampliación de funcionalidad de Sesame está garantizada por la publicación de su código fuente. En cuanto a su integración en otros sistemas, Sesame proporciona diferentes paquetes en Java, correspondientes a las diferentes capas de su arquitectura. Estos paquetes tienen abundante comunicación que proporciona guía y soporte para su implementación. Entre estas extensiones destacamos:

OWLIM : OWLIM está implementado como una capa de almacenamiento e inferencia (SAIL) para **Sesame. La evaluación de las consultas se realiza en memoria lo que garantiza preservar datos , integridad y consistencia.**Por tanto se trata de un repositorio semántico que implementa la interfaz SAIL de Sesame, permitiendo la gestión, integración y análisis de datos heterogéneos, combinando ligeras capacidades de razonamiento. Se podría hacer la analogía de ver OWLIM como una base de datos RDF con alto rendimiento en el razonamiento. Existen dos tipos de soluciones (SwiftOWLIM y BigOWLIM) idénticas en API, sintaxis y lenguajes de consulta, pero que difieren en la su rendimiento.
Así, mientras SwiftOWLIM es bueno para experimentos y para cantidades de datos medias por su extrema rapidez en la carga de datos, BigOWLIM está diseñado para manejar grandes volúmenes de datos y consultas intensivas, de forma que posee una mayor escalabilidad con requisitos de memoria menores.

A continuación se ofrece una tabla comparativa con ambas soluciones:

	SwiftOWLIM	**BigOWLIM**
Escala (Millones de sentencias explícitas)	*10 MSt, usando 1,6GB RAM* *100 MSt, usando 16GB RAM*	*130 MSt, usando 1,6GB*
Velocidad de procesamiento (carga+ inferencia+almacenamiento)	*30 KSt/s en un portátil* *200 KSt/s en un servidor*	*5 KSt/s en un portátil* *60 KSt/s en un servidor*
Optimización de consultas	*No*	*Si*
Persistencia	*Back-up en N-Triples*	*Ficheros de datos binarios e índices*

OWLIM usa como lenguajes de consulta SPARQL, SeRQL, RQL y RDQL, y permite importar y exportar datos a XML, N3 y N-triples a través de Sesame.

5.2.4 JENA

Jena es un framework desarrollado en Java para la construcción de aplicaciones que funcionan con las tecnologías de la web semántica. Provee de un entorno de programación para el desarrollo de, RDF, RDFS y OWL; incluye también un motor de reglas de inferencia. Jena es un programa de código abierto que tuvo su origen en el Programa de Web Semántica del Laboratorio de HP. El frame work incluye:[15]

- Gestión de ontologías, almacenamiento y consultas contra ellas.
- Soporta RDF, DAML y OWL y es independiente del lenguaje.

- Problemas que presenta:
    - El recurso no cambió, pero la clase Java para modelarlo es otra.
    - No podemos modificar la clase dinámicamente.
- Incluye varios componentes:
    - ARP: un parser de RDF
    - API RDF
    - API de Ontologías para OWL, DAML y RDF Schema.
    - Subsistema de razonamiento
    - Soporte para persistencia
    - RDQL: Lenguaje de consultas de RDF

API RDF de Jena:

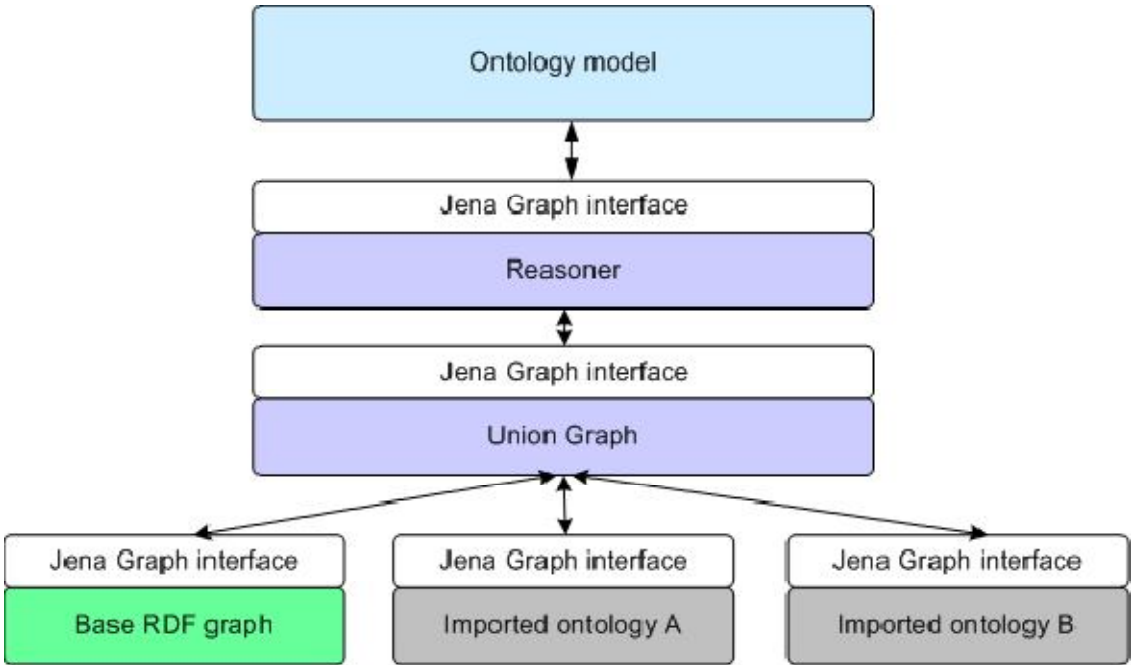
- Permite crear y manipular modelos RDF desde una aplicación Java.
- Proporciona clases java para representar:
    - Modelos: son conjuntos de statements
    - Recursos
    - Propiedades
    - Literales
    - Statements
    - Validador de OWL:
    - Posible realizar una validación básica de OWL: sólo comprueba la sintaxis, no infiere ni razona.
    - Jena 2: para validaciones más complejas.
- **Jena Inference Support:**
    - Inferir es deducir información adicional.
    - Reasoner = código que realiza la tarea de inferir.
    - Jena ofrece mecanismos para añadir nuevos razonadores e incluye un conjunto básico de éstos: OWL Reasoner, DAML Reasoner, RDF Rule Reasoner, etc.
- **RDQL (RDF Data Query Language):**
    - Lenguaje de consultas para RDF desde un enfoque declarativo.
    - Modelo RDF = conjunto de tripletas: (Objeto, Propiedad, Valor).

- Permite especificar patrones que son mapeados contra las tripletas del modelo para retornar un resultado.

- **Modelos Persistentes:**
  - Jena permite crear modelos persistentes: son mantenidos de forma transparente al usuario en una base de datos relacional.
  - Jena 2 soporta MySQL, Oracle, PostgreSQL.

- **Referencias Jena:**
  - Sitio Oficial de Jena 2: http://jena.sourceforge.net/
  - Jena 2 Ontology: http://jena.sourceforge.net/ontology/

La arquitectura de Jena la podemos ver en la siguiente imagen:



Como IDE de desarrollo del software y dado su integración con las librerías de Jena y Sesame utilizaremos Eclipse que describimos a continuación:

5.2.5. ECLIPSE

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.[14]

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

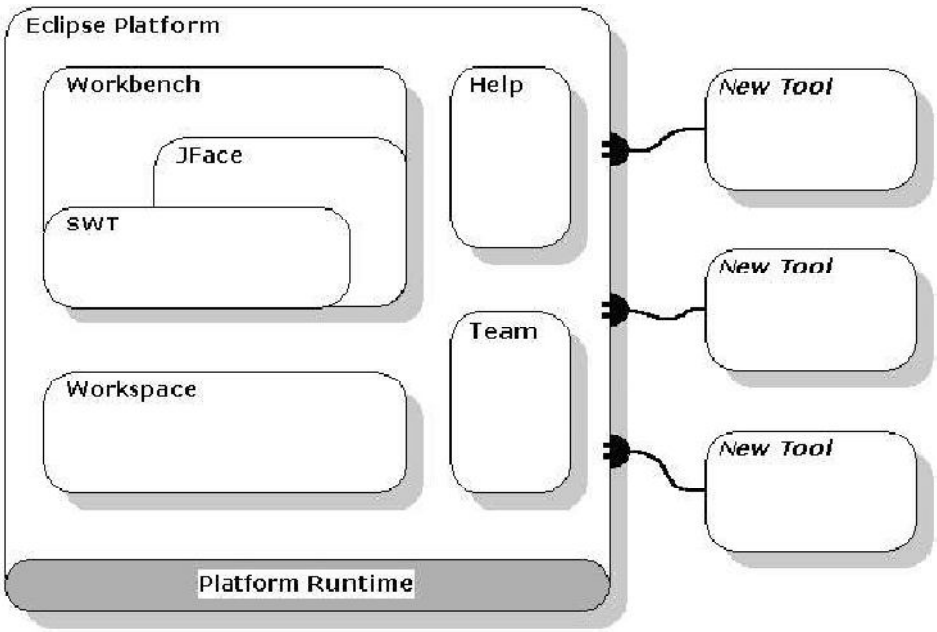
Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

La Plataforma Eclipse está diseñada para afrontar
las siguientes necesidades:

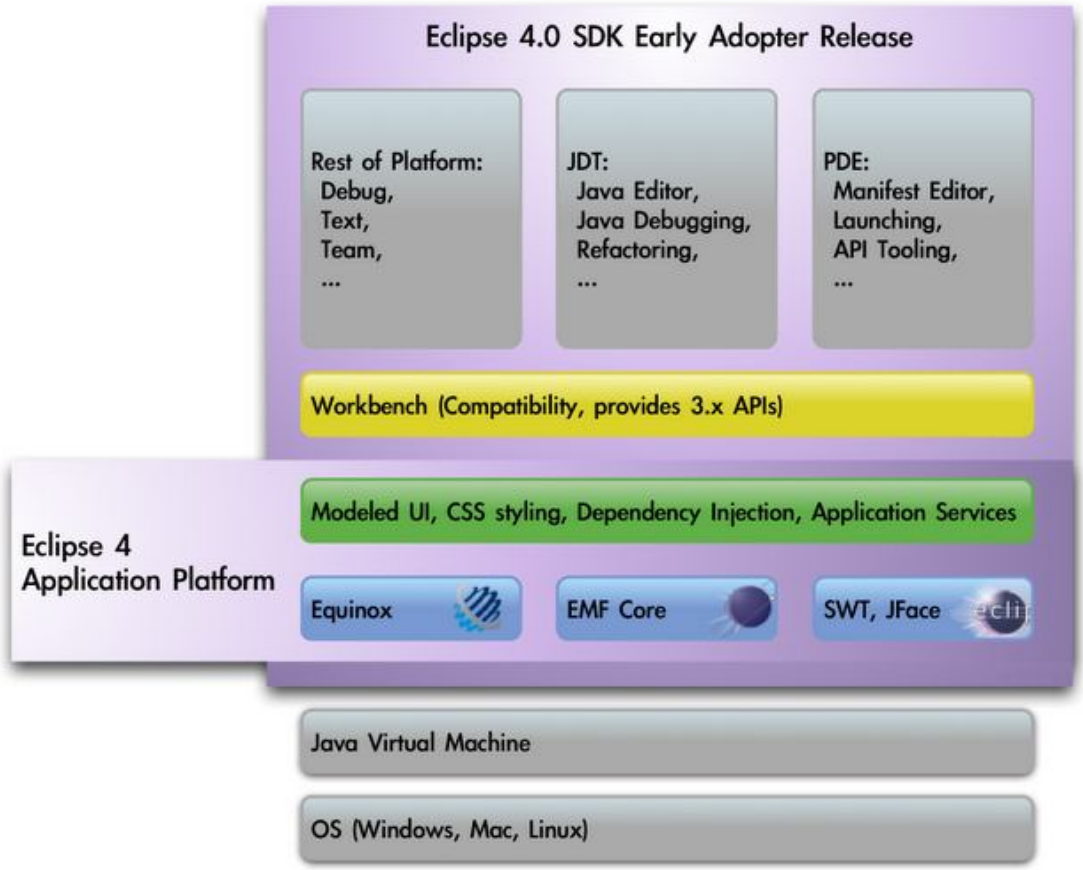
- Soportar la construcción de gran variedad de herramientas de desarrollo.
- Soportar las herramientas proporcionadas por diferentes fabricantes de software independientes (ISV's)
- Soportar herramientas que permitan manipular diferentes contenidos (i.e, HTML, Java, C, JSP, EJB, XML, y GIF).
- Facilitar una integración transparente entre todas las herrramientas y tipos de contenidos sin tener en cuenta al proveedor.
- Proporcionar entornos de dearrollo gráfico (GUI) o no gráficos.
- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows® y Linux™.
- Hacer hincapié en que el lenguaje de programación sea Java para la construcción de nuevos plug-ins.

Mediante APIs interfaces, clases y métodos, se exponen estos mecanismos. La Plataforma también nos posibilita la construcción nuevas herramientas que extenderán la funcionalidad de la Plataforma. La siguiente imagen muestra los componentes APIs, de la Plataforma Eclipse.



ARQUITECTURA ECLIPSE

La siguiente imagen muestra la arquitectura de Eclipse



A continuación describimos algunos de los componentes:

Workspaces

Las diferentes herramientas que son instaladas en la Plataforma Eclipse actúan sobre ficheros regulares que se almacenan en lo que se denominan espacios de trabajo (workspace), que es específico para el usuario. El espacio de trabajo de un usuario contiene varios directorios a nivel más alto (que se denominan proyectos).

Cada proyecto contiene los archivos que son creados y manipulados por el usuario. Todos los archivos en el Jesús Manuel Montero Garrido Plataforma Eclipse. Introducción espacio de trabajo son directamente accesibles por programas standard y herramientas del sistema operativo. . Para minimizar el riesgo de perder archivos, existe un mecanismo de historial mantenido a nivel de cada uno de los proyectos. El usuario puede controlar como el historial se mantiene a través de las preferencias: tamaño, número de días que se mantiene.

Workbench y Toolkits UI

La Plataforma Eclipse tiene interfaz gráfico, construida en base a un workbench (banco de trabajo) que proporciona toda la estructura y presenta un interfaz de usuario (UI) extensible al usuario. Existe también un API de este workbench y su implementación se lleva a cabo a través de lo que se denominan toolkits, que pueden ser de dos tipos:

- SWT.- un conjunto de utilizadas y librerías gráficas integradas con el sistema nativos de ventanas pero con un
- API independiente del sistema operativo.

- JFace.- Un interfaz gráfico implementado sobre SWT que simplifica las tareas de programación

Programación en Equipo
La Plataforma Eclipse permite que un proyecto en el espacio de trabajo de un usuario, pueda ser puesto en un repositorio al cual pueden acceder otros desarrolladores. La Plataforma tiene puntos de extensión y un API de proveedores que permite que nuevos tipos de repositorios puedan ser
instalados.

5.3 ANALISIS FUNCIONAL DEL PARSER

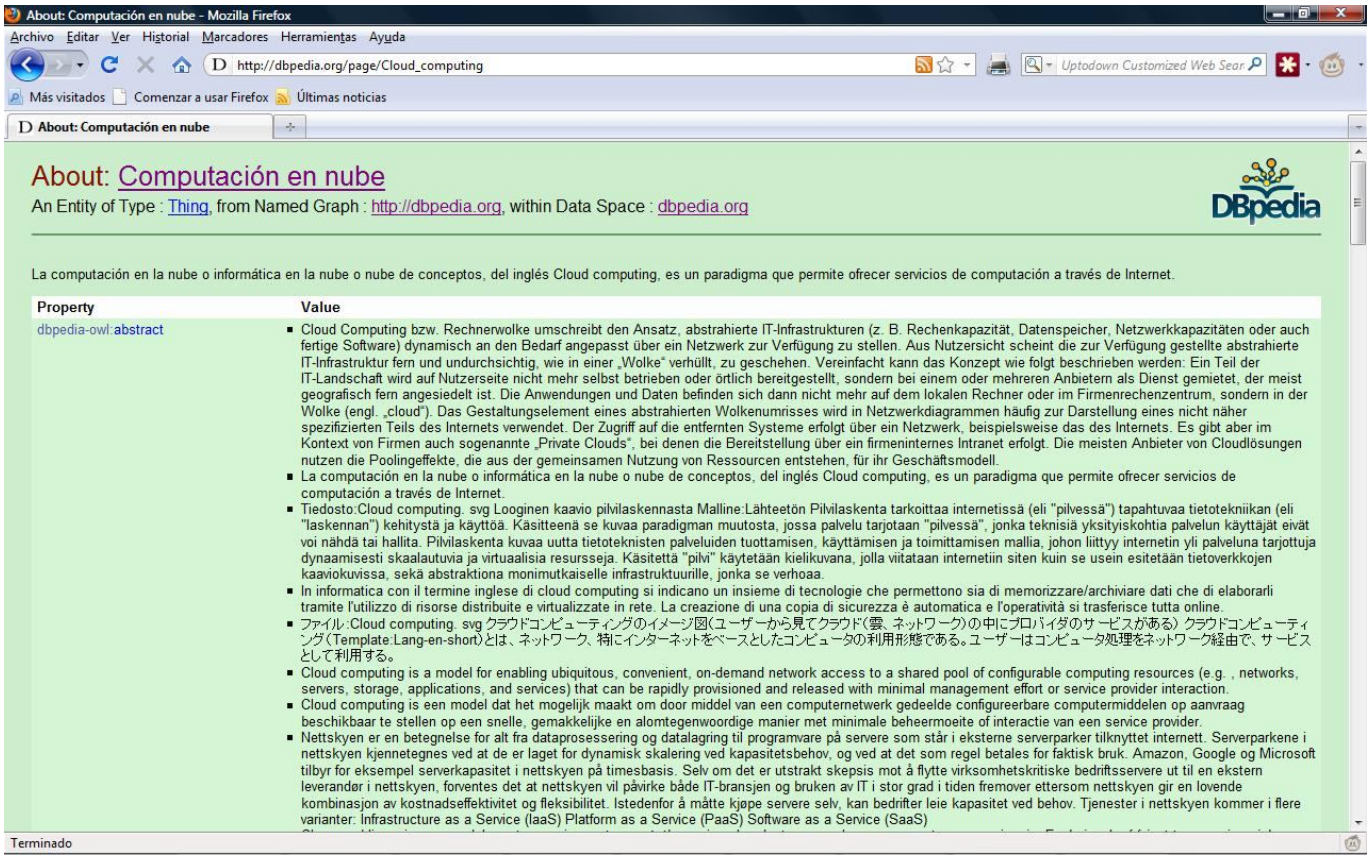
Tal como se ha indicado el objetivo general del Parser es recuperar información de **DBPedia** sobre el dominio del Cloud Computing y rellenar cada una de las clases y atributos que conforman la ontología creada. Por tanto será necesario buscar en DBPedia cada extracto de información que se adecue a cada uno de los atributos de la ontología creada.

Este requerimiento de buscar y encontrar información que encaje con cada uno de los atributos creados es posible que genera una alta complejidad que ha provocado retoques en la definición de la ontología creada previamente con el editor Protege. DBPedia es un proyecto en creación y continuo desarrollo y aunque la información contenida en él está en pleno crecimiento pueden existir carencias de la misma respecto al dominio elegido que obligue a replantearse la definición de esta ontología.

Por ello con el objetivo de encontrar la definición definitiva de la ontología a utilizar en la memoria final hemos investigado los almacenes de información que existen sobre el dominio de cloud computing en DBPedia.

Así tenemos:

En la página http://dbpedia.org/page/Cloud_computing encontramos información sobre el concepto de Cloud Computing almacenado en DBPedia.

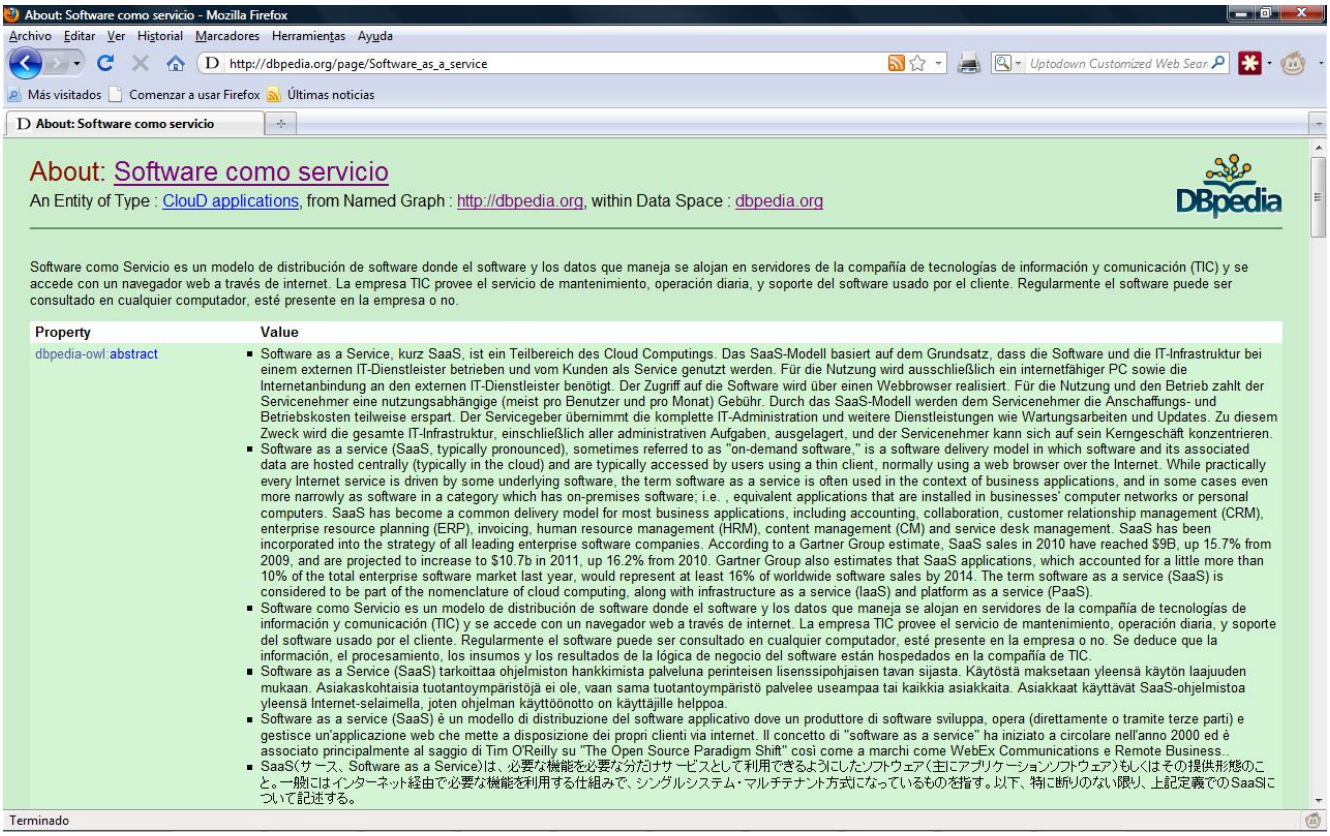


A través de este recurso podremos obtener los siguientes conceptos de información a almacenar en nuestra ontología:

Clase **Cloud_Computing**:

- Denominación : con rdfs:label
- Descripción : con rdfs:coment
- Características: con OWL abstract

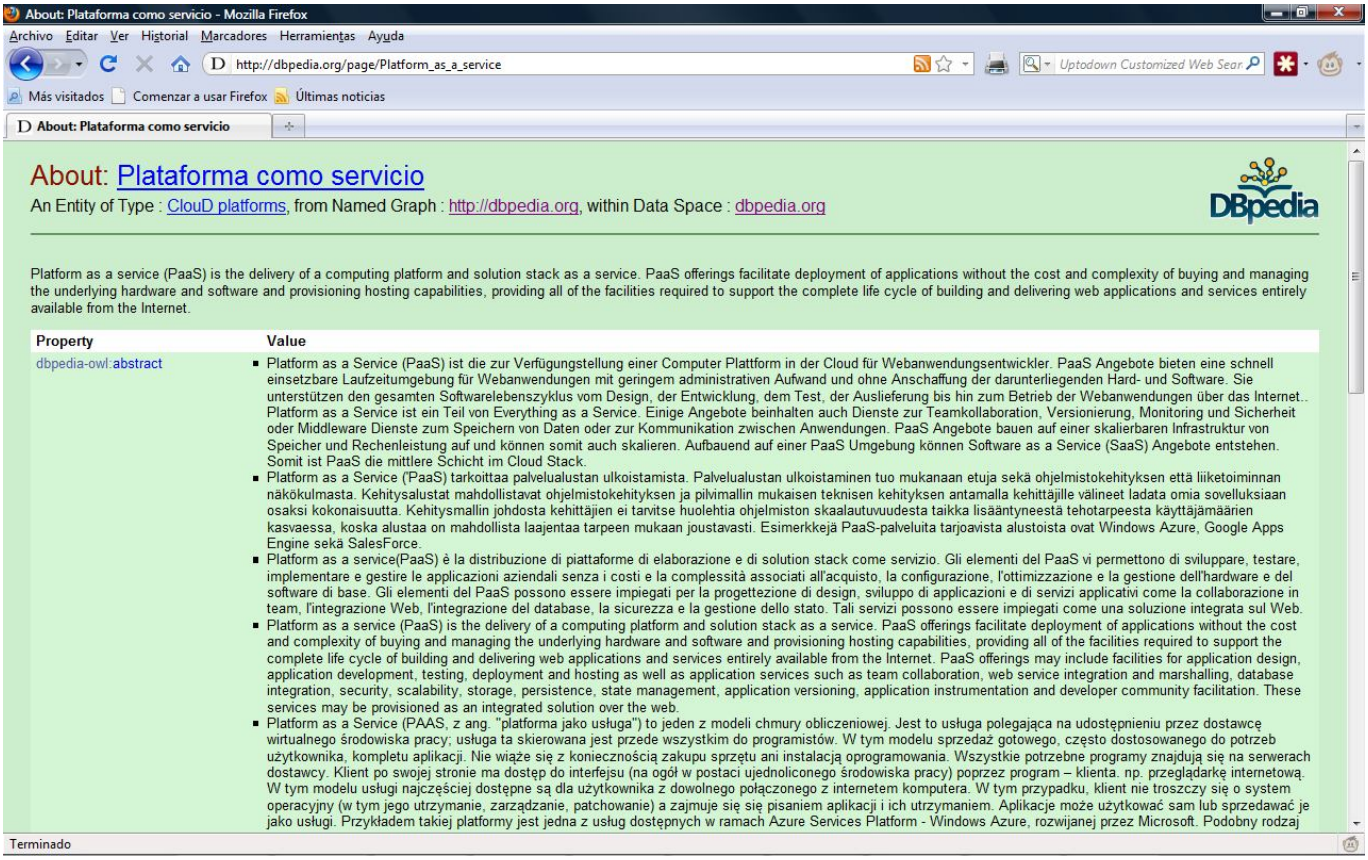
A través de la página http://dbpedia.org/page/Software_as_a_service



Clase PASS:

- Denominación : con rdfs:label
- Descripción : con rdfs:coment
- Características: con OWL abstract

A través de la página http://dbpedia.org/page/Platform_as_a_service (en inglés)



Clase PASS:

- Denominación : con rdfs:label
- Descripción : con rdfs:coment
- Características: con OWL abstract

De la información que no se obtenga por DBPedia y se necesite para rellenar la ontología acudiremos a Wikipedia, mediante un software en código Java descargaremos en un fichero XML la información de los artículos de Wikipedia donde se almacene la información y a partir de ese fichero rellenaremos la ontología.

Los artículos que utilizaremos serán:

http://es.wikipedia.org/wiki/Computaci%C3%B3n_en_nube

http://en.wikipedia.org/wiki/Data_as_a_service

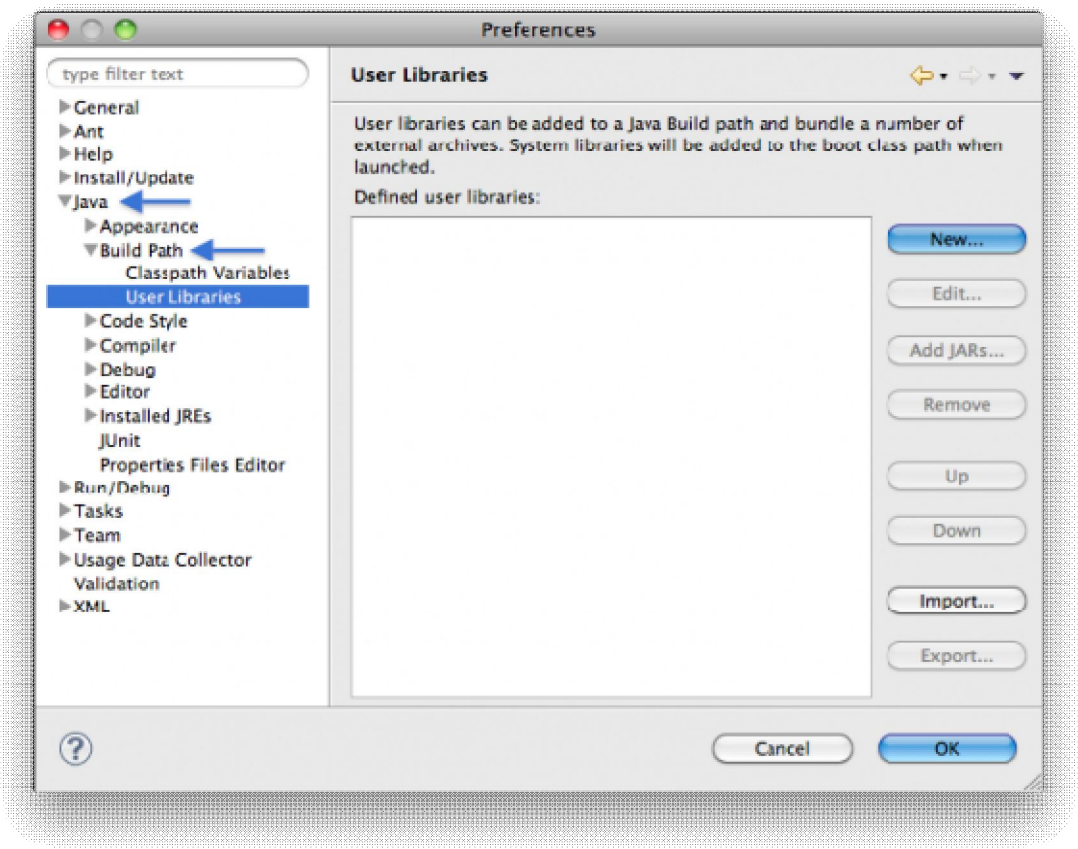
5.4 CONSTRUCCION DEL PARSER PARA POBLAR LA ONTOLOGIA

5.4.1.INTEGRAR ECLIPSE Y JENA

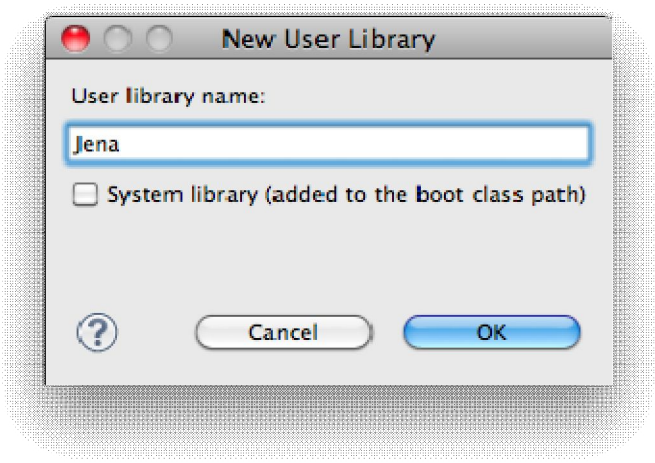
1.Creación de nuestra librería del usuario

Lo primero que tenemos que hacer es crear un nuevo directorio en nuestro sistema en el que almacenaremos todas las librerías del usuario. Vamos a crear un directorio llamado 'UserLibs' dentro de nuestro Workspace. Dentro de 'UserLibs' crearemos un directorio por cada conjunto de .jar que correspondan con una librería del usuario. Por ejemplo para Jena, crearemos un nuevo directorio ('UserLibs/Jena') en el que incluiremos todos los .jar que conforman Jena, sus ficheros fuente .java (en 'UserLibs/Jena/src') y sus javadoc (en 'UserLibs/Jena/javadoc') correspondientes. En este otro post explicaré por qué es interesante incluir los fuentes y javadoc junto con los ficheros .jar ya que creo que es un tema que puede resultar de gran ayuda a más de uno

Posteriormente vamos a crear nuestra primera librería del usuario, en este caso será Jena. Para ello, desde el menú *Preferencias—>Java—>Build Path—>User Libraries y hacemos click en New...*

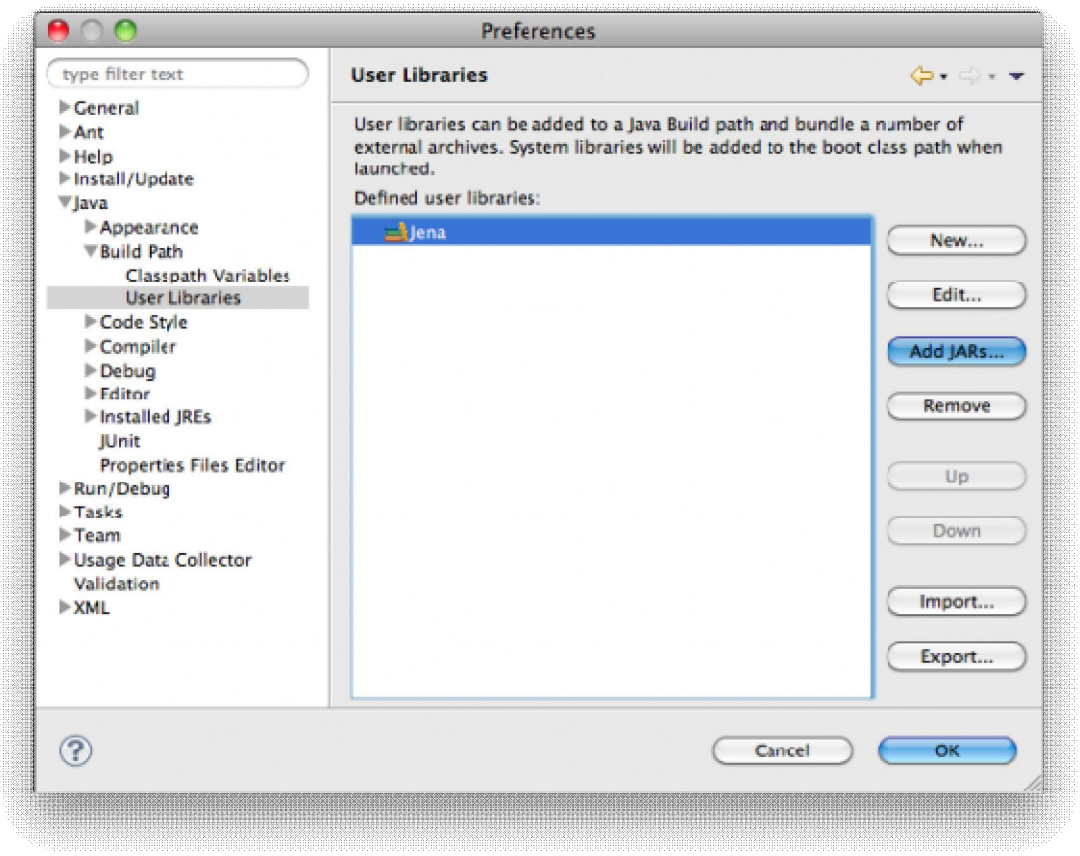


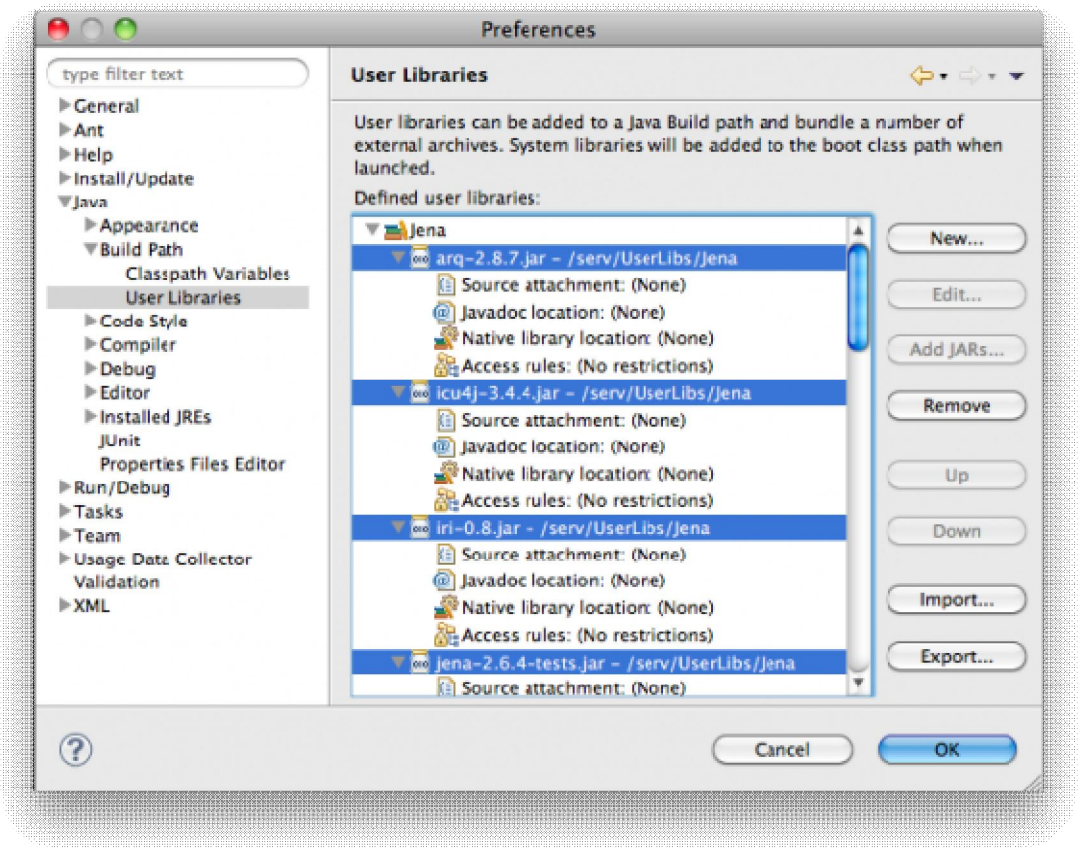
Se nos abrirá una ventanita en la que se nos solicitará un nombre para esa librería que acabamos de crear. En este caso Jena



2. Añadimos los ficheros .jar correspondientes

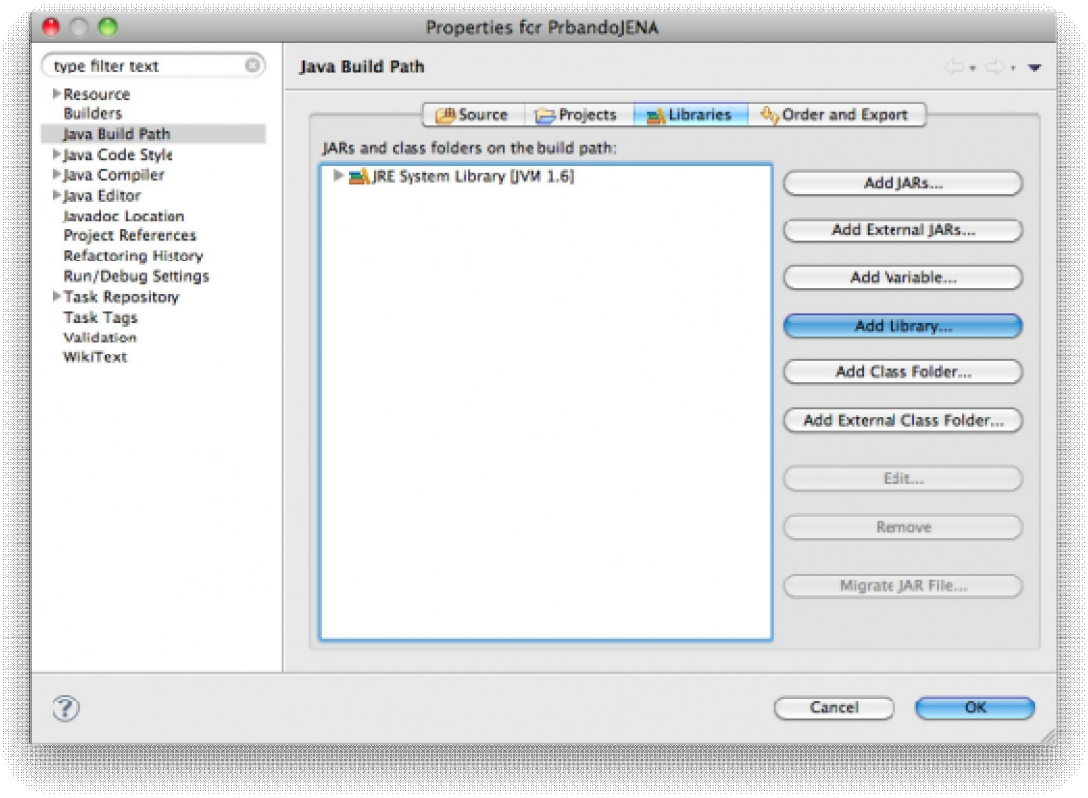
Una vez que presionemos OK y tengamos la librería creada, simplemente tendremos que presionar el botón *Add JARs...*, navegar hasta 'UserLibs/Jena' y añadir todos los .jar que tenemos.



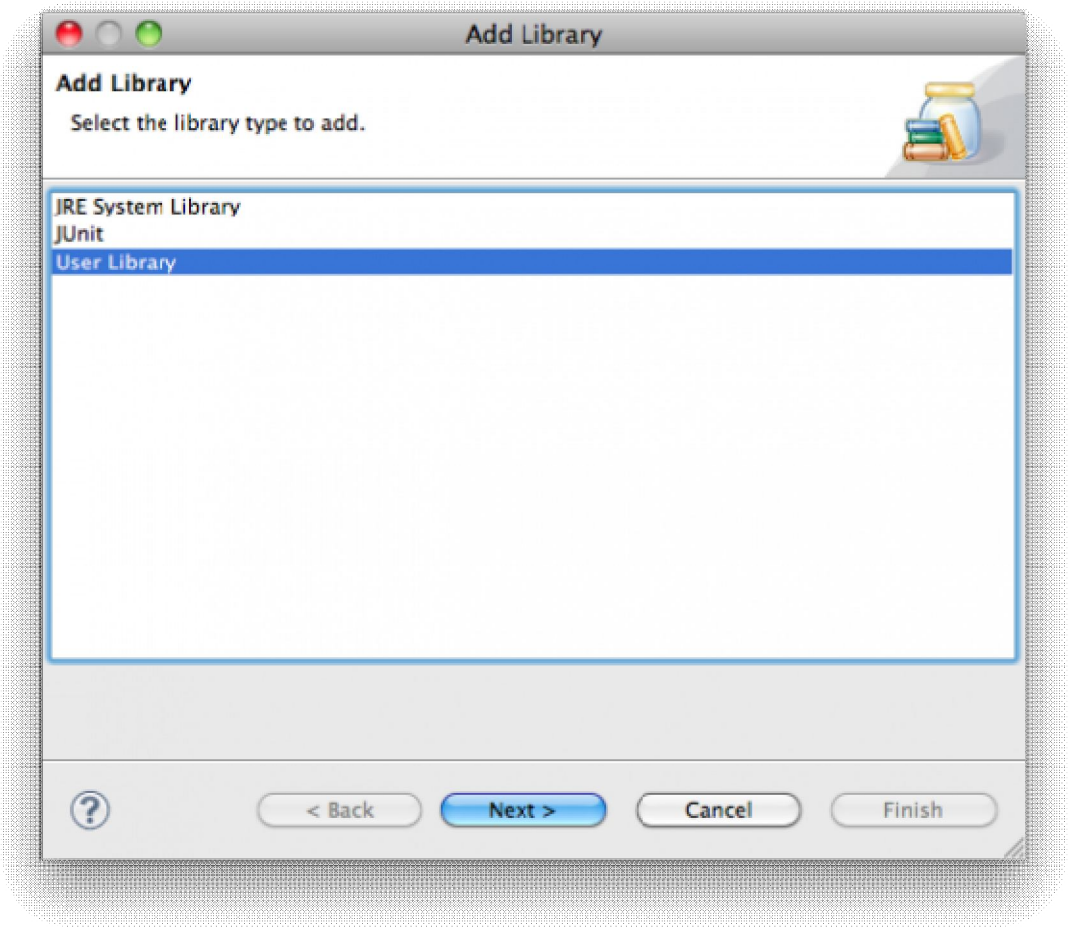


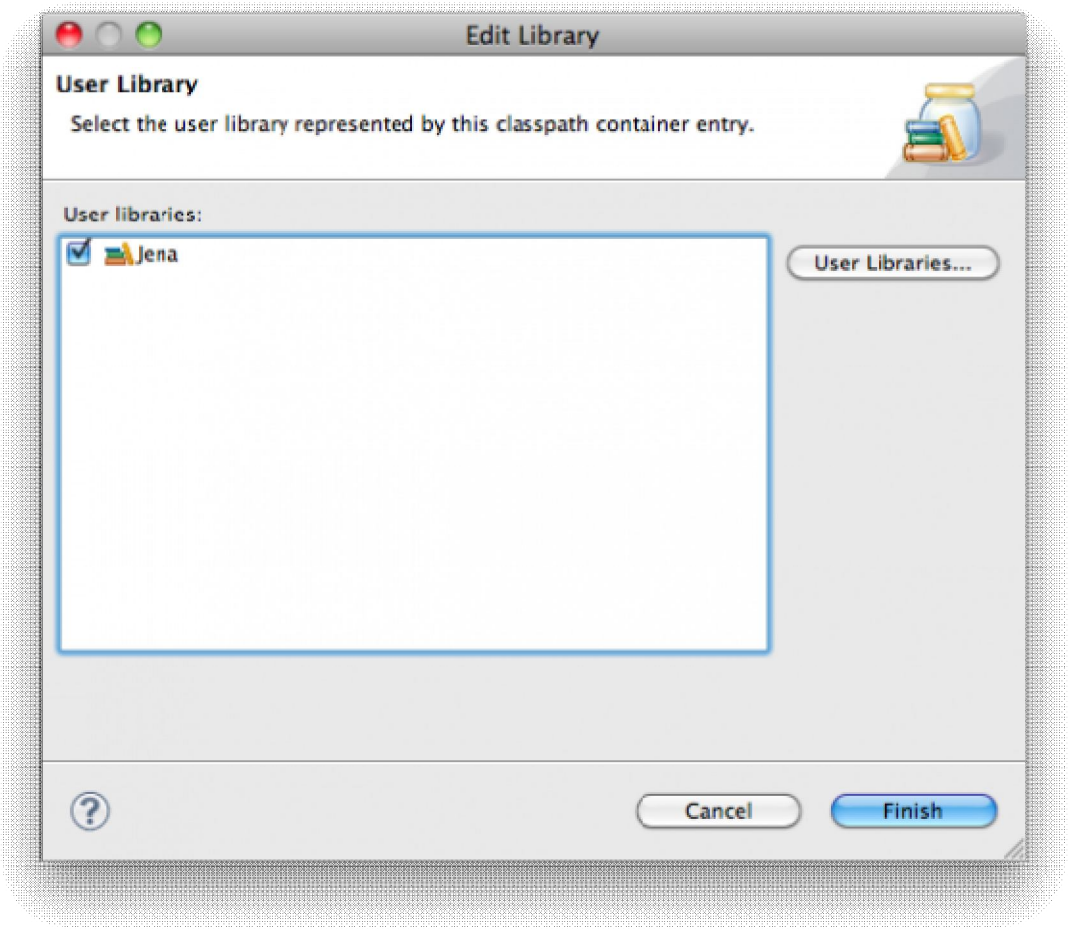
3. Importamos la librería en nuestro proyecto

Se abre las propiedades del proyecto, para ello desde la barra de herramientas superior de eclipse, en el menú *Project—>Properties—>Java Build Path* y hacemos click en "*Add Library...*"



Hay que seleccionar User Library y luego la librería Jena que se acaba de crear y presionamos *Finish*.


Add Library
Add Library
Select the library type to add.
JRE System Library
JUnit
User Library
< Back
Next >
Cancel
Finish



5.5. CONSTRUCCION DEL CODIGO DE PROGRAMACION DEL SOFTWARE PARSER

A continuación vamos a explicar la codificación del software utilizando la librería de Jena

Construcción de la ontología desde el software Java

Con el objetivo de profundizar mejor en el manejo de la ontología vamos a construir el archivo OWL de la ontología Cloud Computing desde el software que programaremos en Java con los recursos de la librería Jena. **De esta manera procederemos a "fabricar" el archivo OWL en tiempo real desde el propio software.** La ontología contendrá las mismas clases y los mismos slots y demás elementos que se ha diseñado previamente con el software Protegé. A través del reto de esta codificación se profundizará más en los elementos de una ontología :Clases, slots , instancias , disjoints etc.. Aparte se considera aporta un elemento diferenciador en cuanto al desarrollo de este trabajo final de carrera.

En cuanto a la estructura del software para la construcción de la ontología en archivo .OWL será la siguiente:

En primer lugar , tal como se ha indicado se utiliza la librería de Jena para el tratamiento de ontología. De esta manera las primeras sentencias que necesitamos en nuestro código son las de importación de librerías de Jena para tratamiento de ontologías que detallamos a continuación:

```
import com.hp.hpl.jena.ontology.*;

import com.hp.hpl.jena.ontology.impl.*;
```

Posteriormente se crea el modelo de Ontología utilizando la clase OntModel de Jena.La sintáxis del código será la siguiente:

```
OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
```

Establecemos el NameSpace por defecto para nuestra ontología con variable String

```
String NS = "cloud";
model.setNsPrefix(NS, "http://www.owl-ontologies.com/OntologyCloud.owl");
```

Instanciamos con la clase OntClass de Jena cada una de los objetos que forman las clases de la Ontología. La nomenclatura a utilizar será el método CreateClass(NameSpace + Nombre Ontología) del objeto model que hemos creado anteriormente.A continuación detallamos cada sentencia que instancia cada una de las clases que conforman la ontología de Cloud Computing que hemos diseñado anteriormente en Protegé:

Creamos la clase Cloud Computing

```
OntClass Cloud_Computing = model.createClass(NS+":"+"Cloud_Computing");
```

Creamos la clase IAAS

```
OntClass IAAS = model.createClass(NS+":"+"IAAS");
```

Creamos la clase DAAS

```
OntClass DAAS = model.createClass(NS+":"+"DAAS");
```

Creamos la clase SAAS

```
OntClass SAAS = model.createClass(NS+":"+"SAAS");
```

Creamos la clase PAAS

```
OntClass PAAS = model.createClass(NS+":"+"PAAS");
```

Añadimos las subclase correspondientes para establecer la estructura que diseñamos , de este modo IAAS,PASS,SAAS son subclases de Cloud_Computing y CAAS y DAAS serían subclases de IAAS.Esta acción se realiza con el método addSubClass de la clase anteriormente creada pasando como parámetro la clase que es superclase.

Añadimos IAAS como subclase de Cloud Computing

```
Cloud_Computing.addSubClass(IAAS);
```

Añadimos PAAS como subclase de Cloud Computing

```
Cloud_Computing.addSubClass(PAAS);
```

Añadimos SAAS como subclase de Cloud Computing

```
Cloud_Computing.addSubClass(SAAS);
```

Añadimos DAAS como subclase de IAAS

DAAS.addSubClass(IAAS);

Añadimos CAAS como subclase de IAAS
CAAS.addSubClass(IAAS);

Ahora creamos las clases Servicios , Proveedor y Tecnología:

OntClass Servicio = model.createClass(NS+":"+"Servicio");
OntClass Proveedor = model.createClass(NS+":"+"Proveedor");
OntClass Tecnologia = model.createClass(NS+":"+"Tecnologia");

Ahora empezaría el proceso de crear las propiedades o slots de cada clase.Para ello utilizaremos la clase DatatypeProperty de la librería Jena. Instanciando un objeto de esta clase de Java se crea cada propiedad. La sintáxis sería la siguiente, objeto DatatypeProperty , nombre de la propiedad y utilizamos el método createDatatypeProperty al que pasamos como parámetros el nameSpace y el nombre de la propiedad. Ejemplo para crear la propiedad Denominación de la clase Cloud Computing con la que va ser un objeto de tipo DatatypeProperty con el método createDatatypeProperty que se añade a nuestro modelo. Los parámetros de createDatatypeProperty tal como hemos dicho van a ser el NS+ nombre de la propiedad , en este caso Denominación.

DatatypeProperty Denominacion = model.createDatatypeProperty(NS+":"+"Denominacion");

Posteriormente añadimos la propiedad a la clase que le corresponde mediante el método addDomain(Nombre de la clase)

Denominacion.addDomain(Cloud_Computing);

Le añadimos con addRange el tipo de valor que va a tener sus instancias y además mediante el método convertToFunctionalProperty(), la fijamos para que solo acepte un valor.

Denominacion.addRange(XSD.xstring); Denominacion.convertToFunctionalProperty();

Añadimos la propiedad Descripcion de Cloud Computing de la misma manera explicada anteriormente:

DatatypeProperty Descripcion = model.createDatatypeProperty(NS+":"+"Descripcion");
Descripcion.addDomain(Cloud_Computing);//Clase a la que pertenece
Descripcion.addRange(XSD.xstring);//Tipo de la propiedad
Descripcion.convertToFunctionalProperty();//Para que solo acepte un valor.

Añadimos la propiedad Caracteristicas de Cloud Computing de la misma manera explicada anteriormente:
DatatypeProperty Caracteristicas = model.createDatatypeProperty(NS+":"+"Caracteristicas");
Caracteristicas.addDomain(Cloud_Computing);//Clase a la que pertenece
Caracteristicas.addRange(XSD.*xstring*);//Tipo de la propiedad
Caracteristicas.convertToFunctionalProperty();//Para que solo acepte un valor.

Añadimos la propiedad Ventajas de Cloud Computing de la misma manera explicada anteriormente:

```
DatatypeProperty Ventajas = model.createDatatypeProperty(NS+":"+"Ventajas");
Ventajas.addDomain(Cloud_Computing);//Clase a la que pertenece
Ventajas.addRange(XSD.xstring);//Tipo de la propiedad
Ventajas.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Añadimos la propiedad Inconvenientes de Cloud Computing de la misma manera explicada anteriormente:

```
DatatypeProperty Inconvenientes =
model.createDatatypeProperty(NS+":"+"Inconvenientes");
Inconvenientes.addDomain(Cloud_Computing);//Clase a la que pertenece
Inconvenientes.addRange(XSD.xstring);//Tipo de la propiedad
Inconvenientes.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Añadimos la propiedad Modelos de Cloud Computing de la misma manera explicada anteriormente:

```
DatatypeProperty Modelos = model.createDatatypeProperty(NS+":"+"Modelos");
Modelos.addDomain(Cloud_Computing);//Clase a la que pertenece
Modelos.addRange(XSD.xstring);//Tipo de la propiedad
Modelos.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Ahora creamos la propiedad denominación de la clase IAAS siguiendo el mismo modelo.

```
DatatypeProperty Denominacion_IAAS =
model.createDatatypeProperty(NS+":"+"Denominacion_IAAS");
Denominacion_IAAS.addDomain(IAAS);//Clase a la que pertenece
Denominacion_IAAS.addRange(XSD.xstring);//Tipo de la propiedad
Denominacion_IAAS.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Ahora creamos la propiedad descripción de la clase IAAS siguiendo el mismo modelo.

```
DatatypeProperty Descripcion_IAAS =
model.createDatatypeProperty(NS+":"+"Descripcion_IAAS");
Descripcion_IAAS.addDomain(IAAS);//Clase a la que pertenece
Descripcion_IAAS.addRange(XSD.xstring);//Tipo de la propiedad
Descripcion_IAAS.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Ahora creamos la propiedad proveedor de la clase IAAS siguiendo el mismo modelo.

```
DatatypeProperty Proveedor_IAAS =
model.createDatatypeProperty(NS+":"+"Proveedor_IAAS");
Proveedor_IAAS.addDomain(IAAS);//Clase a la que pertenece
Proveedor_IAAS.addRange(Proveedor);//Tipo de la propiedad
Proveedor_IAAS.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Ahora creamos la propiedad denominación de la clase DAAS siguiendo el mismo modelo.

```
DatatypeProperty Denominacion_DAAS =
model.createDatatypeProperty(NS+":"+"Denominacion_DAAS");
Denominacion_DAAS.addDomain(DAAS);//Clase a la que pertenece
Denominacion_DAAS.addRange(XSD.xstring);//Tipo de la propiedad
Denominacion_DAAS.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Ahora creamos la propiedad descripción de la clase DAAS siguiendo el mismo modelo.

```
DatatypeProperty Descripcion_DAAS =
model.createDatatypeProperty(NS+":"+"Descripcion_DAAS");
Descripcion_DAAS.addDomain(DAAS);//Clase a la que pertenece
Descripcion_DAAS.addRange(XSD.xstring);//Tipo de la propiedad
Descripcion_DAAS.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Ahora creamos la propiedad Característica de la clase DAAS siguiendo el mismo modelo.

```
DatatypeProperty Caracteristica_DAAS =
model.createDatatypeProperty(NS+":"+"Descripcion_DAAS");
Descripcion_DAAS.addDomain(DAAS);//Clase a la que pertenece
Descripcion_DAAS.addRange(XSD.xstring);//Tipo de la propiedad
Descripcion_DAAS.convertToFunctionalProperty();//Para que solo acepte un valor.
```

Esta misma metodología vamos a seguir para la creación de las siguientes propiedades de sus correspondientes clases:

Clase PAAS y Slots
- Denominación
- Descripción
- Proveedor

Clase SAAS y Slots
- Denominación
- Descripción
- Proveedor

Clase CAAS y Slots
- Denominación
- Descripción
- Caracteristicas

Clase Tecnología y Slots
- Denominación
- Descripción
- Uso

Clase Proveedor y Slots
- Denominación
- Descripción
- Pais
- Fundación
- Servicios ofertados

Clase Proveedor y Slots
- Denominación
- Descripción
- Servicios

5.6. CONSTRUCCION SOFTWARE PARA POBLAR ONTOLOGIA

Para poblar la ontología , tal como hemos indicado anteriormente utilizaremos información de DBPEDIA y la que no encontremos en este repositorio la recogeremos de **Wikipedia.** A continuación se explica la codificación de cada uno de estas funcionalidades.

5.6.1.EXTRACCION DE INFORMACIÓN DESDE DBPEDIA.

Esta clase contendrá las consultas SPARQL construidas mediante la estructura de tripleta, ***sujet- predicado-objeto*** estudiadas anteriormente.

En primer lugar y de cara a profundizar más en el conocimiento de SPARQL y de sus elementos, utilizamos el elemento ASK(elemento de pregunta) para interrogar sobre el estado del servicio de DBPedia. Esta propiedad la utilizamos en el método TestDBPedia de la clase ConsultaJena en la que comprobamos la disponibilidad de conexión con DBPedia y que no sufre ninguna contingencia.

```
public  static boolean TestDBpedia() {
String service = "http://dbpedia.org/sparql";
boolean prueba =false;
String query = "ASK { }";
QueryExecution qe = QueryExecutionFactory.sparqlService(service, query);
                try {
                   if (qe.execAsk()) {
                      prueba= true;
                   } // end if
                } catch (QueryExceptionHTTP e) {
                   prueba= false;
                } finally {
                   qe.close();
                } // end try/catch/finally
                 return prueba;
              } // end method
```

Este método devuelve un booleano comunicando si el servicio está activo o no.

Posteriormente en el método DataDbPedia de la clase ConsultaJena y se realiza las consultas SPARQL para recoger la información DBPedia que rellenará las siguientes propiedades de la ontología de Cloud_Computing.

SPARQL para rellenar el SLOT Descripción Cloud_Computing:

```
"PREFIX dbont: <http://dbpedia.org/ontology/> "+
                        "PREFIX dbp: <http://dbpedia.org/property/>"+
                        "SELECT ?abstract WHERE {{
<http://dbpedia.org/resource/Cloud_computing> <http://dbpedia.org/ontology/abstract>
?abstract ." +
                        "FILTER langMatches( lang(?abstract), 'es') }" +
                                "}";
```

SPARQL para rellenar el SLOT Descripción IAAS:

```
"PREFIX dbont: <http://dbpedia.org/ontology/> "+
```

```
                    "PREFIX dbp: <http://dbpedia.org/property/>"+
                    "SELECT ?abstract WHERE {{
<http://dbpedia.org/page/Cloud_computing%23Infrastructure>
<http://dbpedia.org/ontology/abstract> ?abstract ." +
                    "FILTER langMatches( lang(?abstract), 'en') }" +
                                "}";
```

SPARQL para rellenar el SLOT Descripción PAAS:

```
"PREFIX dbont: <http://dbpedia.org/ontology/> "+
                    "PREFIX dbp: <http://dbpedia.org/property/>"+
                    "SELECT ?abstract WHERE {{
<http://dbpedia.org/resource/Platform_as_a_service>
<http://dbpedia.org/ontology/abstract> ?abstract ." +
                                "FILTER langMatches( lang(?abstract), 'en') }}"
```

SPARQL para rellenar el SLOT Descripción SAAS:

```
"PREFIX dbont: <http://dbpedia.org/ontology/> "+
"PREFIX dbp: <http://dbpedia.org/property/>"+
"SELECT ?abstract WHERE {{ <http://dbpedia.org/resource/Software_as_a_service>
<http://dbpedia.org/ontology/abstract> ?abstract ." +
                    "FILTER langMatches( lang(?abstract), 'es') }" +
                                "}";
```

SPARQL para rellenar el Denominación de Cloud Computing:

```
"PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +
“PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
"PREFIX dbp: <http://dbpedia.org/property/>"+
 "PREFIX dbont: <http://dbpedia.org/ontology/>"+
 "PREFIX dbp: <http://dbpedia.org/property/>"+
  "SELECT ?label "+
  "WHERE { <http://dbpedia.org/resource/Cloud_computing> rdfs:label ?label ."+
   "FILTER langMatches( lang(?label), 'es') }";
```

SPARQL para rellenar el Denominación de PAAS

```
"PREFIX dbont: <http://dbpedia.org/ontology/>"+
"PREFIX dbp: <http://dbpedia.org/property/>"+
 "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +
"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
  "SELECT ?label  "+ " WHERE { <http://dbpedia.org/resource/Platform_as_a_service>
rdfs:label ?label ."+" FILTER langMatches( lang(?label), 'es') }";
```

SPARQL para rellenar el Denominación de SAAS

```
"PREFIX dbont: <http://dbpedia.org/ontology/>"+
"PREFIX dbp: <http://dbpedia.org/property/>"+
 "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +
 "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
  "SELECT ?label  "+" WHERE { <http://dbpedia.org/resource/Software_as_a_service>
rdfs:label ?label ."+ " FILTER langMatches( lang(?label), 'es') }";
```

SPARQL para rellenar el Denominación de IAAS

```
"PREFIX dbont: <http://dbpedia.org/ontology/>"+
 "PREFIX dbp: <http://dbpedia.org/property/>"+
  "PREFIX dbont: <http://dbpedia.org/ontology/>" +
 "PREFIX dbp: <http://dbpedia.org/Category/>"+
  "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +
  "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
  "SELECT ?label WHERE {{
http://dbpedia.org/resource/Category:Infrastructure_as_a_Service> rdfs:label ?label
"+FILTER langMatches( lang(?label), 'en')}}";
```

SPARQL para rellenar el Descripción Cloud

```
"PREFIX dbont: <http://dbpedia.org/ontology/>" +
"PREFIX dbp: <http://dbpedia.org/property/>" +
 "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +
"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
 " SELECT ?comment WHERE { <http://dbpedia.org/resource/Cloud_computing> rdfs:comment
?comment ." +    " FILTER langMatches( lang(?comment), 'es') }";
```

SPARQL para rellenar el Descripción SAAS

```
PREFIX dbont: <http://dbpedia.org/ontology/>" +
"PREFIX dbp: <http://dbpedia.org/property/>" +
 "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +
 "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
 " SELECT ?comment WHERE { <http://dbpedia.org/resource/Software_as_a_service>
rdfs:comment ?comment ." + " FILTER langMatches( lang(?comment), 'es') }";
```

SPARQL para rellenar el Descripción PAAS

```
"PREFIX dbont: <http://dbpedia.org/ontology/>" +
```

```
"PREFIX dbp: <http://dbpedia.org/property/>" +

"PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +

"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +

" SELECT ?comment WHERE { <http://dbpedia.org/resource/Platform_as_a_service>
rdfs:comment ?comment ." +   " FILTER langMatches( lang(?comment), 'en') }";
```

SPARQL para rellenar la propiedad Localización de Proveedor

```
"PREFIX dbont: <http://dbpedia.org/ontology/>" +

"PREFIX dbp: <http://dbpedia.org/property/>" +

"PREFIX dbo: <http://dbpedia.org/ontology/>" +

"PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +

"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +

"PREFIX dc: <http://purl.org/dc/elements/1.1/>" +

"PREFIX foaf: <http://xmlns.com/foaf/0.1/>" +

"PREFIX dbpprop: <http://dbpedia.org/property/>" +

"SELECT  ?locationCity  WHERE {" +

"<http://dbpedia.org/resource/WorldAPP> <http://dbpedia.org/ontology/locationCity>
?locationCity ."+ "}";
```

SPARQL para rellenar la propiedad Nombre de Proveedor

```
"PREFIX dbont: <http://dbpedia.org/ontology/>" +

"PREFIX dbp: <http://dbpedia.org/property/>" +

"PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +

"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +

"PREFIX foaf: <http://xmlns.com/foaf/0.1/>" +

"SELECT ?name WHERE { <http://dbpedia.org/resource/WorldAPP> foaf:name ?name ." +" }";
```

SPARQL para rellenar la propiedad Año Fundación de Proveedor

```
"PREFIX dbont: <http://dbpedia.org/ontology/>" +

"PREFIX dbp: <http://dbpedia.org/property/>" +
```

```
"PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +

"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +

 "PREFIX dbpprop: <http://dbpedia.org/property/>" +

"SELECT ?foundation WHERE { <http://dbpedia.org/resource/WorldAPP> dbpprop:foundation
?foundation ." +  "}";
```

SPARQL para rellenar la propiedad Descripción de Proveedor

```
"PREFIX dbont: <http://dbpedia.org/ontology/>" +

"PREFIX dbp: <http://dbpedia.org/property/>" +

"SELECT ?abstract WHERE {{ <http://dbpedia.org/resource/WorldAPP>
<http://dbpedia.org/ontology/abstract> ?abstract ." + "}}";
```

SPARQL para rellenar la propiedad Descripción de Servicio

```
"PREFIX dbont: <http://dbpedia.org/ontology/> "+

 "PREFIX dbp: <http://dbpedia.org/property/>"+

  "SELECT ?abstract WHERE {{ <http://dbpedia.org/resource/Virtualization>
<http://dbpedia.org/ontology/abstract> ?abstract ." +

  "FILTER langMatches( lang(?abstract), 'es') }" +"}";
```

SPARQL para rellenar la propiedad Denominación de Servicio

```
"PREFIX dbont: <http://dbpedia.org/ontology/>"+

 "PREFIX dbp: <http://dbpedia.org/property/>"+

  "PREFIX dbont: <http://dbpedia.org/ontology/>" +

 "PREFIX dbp: <http://dbpedia.org/Category/>"+

  "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +

   "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +

   "SELECT ?label WHERE {{ <http://dbpedia.org/resource/Virtualization> rdfs:label ?label  "+

  " FILTER langMatches( lang(?label), 'es')}}";
```

5.6.2.EXTRACCION DE INFORMACIÓN DESDE WIKIPEDIA.

Este proceso tiene dos pasos descargar la información de los artículos de Wikipedia requeridos en un fichero XML y posteriormente extraer la información de ese fichero para poblar las instancias de la ontología de Cloud Computing.

Descargar información de Wikipedia

Lo haremos mediante la clase DownloadDataWikipedia que explicamos a continuación:

Inicializamos los objetos y variables que necesitamos ,una URL un InputStream y un DataInputStream y una variable de tipo String

```
URL u;
InputStream is = null;
DataInputStream dis;
String s;
```

Indicamos la URL del artículo de Wikipedia que queremos descargar.

```
u = new
URL("http://es.wikipedia.org/wiki/Especial:Exportar/Computaci%C3%B3n_en_nube");
```

Abrimos el InputStream y lo convertimos en un buffered DataInputStream que hara la lectura;

```
is = u.openStream();        // throws an IOException
dis = new DataInputStream(new BufferedInputStream(is));
```

Leemos cada registro del inputStream y lo guardamos en el fichero xml (indicamos la ruta del fichero donde queremos que se guarde) que posteriormente utilizaremos para extraer los datos que necesitemos

```
FileWriter outFile = new FileWriter("C:/Users/prevalia/Cloud.xml");
 PrintWriter out = new PrintWriter(outFile);

      while ((s = dis.readLine()) != null) {
          out.println(s);
      }
      out.close();

    } catch (MalformedURLException mue) {

      System.out.println(" MalformedURLException.");
      mue.printStackTrace();
      System.exit(1);

    } catch (IOException ioe) {

      System.out.println("IOException.");
      ioe.printStackTrace();
      System.exit(1);
```

} finally {

Para que la información descargada de Wikipedia se muestre correctamente en este archivo XML es necesario establecer la codificación adecuada que hacemos mediante la siguiente sentencia:

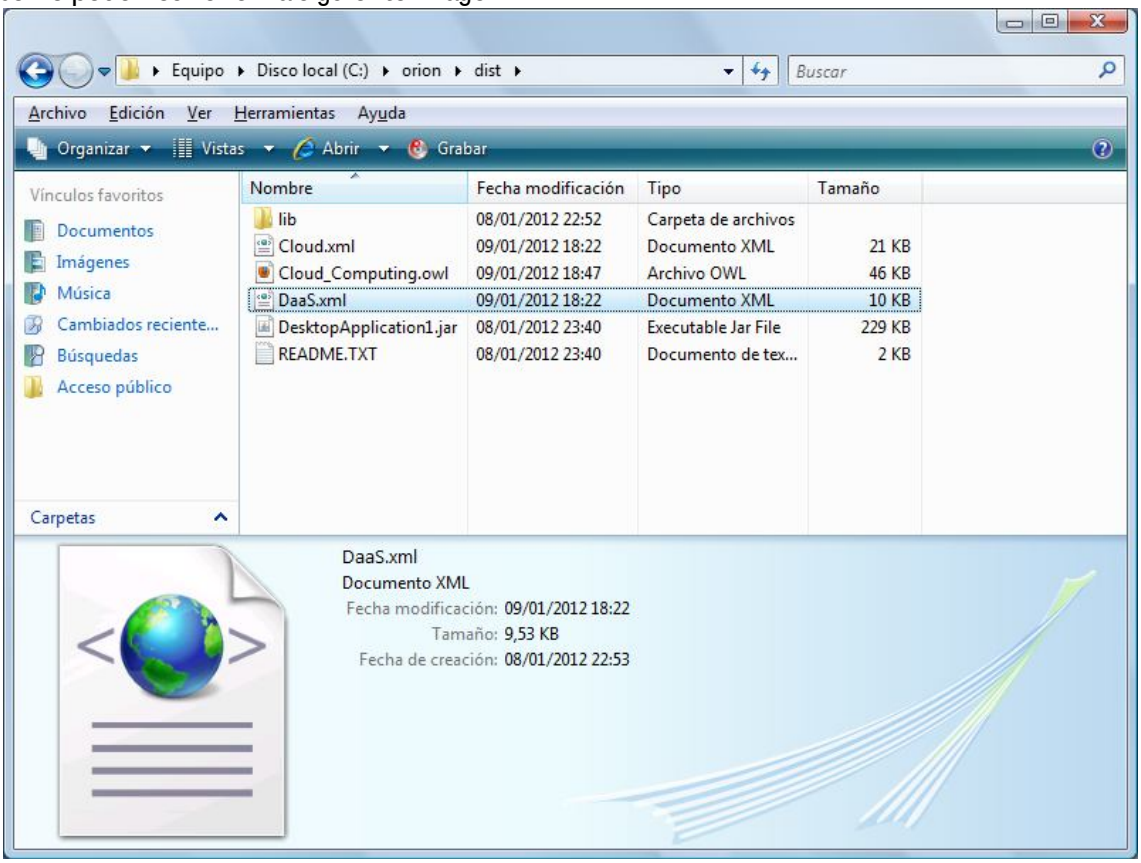
Out.println("<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>");

Con esta programación almacenamos la información del artículo de wikipedia http://es.wikipedia.org/wiki/Computacion_en_nube en un archivo Cloud.xml en el path de la aplicación: con este archivo rellenaremos las propiedades Características, Ventajas Inconvenientes y Tipos de la clase Cloud Computing.

Este mismo proceso realizamos para descargar la información de la clase CAAS , que descargamos del siguiente artículo de Wikipedia:

http://en.wikipedia.org/wiki/Data_as_a_service

Con la información de este artículo se crea el archivo DAAS.xml en el path de la aplicación como podemos ver en la siguiente imagen:



5.6.3 EXTRAER DATOS DEL FICHERO XML CREADO CON LA INFORMACION DEL ARTICULO DE WIKIPEDIA

Creamos las variables donde almacenaremos la información de cada uno de las propiedades o slots de la instancia que de la ontología Cloud Computing que vamos a rellenar.

```
public static String Beneficios=null;
public static String Desventajas=null;
```

Abrimos y creamos un parse del documento XML que vamos a leer y que hemos creado en la clase DownloadDataWikipedia.

```
DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
File f = new File("C:/Users/prevalia/Cloud.xml");
Document documento = builder.parse(f);
```

Recorremos los nodos y cuando encontramos el de nombre text que contienen la información del articulo separamos mediante substring cada apartado de la información que pertenece a cada slot o propiedad. Este substring busca el índice de las palabras que corresponden a los apartados:
Ventajas, Inconvenientes, Introducción, Tipos, Características de la clase Cloud_Computing. También se busca la información correspondiente al aparatado Descripción de la clase IAA. El índice lo obtenemos con el la propiedad *indexOf* de la cadena que corresponde al nombre de cada propiedad.

```
if(nodo.getNodeName()=="text") {
        String valor=nodo.getNodeValue();
        String TextoCompleto= nodo.getTextContent().toString();
        int inicio= TextoCompleto.indexOf("== Beneficios ==");
        int finalizacion = TextoCompleto.indexOf("== Desventajas ==");
        Beneficios = ReplaceCode(TextoCompleto.substring(inicio, finalizacion));
         inicio= TextoCompleto.indexOf("== Desventajas ==");
        finalizacion = TextoCompleto.indexOf("== Capas ==");
        Desventajas = ReplaceCode(TextoCompleto.substring(inicio, finalizacion));
        inicio= TextoCompleto.indexOf("== Tipos de nubes ==");
        finalizacion = TextoCompleto.indexOf("== Comparaciones ==");
        Tipos_Nubes = ReplaceCode(TextoCompleto.substring(inicio, finalizacion));
        inicio= TextoCompleto.indexOf("== IntroducciÃ³n ==");
        finalizacion = TextoCompleto.indexOf("== Comienzos ==");
        Caracteristicas = ReplaceCode(TextoCompleto.substring(inicio, finalizacion));
        inicio= TextoCompleto.indexOf("== Infraestructura como servicio ==");
        finalizacion = TextoCompleto.indexOf("== Tipos de nubes ==");
        IAAS = ReplaceCode(TextoCompleto.substring(inicio, finalizacion));
        }
      NodeList hijos = nodo.getChildNodes();
      for(int i = 0; i < hijos.getLength(); i++){
         Node nodoNieto = hijos.item(i);
         Resultados(fichero,nodoNieto,Campo);
      }

      }
```

Tal como se ha indicado en el apartado anterior en esta clase también se busca la información de la clase CAAS , en concreto los apartados Descripción DAAS y Características DAAS.

```
//CAMPOS DE WIKIPEDIA PARA CAAS
File fDaaS = new File(path + "\\DaaS.xml");
Document documentoDaas = builder.parse(fDaaS);

        if(nodo != null){


        if(nodo.getNodeName()=="text") {
        String valor=nodo.getNodeValue();
        String TextoCompleto= nodo.getTextContent().toString();
        int inicio= TextoCompleto.indexOf("'''Data as a service'''");
        int finalizacion = TextoCompleto.indexOf("==Benefits==");
        DescripcionDaaS = ReplaceCode(TextoCompleto.substring(inicio, finalizacion));
        inicio= TextoCompleto.indexOf("==Benefits==");
        finalizacion = TextoCompleto.indexOf("==Pricing models==");
        CaracteristicasDaaS = ReplaceCode(TextoCompleto.substring(inicio, finalizacion));
                }
      NodeList hijos = nodo.getChildNodes();
      for(int i = 0; i < hijos.getLength(); i++){
         Node nodoNieto = hijos.item(i);
         Resultados(fichero,nodoNieto,Campo);
      }

      }
        if("descripcionDaas".equals(Campo)){
        Campo= DescripcionDaaS;
        }if("caracteristicaDaaS".equals(Campo)){
        Campo= CaracteristicasDaaS;
        }

        }
```

5.6.4. CREAR LA INSTANCIA DE LA CLASE Y RELLENAR LA ONTOLOGIA CON LOS DATOS DE DBPEDIA Y WIKIPEDIA EXTRAIDOS.

En la clase Parser.java se crean las instancias de la clase y mediante llamadas a las clases que hemos visto anteriormente que recogen la información de Wikipedia y DBPedia se puebla la ontología con esos datos.

El proceso es crear una instancia de tipo Individual (objeto de la clase Individual de Jena) y crear la instancia mediante el método createIndividual con los parámetros Nombre y clase de la ontología a la que instanciar.

```
Individual cloud =
model.createIndividual(NS+":"+"Computacion_en_nube",Cloud_Computing);
```

Posteriormente se puebla el slot de esa instancia de dos formas según sea dato de DBPedia o de Wikipedia.

Si es DBPedia mediante el objeto setPropertyValue de la instancia creada se incluye el dato a través del método createTypedLiteral que pide como parámetro el valor de la información que se obtienen a través de una llamada a la clase ConsultaJena método *DataDbpedia y* como parámetro el nombre de la instancia que queremos poblar.Ejempplo instancia Descripción de Cloud Computing:

cloud.setPropertyValue(Descripcion,
model.createTypedLiteral(ConsultaJena.DataDbpedia("Cloud_Computing")));

Si es Wikipedia mediante el objeto setPropertyValue de la instancia creada se incluye el dato a través del método createTypedLiteral que pide como parámetro el valor de la información que se obtienen a través de una llamada a la clase ExtractDataWikipedia método *readxml y* como parámetro el nombre del fichero xml que contienen la información y el de la instancia que queremos poblar. Ejemplo poblar slot Ventajas de Cloud Computing.

cloud.setPropertyValue(Ventajas,
model.createTypedLiteral(ExtractDataWikipedia.readxml("C:/Users/prevalia/Cloud.xml",
"ventajas")));

Con esta metodología se rellena la información de los siguientes slots y clases:

- Slot Denominación de la clase Cloud Computing

cloud.setPropertyValue(Denominacion,
model.createTypedLiteral(ConsultaJena.DataDbpedia("Denominacion_Cloud")));

- Slot Descripción de la clase Cloud Computing

cloud.setPropertyValue(Descripcion,
model.createTypedLiteral(ConsultaJena.DataDbpedia("Descripcion_Cloud")));

- Slot Descripción de la clase Características

cloud.setPropertyValue(Caracteristicas,
model.createTypedLiteral(ExtractDataWikipedia.readxml(path + "\\Cloud.xml",
"caracteristica")));

- Slot Ventajas de la clase Características

cloud.setPropertyValue(Ventajas,
model.createTypedLiteral(ExtractDataWikipedia.readxml(path + "\\Cloud.xml",
"beneficio")));

- Slot Ventajas de la clase Inconvenientes

cloud.setPropertyValue(Inconvenientes,
model.createTypedLiteral(ExtractDataWikipedia.readxml(path + "\\Cloud.xml",
"desventaja")));

- Slot Modelos de la clase Inconvenientes

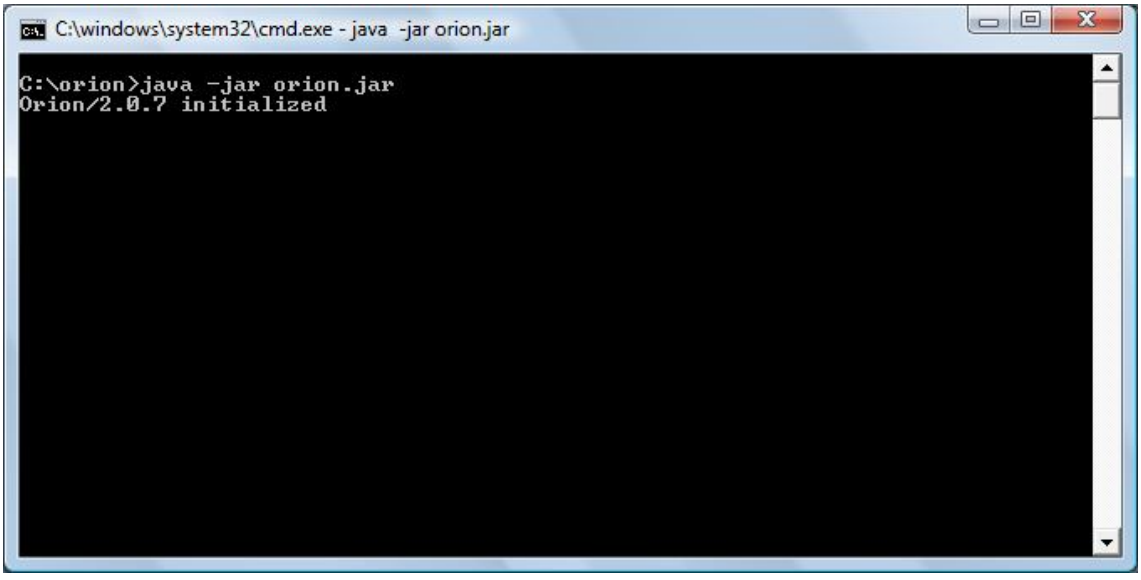
cloud.setPropertyValue(Modelos,
model.createTypedLiteral(ExtractDataWikipedia.readxml(path + "\\Cloud.xml", "tipos")));

5.6.5 NAVEGADOR

Para el visualizador a través del navegador que se ha desarrollado hemos utilizado tecnología JSP Java Server Pages. Para la ejecución de archivos de esta tecnología se necesita un servidor de aplicaciones java. En este caso nos hemos decidido por orion server http://www.orionserver.com/ por ser gratuito , muy sencillo de utilizar y extremadamente ligero.

El proceso de su instalación y configuración es muy sencillo tan solo hace falta **descargárselo** de la web , y seguir los siguientes paso:

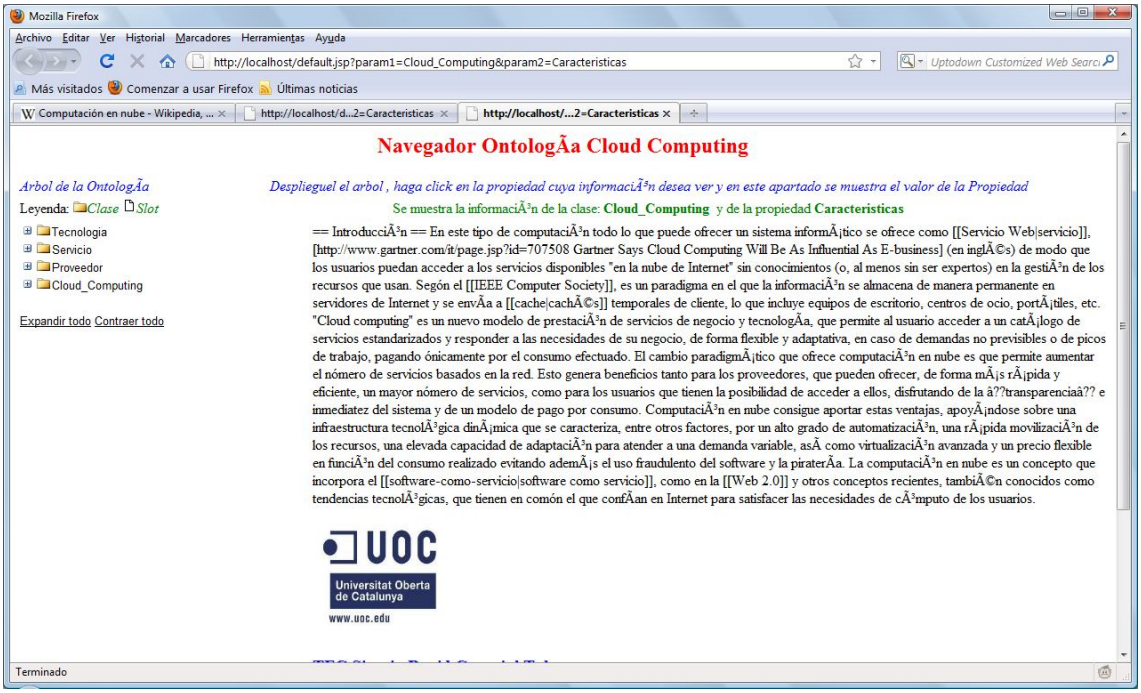
- **Descomprimir el fichero** , en esta caso se ha elegido la raíz del disco principal para su descompresión(se ha descomprimido en C)
- Instalar el **JSDK** de JAVA , en nuestro caso se ha instalado **jdk1.6.0_23** y la máquina virtual de **java JRE 6**
- Asegurarse de tener la variable de entorno **JAVA_HOME** apuntando al directorio de instalación de Java
- Abrimos una ventana de consola MS-DOS (en Windows con cmd desde inicio) o de consola en Linux-Solaris y ejecutamos la instrucción siguiente: **Ejecutar** el comando (añadir password) **Java –jar orion.jar –install**
- Para **arrancar orion** en la misma ventana **MS-DOS o de consola ejecutamos Arrancar** el servidor con Java –jar orion.jar



Nos aparecerá un mensaje de que Orion se ha inicializado.

5.6.6.CODIFICACION DEL VISUALIZADOR DE LA ONTOLOGIA A TRAVES DEL NAVEGADOR

La siguiente funcionalidad de la aplicación es la creación en tiempo real de un navegador que permita desplazarse por las clases y propiedades de la ontología Cloud Computing y ver la información almacenada en ellos a través de la población de una instancia de dicha ontología que hemos explicado anteriormente. La imagen del navegador es la siguiente:



Este navegador consta de un menú árbol que muestra las clases y propiedades de la ontología. La codificación utilizada para su creación basada en la librería Jena ha sido la siguiente:

Se crea una archivo DataInputStream para leer el ,owl que contiene la ontología.

```
dis = new DataInputStream(new BufferedInputStream(is));
```

Creamos el archivo .jsp en el path de orion (servidor de aplicaciones JSP) en la carpeta defaultApp para que lo tome como la aplicación a deployar por defecto.

```
String path= System.getProperty("user.dir") ;
String pathOrion= "C:/orion/default-web-app/";
FileWriter outFile;
outFile = new FileWriter(pathOrion + "\\default.jsp");
PrintWriter out = new PrintWriter(outFile);
```

Instanciamos la ontología que vamos a utilizar para recoger los datos del archivo .owl que vamos a leer.

```
OntModel model = null;

model = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM_RULE_INF );
```

Abrimos el archivo con la ontología

```
java.io.InputStream in = FileManager.get().open( "Cloud_Computing.owl" );
if (in == null) {
   throw new IllegalArgumentException("Archivo no encontrado");
}
```

Leemos el archivo RDF/XML

```
model.read(in, "");
```

Posteriormente mediante un iterador recorremos la clases

```
ExtendedIterator iteratorClasses = model.listClasses();
```

Ahora empezamos a construir el archivo .jsp mediante las correspondientes etiquetas html y jsp.

```
out.println("<html>");
out.println("<head>");
out.println("<meta http-equiv='content-type' content='text/html; charset='iso-8859-1'>");
    //out.println("<title>Arbol de Navegación de la ontología</title>");
out.println("<%@page import=\"TFC.TextoPropiedades\"%>");
out.println("<%@page import=\"java.io.*,java.util.*\"%>");
    out.println("<link rel='stylesheet' href='css/folder-tree-static.css' type='text/css'>");
    out.println("<link rel='stylesheet' href='css/context-menu.css' type='text/css'>");
    out.println("<script type='text/javascript' src='js/ajax.js'></script>");
    out.println("<script type='text/javascript' src='js/folder-tree-static.js'></script>");
    out.println("<script type='text/javascript' src='js/context-menu.js'></script>");
out.println("</head>");
out.println("<body>");
```

Escribimos el titulo mediante una etiqueta H2

```
 out.println("<H2><CENTER><font color=red>Navegador Ontología Cloud
Computing</font></center></H2>");
    out.println("<table border=0 width=100%>");
```

Escribimos el titulo del menú arbol

```
    out.println("<tr><td><font color=blue><i>Arbol de la
Ontología</i></font></td><td><center><font color=blue><i>Desplieguel el arbol , haga click en
la propiedad cuya información desea ver y en este apartado se muestra el valor de la
Propiedad</font> </center></td></tr>");
    out.println("<tr>");
```

Escribimos la leyenda de los iconos del árbol.

```
out.println("<td>Leyenda: <img src='images/dhtmlgoodies_folder.gif'><i><font
color=green>Clase <img src='images/dhtmlgoodies_sheet.gif'>Slot</font></td>");
```

Escribimos el titulo de la clase y propiedad cuya información estamos visionando al haberla seleccionando el menú árbol.

```
 out.println("<td><center><font color=green>Se muestra la información de la
clase: <b><%= request.getParameter(\"param1\")%></b>  y de la
propiedad <b><%= request.getParameter(\"param2\")%></font></td>");
```

Recorremos el iterador y según nos devuelva una clase o una propiedad la añadimos al menú

```
      while ( iteratorClasses.hasNext() ){
      OntClass ontClass = (OntClass) iteratorClasses.next();

      out.println("<li><a href='#' id=100>"+ ontClass.getLocalName() + "</a>");
      ExtendedIterator iteratorProperties= ontClass.listDeclaredProperties();

      while (iteratorProperties.hasNext() ){

      OntProperty ind= (OntProperty) iteratorProperties.next();
      if (!property){
            out.println("<ul>");
            property=true;
        }
       //out.println("<li class='dhtmlgoodies_sheet.gif'>"+ ind.getLocalName() + "</a>");
       out.println("<li class='dhtmlgoodies_sheet.gif'><a
href='http://localhost/default.jsp?param1=" + ontClass.getLocalName() + "&param2=" +
ind.getLocalName() +"' id='node_101'>"+ ind.getLocalName() + "</a>");
        }
 ExtendedIterator iteratorSubClasses = ontClass.listSubClasses();

      while ( iteratorSubClasses.hasNext() ){
         /*subclass=false;
         if (!subclass){
            out.println("<ul>");
            subclass=true;
         }***/
      OntClass ontSubClass = (OntClass) iteratorSubClasses.next();
      if(ontSubClass.getNameSpace().equals("cloud:") &&
(!ontSubClass.getLocalName().equals("CAAS") &&
!ontSubClass.getLocalName().equals("DAAS"))){
      out.println("<li><a href='#' id=100>"+ ontSubClass.getLocalName() + "</a>");
      ExtendedIterator iteratorPropertiesSubClass= ontSubClass.listDeclaredProperties();

      while (iteratorPropertiesSubClass.hasNext() ){

      OntProperty indSubClass= (OntProperty) iteratorPropertiesSubClass.next();
      if (!propertySubClass ){
            out.println("<ul>");
            propertySubClass=true;
         }
      //out.println("<li class='dhtmlgoodies_sheet.gif'>"+ indSubClass.getLocalName() +
"</a>");
```

```
      out.println("<li class='dhtmlgoodies_sheet.gif'><a
href='http://localhost/default.jsp?param1=" + ontSubClass.getLocalName() + "&param2=" +
indSubClass.getLocalName() +"' id='node_101'>"+ indSubClass.getLocalName() + "</a>");

      }
       propertySubClass=false;
       finpropertySubClass=false;
      if (!finpropertySubClass && !ontSubClass.getLocalName().equals("IAAS") &&
!ontSubClass.getLocalName().equals("CAAS") &&
!ontSubClass.getLocalName().equals("DAAS")){
            out.println("</ul>");
            finpropertySubClass=true;
        }

      ExtendedIterator iteratorSubSubClasses = ontSubClass.listSubClasses();
      boolean subsubclass=false;

      while ( iteratorSubSubClasses.hasNext() ){
        finsubclass=false;
         if(!subsubclass && !ontSubClass.getLocalName().equals("IAAS") ){
      out.println("<ul>");
      subsubclass=true;
        }
      OntClass ontSubSubClass = (OntClass) iteratorSubSubClasses.next();
      out.println("<li><a href='#' id='100'>" + ontSubSubClass.getLocalName() + "</a>");
      ExtendedIterator iteratorPropertiesSubSubClass=
ontSubSubClass.listDeclaredProperties();

      while (iteratorPropertiesSubSubClass.hasNext() ){

      indSubSubClass= (OntProperty) iteratorPropertiesSubSubClass.next();
      if (!propertySubSubClass){
            out.println("<ul>");
            propertySubSubClass=true;
        }
       //out.println("<li class='dhtmlgoodies_sheet.gif'><a href='#' id='node_101'>"+
indSubSubClass.getLocalName() + "</a>");
        out.println("<li class='dhtmlgoodies_sheet.gif'><a
href='http://localhost/default.jsp?param1=" + ontSubSubClass.getLocalName() +
"&param2=" + indSubSubClass.getLocalName() +"' id='node_101'>"+
indSubSubClass.getLocalName() + "</a>");
      }
       propertySubSubClass=false;
       finpropertySubSubClass=false;
      if (!finpropertySubSubClass){
            out.println("</ul>");
            finpropertySubSubClass=true;
        }
      }
      if (!finsubclass)
      out.println("</ul>");
      finsubclass=true;
      }}
      }
```

```
}
out.println("</ul>");
out.println("</ul>");
out.println("<br>");


out.println("<a href='#' onclick=\"expandAll('dhtmlgoodies_tree');return false\">Expandir
todo</a>");
out.println("<a href='#' onclick=\"collapseAll('dhtmlgoodies_tree');return false\">Contraer
todo</a>");
```

Finalmente y como una de las partes más importante del navegador recuperamos la información de la instancia de la propiedad seleccionada en el menú árbol mediante una llamada a la clase TextoPropiedad y al método LeerDatos de la propiedad pasándolo como parámetros clase y propiedad.Esta clase recupera la información y la muestra en la parte central del navegador.

```
valor=TextoPropiedades.LeerPropiedad(request.getParameter(\"param1\"),request.getParame
ter(\"param2\"));%>");
out.println("<%=valor%>");
```

El código de la clase TextoPropiedad y del método LeerDatosPropiedad es el siguiente:

```
model.read(in, "");
   for (Iterator<OntClass> i = model.listClasses();i.hasNext();){
          OntClass cls = i.next();
          if(cls.getLocalName().equals(Class)){
       for(Iterator it = cls.listDeclaredProperties();it.hasNext();){
        OntProperty prope= (OntProperty) it.next()  ;
        if (prope.getLocalName().equals(Property)){
        for(Iterator it2 = cls.listInstances(true);it2.hasNext();){
                    Individual ind = (Individual)it2.next();
                    if(ind.getOntClass().getLocalName().equals(Class)){
                Statement prop= ind.getProperty(prope);
                System.out.println(prop.getString());
                valor=prop.getString(); }}}}
```

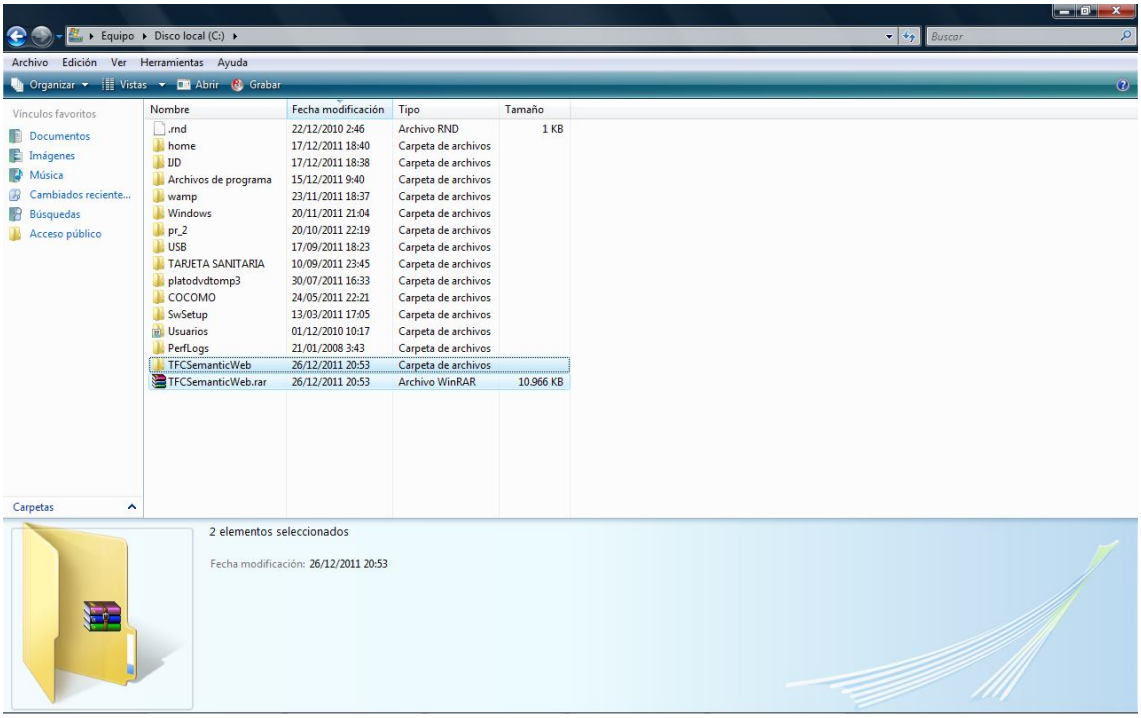
6. INSTRUCCIONES FUNCIONAMIENTO PROTOTIPO SOFTWARE

(Plataforma Windows)

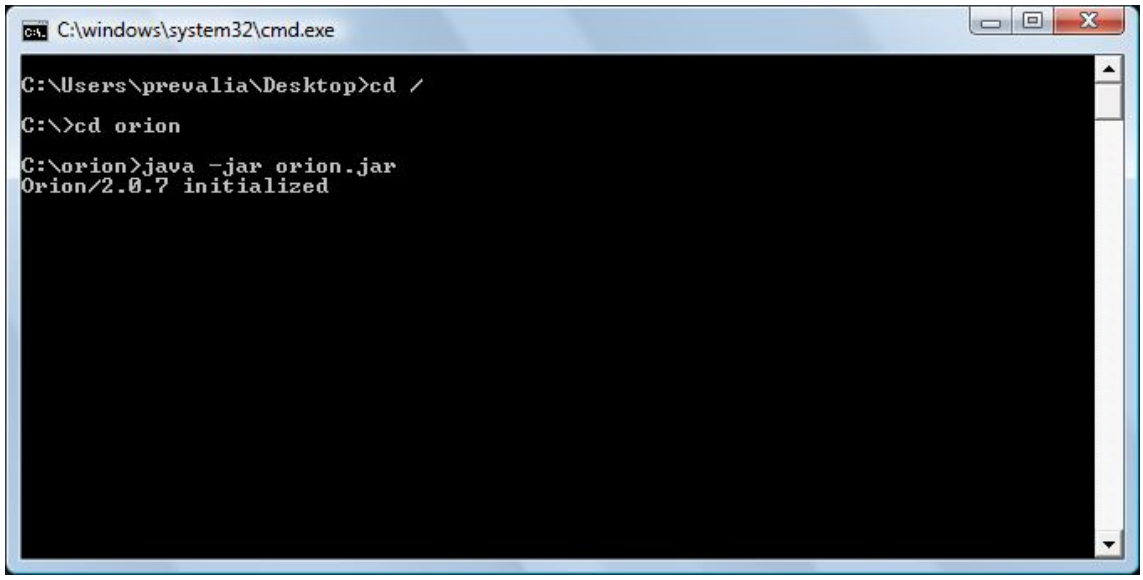
ES IMPRESCINDIBLE TENER EL JDK DE JAVA 1.6 Y EL JRE 6.Las variables de entorno también deben estar configuradas

ES NECESARIO CONEXIO A INTERNET PARA EXTRAER LOS DATOS DE WIKIPEDIA Y DBPEDIA.

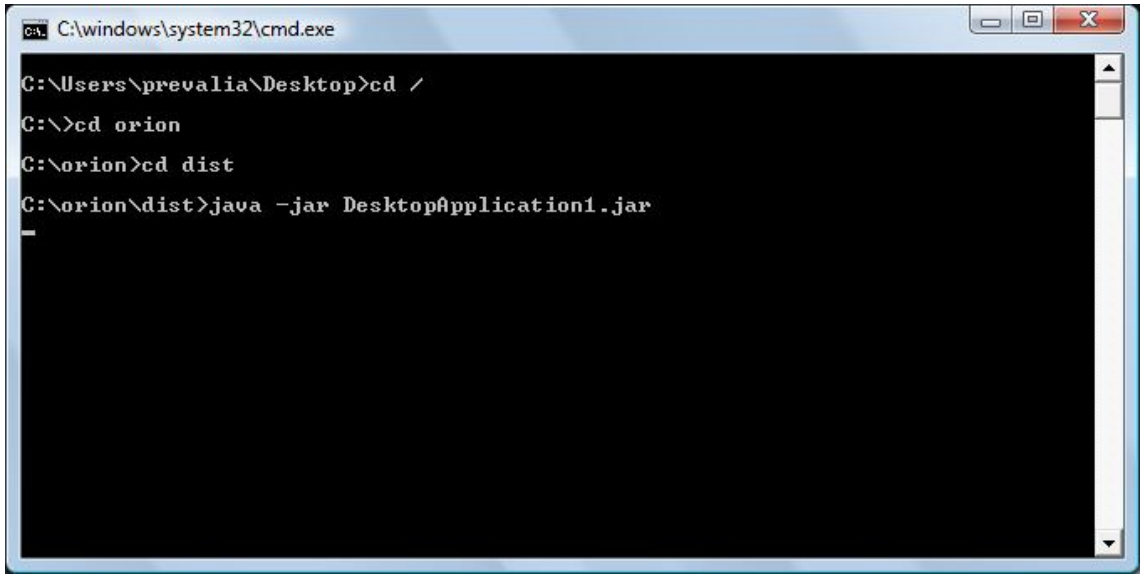
Paso 1. Descomprimir el archivo zip en la raíz C: tal como se puede ver la imagen:



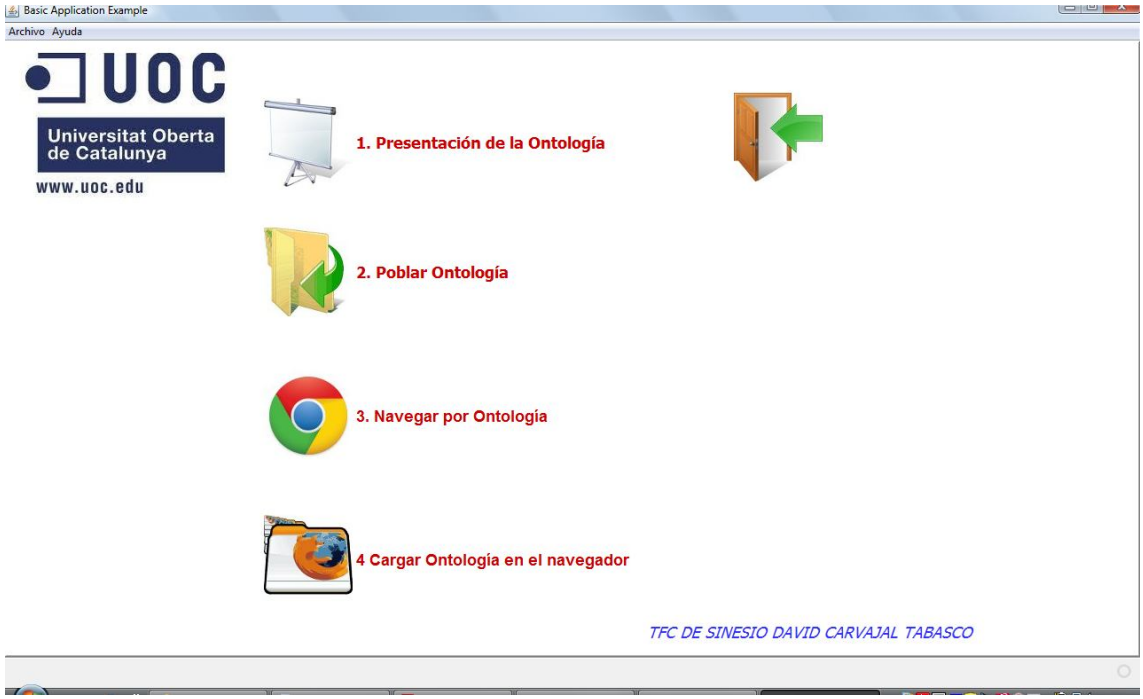
Paso 2. Es necesario ejecutar el script TFC_orion.bat que ejecuta y pone en marcha el servidor orion.



Paso 3. Es necesario ejecutar el script TFC.bat que **INICIA LA APLICACIÓN DE TFC.** Este script contiene las sentencias correctas para situarse en el path de la aplicación y ejecutar el .jar de la misma

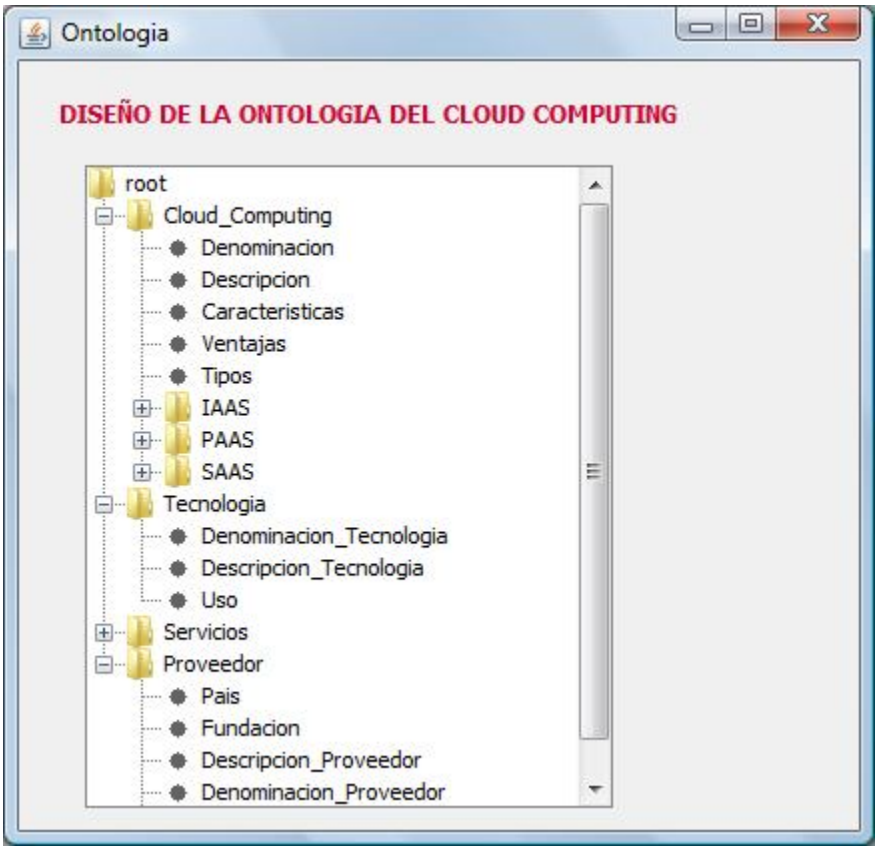


Una vez ejecutada la aplicación nos aparece la ventana de inicio de la misma.



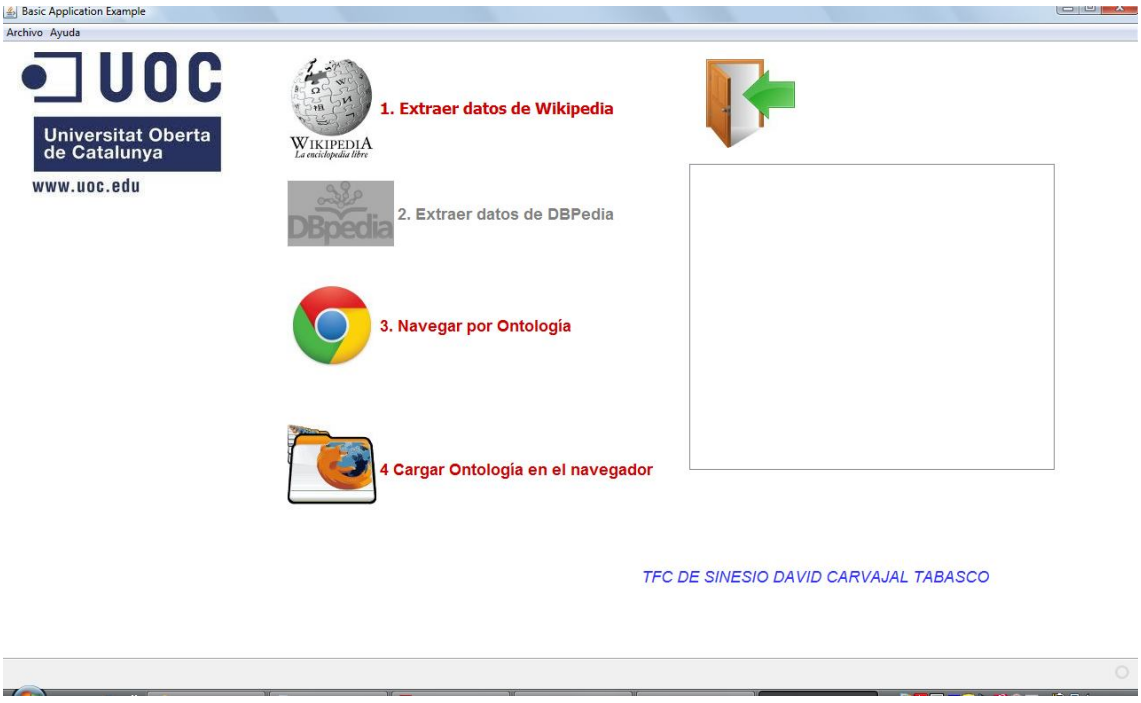
Paso 4. Ahora ya estamos en la aplicación , simplemente debemos de elegir la opción que corresponda. El proceso que hay que seguir es lineal y es el siguiente:

1.Pulse la opción de **Presentación de la ontología** para acceder a una presentación del diseño de la misma y conocer más sobre ella como vemos en la siguiente imagen:



En esta pantalla se despliegan sus clases y propiedades.

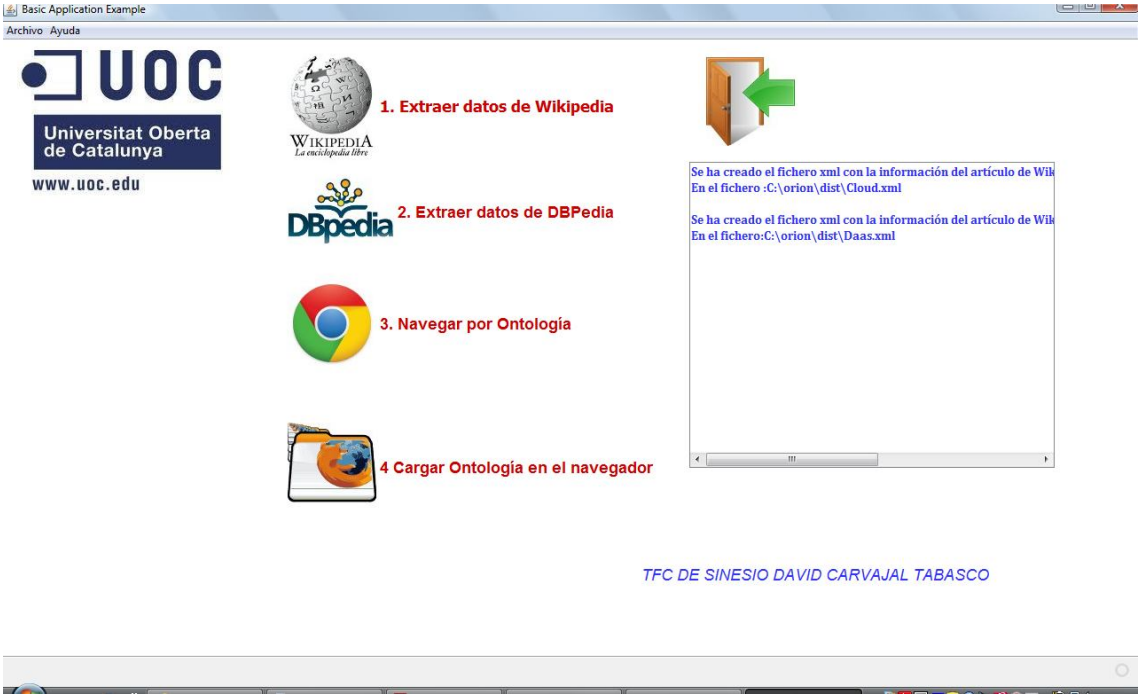
2.En este paso debemos pulsar la opción **Poblar la ontología** le saldrá la siguiente pantalla:



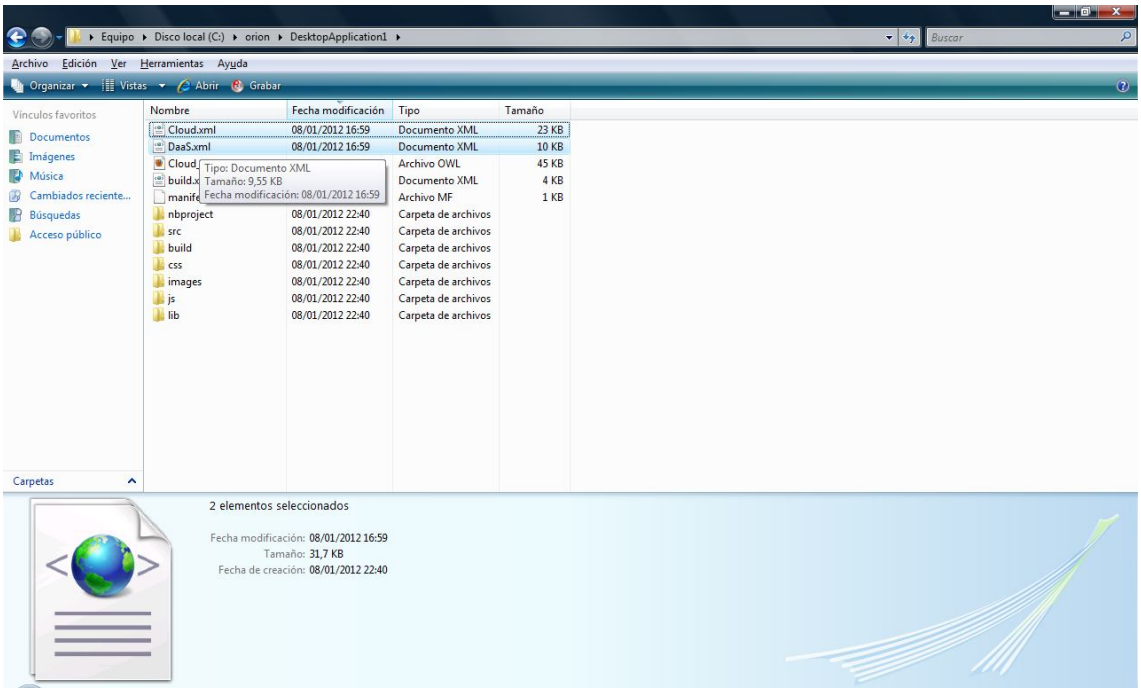
En esta pantalla debe seguir el siguiente itinerario:

En primer lugar click sobre la opción de Extraer datos de Wikipedia para extraer en un archivo xml la información de las propiedades de los slots de la ontología cuya información no se ha encontrado en Wikipedia.

Al pulsar esta opción se deberá cargar el archivo y aparecerá un aviso en el cuadro de texto de la derecha tal como podemos ver en la siguiente imagen.



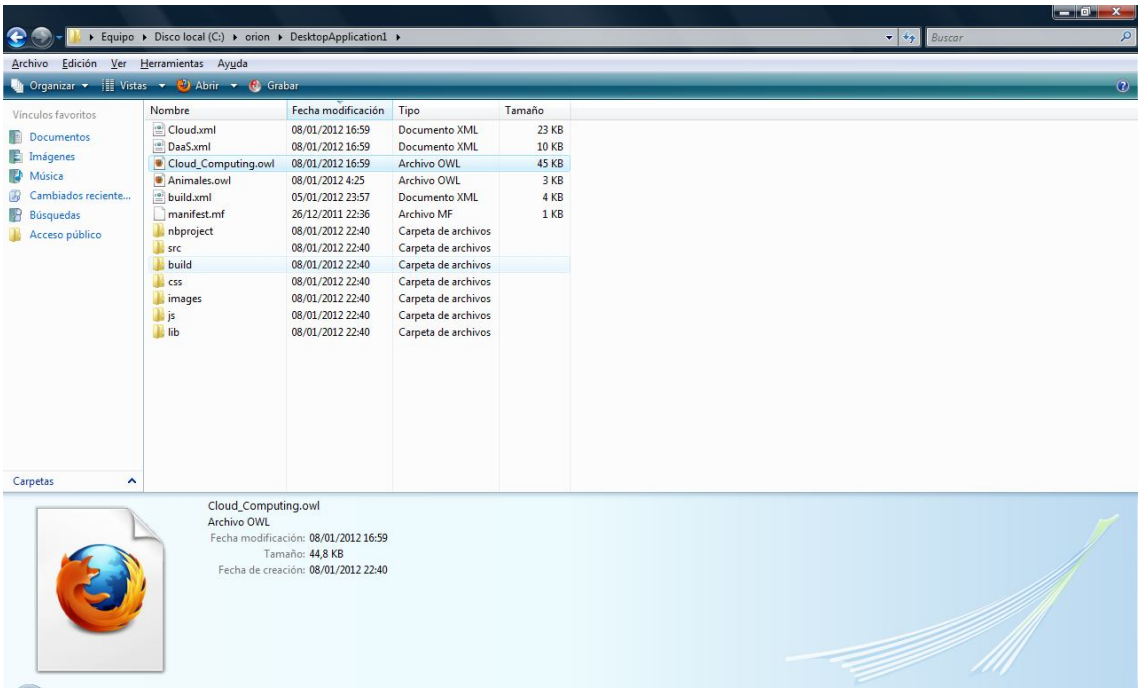
La información queda almacenada en dos archivos.xml ***Cloud.xml y DAAS.xml*** en el path de la aplicación como se puede ver en la siguiente imagen.



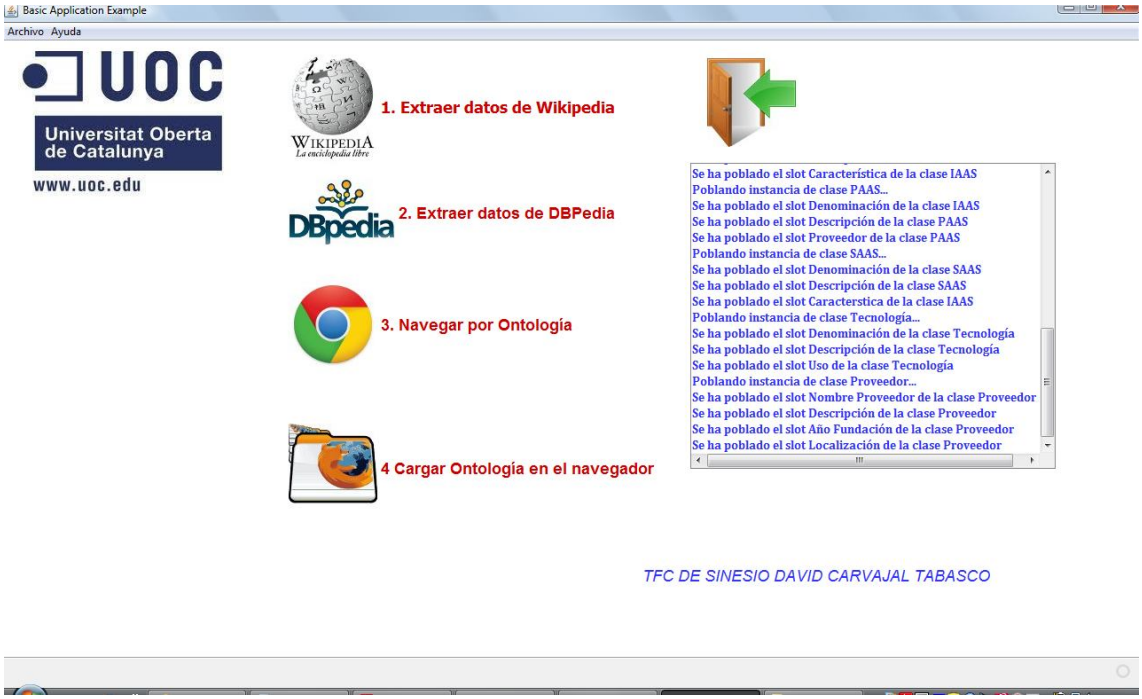
Una vez terminado el proceso de extracción de información y de creación de los archivos para almacenar dicha información, se habilitará la opción de Extraer Datos de DBPedia que forma parte del siguiente punto y que detallamos a continuación.

En segundo lugar debemos pulsar la opción Extraer Datos de DBPedia esta opción realiza dos acciones:

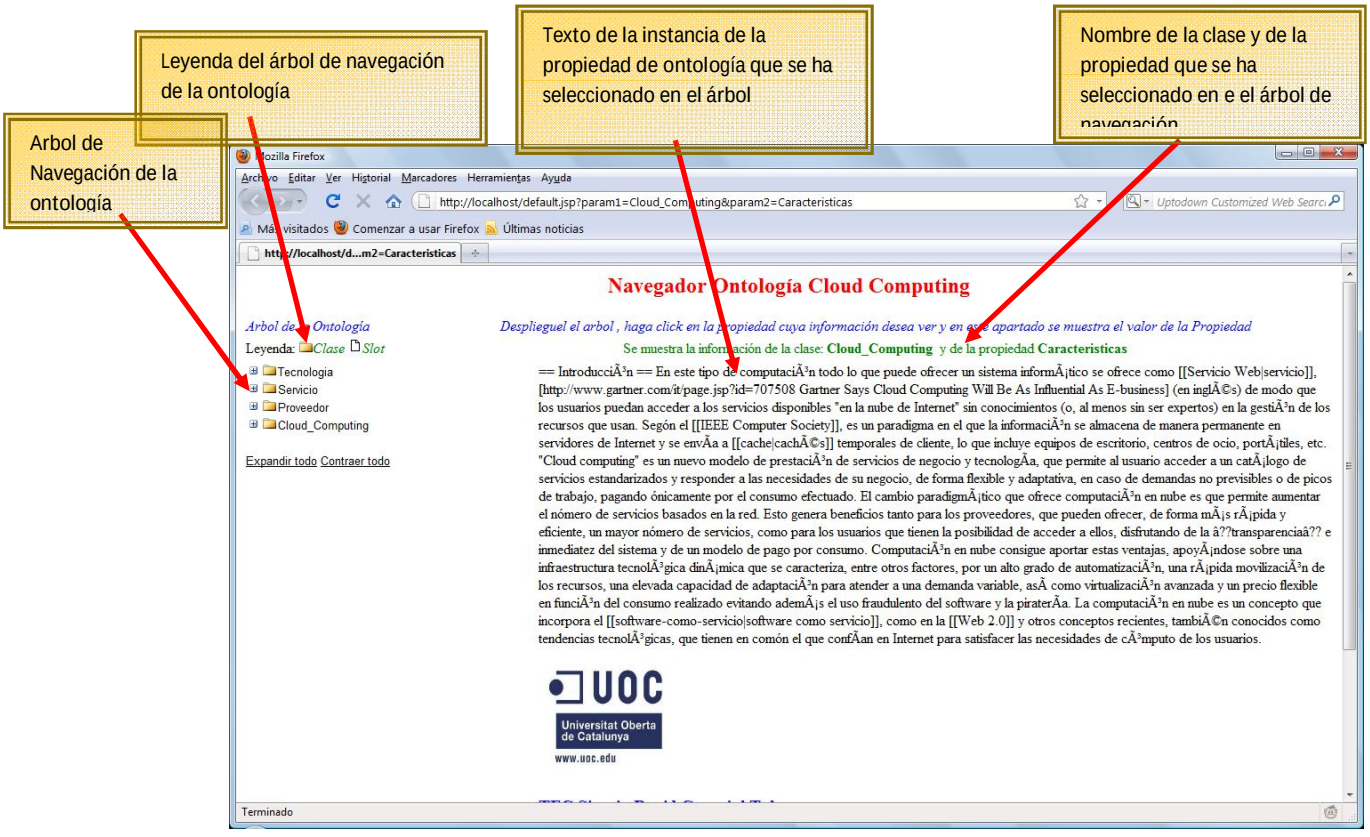
A. **CREA LA ONTOLOGÍA EN TIEMPO REAL** utilizando las librerías de Jena al respecto tal como se ha explicado en el apartado 5.6.4.La ontología se guarda en un archivo llamado Cloud_Computing.owl en el path de la aplicación. Se muestra en la siguiente imagen:



B. Extraer información del archivo Cloud.xml que descargo la información de Wikipedia en el paso anterior y de DBPedia y rellenar una instancia de la ontología. Cuando haya terminado aparecerá un mensaje en el cuadro de la derecha de la pantalla , tal como podemos ver en la siguiente imagen.(**ALERTA, hasta que no aparezca el mensaje de población de la ontología no realizar el paso siguiente**).



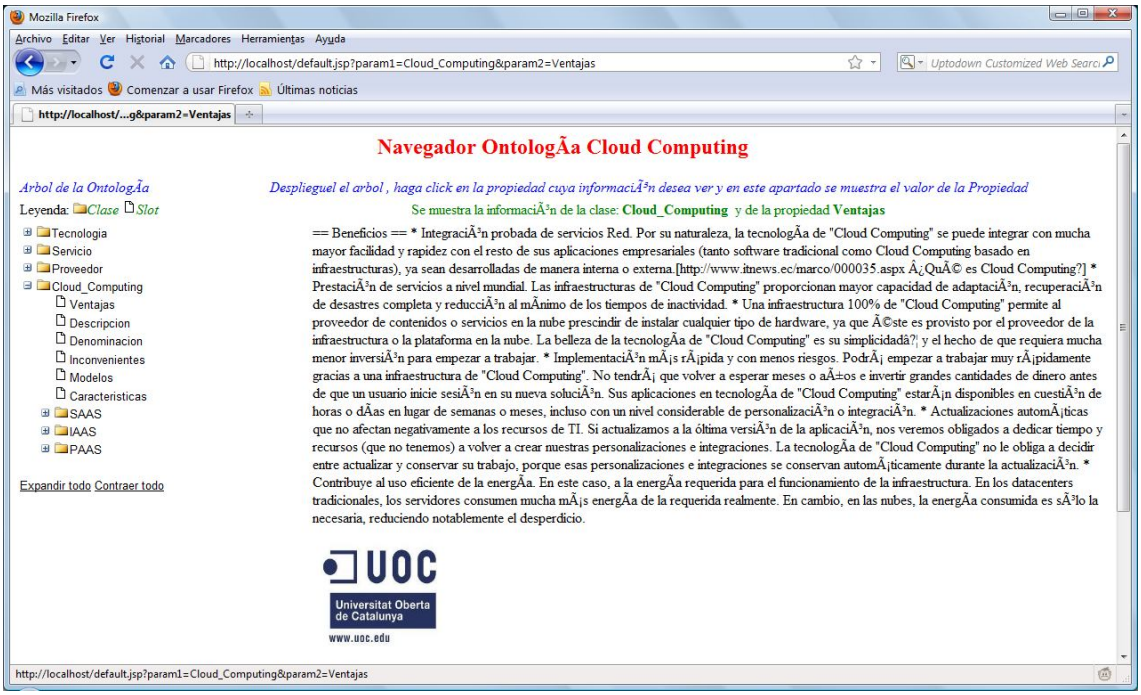
Paso 5 Ahora debemos hacer click en la opción navegar por la ontología de manera que se nos abrirá una web .jsp (**DEBE ESTAR LIBRE EL PUERTO 80**) en la dirección http://localhost . El navegador tendrá el aspecto que se muestra en la siguiente imagen:



Como podemos ver en la imagen en el árbol de la derecha se muestran las clases que conforman la ontología con sus propiedades. Este árbol se puede ir desplegando y

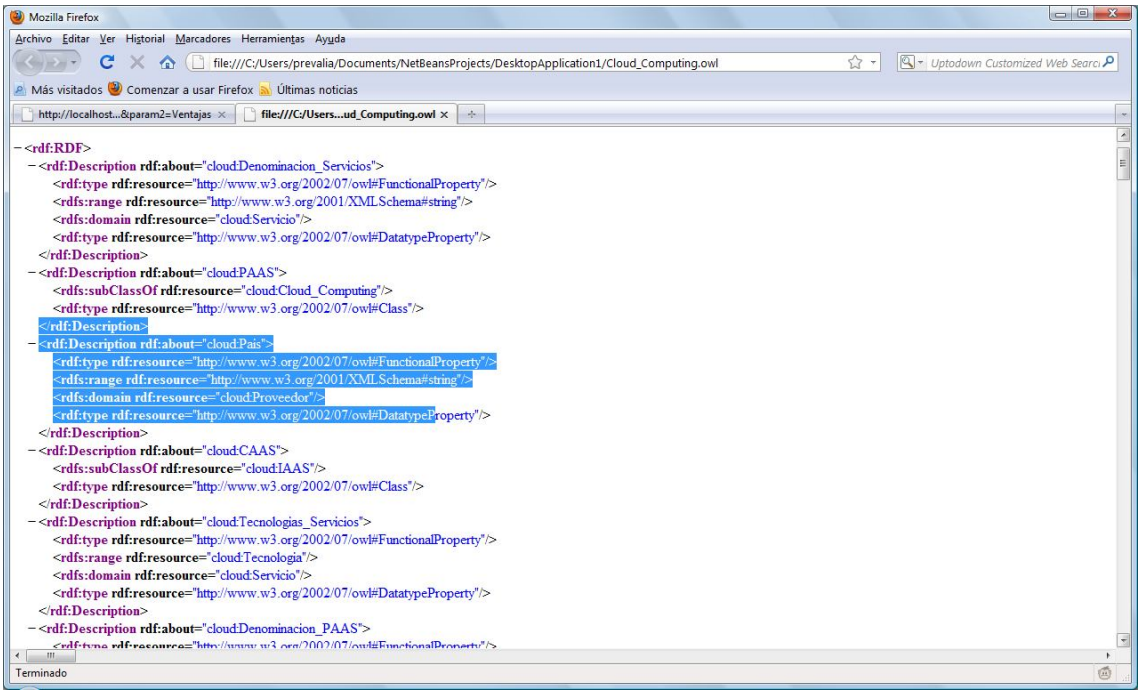
navegando sobre él para conocer la composición de la ontología .Al final del menú árbol aparecen además las opciones Contraer todo y Expandir todo (contrae o expande las opciones del árbol).Existe una leyenda que explica el significado de los iconos de cada elemento del árbol .Un icono de archivo es una clase y un icono de una hoja es una propiedad de la misma. Además cada subclase queda encuadrada dentro de la clase correspondiente.

Cada elemento del árbol es un enlace y al pulsar sobre él en el apartado de la derecha se muestra el texto de la instancia de la propiedad que se ha elegido. Así si pulsamos sobre la propiedad Ventajas de la clase Cloud Computing se mostrara el texto correspondiente a las ventajas del Cloud Computing. En la parte superior del texto se muestra el nombre de la clase y de la propiedad que estamos visitando. Así lo podemos ve en la siguiente imagen:



Esta funcionalidad resulta muy útil para navegar por la ontología y desplegar y conocer sus clase, propiedades y la información de la instancia poblada.

Paso 6 La última opción del menú es la de Cargar la ontología en el navegador. Al elegir esta opción se despliega el archivo .owl en el navegador(optimizado para Firefox , Google Chrome).De esta manera podemos navegar por la ontología con lenguaje RDF-XML y conocer más a fondo las características de los lenguajes de la web semántica y de la ontología de Cloud Computing diseñada con Protegé.En la siguiente imagen podemos comprobar cómo se mostraría el archivo Cloud_Computing.owl creado anteriormente al pulsar la opción Extraer Datos de DBPedia.



7. ENTREGABLES

Los entregables de este Trabajo Fin de Carrera son los que detallamos a continuación:

- Documento PDF memoria Final
- Archivo de proyecto protege con el diseño de la ontología Cloud Computing
- Archivo .rar TFC con el empaquetado de la aplicación + la carpeta que contiene el server de JSP orion + los scripts de inicio de la aplicación y de orion.
- Video explicativo del funcionamiento de la aplicación. El archivo es un .swf que se encuentra en el empaquetado de entregable del proyecto pero además existe un html que embebe ese archivo. Desde es html llamado TFC.html se puede visionar el video explicativo del funcionamiento del software.

8. GLOSARIO

CaaS:(Comunication as a Service) Comunicaciones como Servicio

Cloud Computing: Computación en nube. Servicios de informática ofrecidos a través de Internet.

DaaS:(Data as a Service) Almacenamiento de Datos como Servicio

Eclipse: es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores .IDE que permite trabajar con Java u otros lenguajes.

Escalabilidad: es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para extender el margen de operaciones sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

IaaS: (Infraestructure as a Service) Infraestructura como Servicio

Jena: API de Java para trabajar con ontologías

Multitenancy: es un concepto donde une sola copia de la aplicación corre en un servidor y es usada simultáneamente por varios clientes (tenants). La aplicación está diseñada para repartir su información y su configuración, para que así cada empresa pueda trabajar con una instancia personalizada y privada de la aplicación.

OWL es el acrónimo del inglés Web Ontology Language, un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW. OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML.

Ontología: El término ontología en informática hace referencia a la formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados; con la finalidad de facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades.

PaaS:(Platform as a Service) Plataforma como servicio

Parser: Un analizador sintáctico (en inglés parser) es una de las partes de un compilador que transforma su entrada en un árbol de derivación.

Protegé: Editor de ontologías open Source

SaaS:(Software as a Service) Software como Servicio

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

SPARQL es un acrónimo recursivo del inglés SPARQL Protocol and RDF Query Language. Se trata de un lenguaje estandarizado para la consulta de grafos RDF, normalizados por el RDF Data Access Working Group (DAWG) del Word Wide Web Consortium (W3C).

Virtualización: es la creación -a través de software- de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red.

Web Semántica:(Software as a Service) Software como Servicio

XML, siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

9. BIBLIOGRAFIA

1. [INTERNET] GUIA BREVE DE LA WEB SEMANTICA.
http://www.w3c.es/divulgacion/guiasbreves/websemantica

2. CONTRERAS JESUS Y MARTINEZ COMECHE JUAN ANTONIO. Tutorial de Ontologías. Universidad Complutense de Madrid.
http://www.sedic.es/gt_normalizacion_tutorial_ontologias.pdf

3 BERNES LEE TIM , HENDELER JAMES , LASSILA ORA Scientific American

http://www.semanticweb.org/, http://www.ontology.org/

4. [INTERNET] TECNOLOGIAS XML Y WEB SEMANTICA
http://www.di.uniovi.es/~labra/cursos/ver04/pres/SemWeb1.pdf

5. [INTERNET] WEB SEMANTICA
http://usuarios.multimania.es/arceizb/LaWebSemantica.pdf

6[INTERNET IMAGEN] LA WEB SEMANTICA

http://sociedadinformacion.fundacion.telefonica.com/DYC/SHI/seccion=1188&idioma=es_ES&id=2009100116310013&activo=4.do?elem=4299

7.[INTERNET] UNICODE EN WIKIPEDIA

http://es.wikipedia.org/wiki/Unicode

8. [INTERNET] URI

http://nicolasmendez.com.ar/blog/index.php/2011/10/26/que-es-una-uri-url-wtf/

9. [INTERNET] ONTOLOGIA [INFORMATICA]

http://enciclopedia.us.es/index.php/Ontolog%C3%ADa_%28Inform%C3%A1tica%29

10.[INTERNET] CLOUD COMPUTING WIKIPEDIA

http://es.wikipedia.org/wiki/Unicode

11.[INTERNET] *RESOURCE DESCRIPTION FRAMEWORK*

www.bib.uc3m.es/~mendez/publicaciones/7jc99/**rdf**.htm

12.[INTERNET] *RDF*

www.cobdc.org/jornades/7JCD/1.pdf

13.[INTERNET] *SESAME*

http://e-archivo.uc3m.es/bitstream/10016/6566/1/PFC_Jorge_Hernandez.pdf

14 .[INTERNET] ECLIPSE

http://es.wikipedia.org/wiki/Eclipse_(software)

15[INTERNET] JENA

http://www.gnoss.com/comunidad/nextweb/recurso/Jena-Semantic-Web-Framework/1c461543-43c2-407e-aa51-f63861b91624