Linked Data

CARLOS ALBERTO BREWER RAMOS

MASTER EN INGENIERIA INFORMATICA

WEB SEMANTICA Y REPRESENTACION DEL CONOCIMIENTO

Felipe Geva Urbano

Jordi Conesa Caralt

Enero de 2017

LICENCIA

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Linked Data*
Nombre del autor:	*Carlos Alberto Brewer Ramos*
Nombre del consultor/a:	*Felipe Geva Urbano*
Nombre del PRA:	*Jordi Contesa Caralt*
Fecha de entrega (mm/aaaa):	01/2017
Titulación::	*Master en Ingeniería Informatica*
Área del Trabajo Final:	*Web Semantica y Representacion del Conocimiento*
Idioma del trabajo:	*Español/Castellano*
Palabras clave	*Linked, Data, Web*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El presente documento presenta las memorias resultado de la realización del proyecto de fin de maestría de los estudios de MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA de la Universidad Oberta de Catalunya.

Este proyecto tiene como finalidad el cumplimiento de dos objetivos:

El primer objetivo es entender y analizar en profundidad que es el “linked data” y todos los conceptos de la “web semántica”, además de conocer algunos repositorios de datos de esta misma y conocer su operatividad.

El segundo objetivo consiste en Implementar una API (Interfaz de Aplicación de Programa) de forma que utilice la información de la web de datos para identificar la bibliografía de un autor (o conjunto de ellos).

El proyecto se enmarca en un contexto meramente académico, de donde se parte que los desarrollos logrados durante el mismo no tienen aplicación comercial dadas sus limitaciones en funcionalidad y operatividad aunque posteriores proyectos puedan ampliar la operatividad de los desarrollos aquí logrados

La metodología usada en el desarrollo del proyecto consistió en una etapa de aprendizaje e investigación y posterior a esta una etapa de desarrollo de una solución en software que aplicase los conocimientos adquiridos. Las dos

etapas se dividieron en seis segmentos a fin de llevar de manera más fácil el desarrollo de la solución.

De este trabajo se concluye que se logró una comprensión adecuada del concepto de “linked data” y la web semántica. Además, se desarrolló una API con un motor de búsqueda básico y un motor de carga de datos semánticos.

Abstract (in English, 250 words or less):

This document contains the results of the end of master project for engineering informatics at Universidad Oberta de Catalunya.

This project has two main objectives:

First one is to understand and analyze what is linked data and all semantic web concepts; also review some data repositories that belongs to the semantic web concept and find out how these repositories work.

Second one is to implement and API (application programing interface) that uses semantic web data to identify an author or set of authors’ bibliography.

This project is contextualized on a mere academic environment, not intended for commercial use, given limitations to the way the concepts are developed here and operational restrictions of the developed solution. Further projects may enhance these developments.

This project was divided into two stages, first stage was intended solely to develop an understanding of all knowledge related to the semantic web and linked data concepts; second stage was dedicated to develop a software solution win the intention to apply all acquired knowledge from the first stage. Both stages were subdivided into six sub stages to better undergo all the process.

As a conclusion of this work, a good understanding of both linked data and semantic web concepts was achieved, also a basic API to search for semantic data was developed and a very basic search engine was created to help in the process of crawling for semantic information on authors and its bibliography.

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

LISTA DE TABLAS

1. INTRODUCCION

El presente documento presenta las memorias resultado de la realización del proyecto de fin de maestría de los estudios de MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA de la Universidad Oberta de Catalunya.

Este proyecto tiene dos partes a saber:

La primera entender y analizar en profundidad que es el “linked data” y todos los conceptos de la “web semántica”, además de conocer algunos repositorios de datos de esta misma y conocer su operatividad.

La segunda parte consiste en Implementar una API (Interfaz de Aplicación de Programa) de forma que utilice la información de la web de datos para identificar la bibliografía de un autor (o conjunto de ellos).

La API lee la información de un autor, sus publicaciones usando mini aplicaciones y código JAVA con librerías JENA en su versión 3.1.1, utilizando las funciones que incorpora la mencionada librería.

Las librerías que incorporan las funciones a usar en las clases de las API desarrolladas son: *jena.rdf.model.Model,.jena.rdf.model.ModelFactory, .jena.rdf.model.NodeIterator, .jena.rdf.model.Property, .jena.rdf.model.RDFNode, .jena.rdf.model.ResIterator, .jena.rdf.model.Resource, .jena.rdf.model.Statement, .jena.rdf.model.StmtIterator*

Las funciones incorporadas en las librerías en mención y usadas en el desarrollo son:

ModelFactory, listStatements, StmtIterator, listObjectsOfProperty, entre otras varias que se utilizaron para conseguir leer el modelo de una dirección URI determinada.

De esta manera se llega a desarrollar dos clases principales a través de las cuales se logra identificar una URI y acceder a los recursos y distintos enlaces que están contenidos en el documento RDF de la misma. Una vez ejecutado el código se puede obtener la información completa de esa dirección.

Las dos clases desarrolladas se exportan como API para permitir su uso en distintas aplicaciones desarrolladas en JAVA.

1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

El presente trabajo trata de aportar una visión actual del mundo de la web de los datos (conocida como web of data o linked data). Para hacerlo se propone hacer una aplicación en el ámbito de la investigación. En particular, se propone identificar, dado un autor, sus publicaciones sobre investigación realizadas en el campo de la informática. Para hacerlo habrá que desarrollar cierta habilidad en las tecnologías relacionadas con la web semántica, identificar repositorios de datos existentes y hacer una implementación que use esta tecnología para aprovechar esta información.

Hoy en día, uno de los retos en el mundo de la investigación, al igual que en el resto de los procesos donde se utiliza información, es la de gestionar la enorme cantidad de datos referentes a publicaciones que se encuentran en la red a nuestro alcance.
Una solución para gestionar esta gran cantidad de información es la aparición de un conjunto de buenas prácticas para la publicación y la conexión de datos estructurados en la Web conocidas como datos enlazados (linked data). Estas buenas prácticas se basan en tecnologías ya existentes que se conjugan para permitir la automatización del procesamiento de los datos, de forma que la web camine hacia un espacio de datos global que conecte los datos de varios dominios.

Los resultados esperados del Trabajo son:

- No se pretende entregar una aplicación comercial, la solución es meramente académica y para fines de exploración y comprensión de los conceptos de “web semántica” y “linked data”.
- Se espera lograr una completa comprensión de los conceptos relacionados con los temas de la investigación
- Se espera superar la asignatura y culminar el plan de estudios de la maestría con buena calificación.
- Al finalizar se quiere ser capaz de identificar los repositorios de datos y poder trabajar con ellos con agilidad.
- Ser capaz de identificar repositorios de datos enlazados a la web de datos
- Ser capaz de consultar diferentes repositorios de la web de datos para obtener información relevante
- Ser capaz de implementar servicios que accedan a la web de datos y usar la información que contiene

1.2 OBJETIVOS GENERALES DEL TRABAJO

- Entender qué es linked data en el ámbito de la web semántica.
- Hacer un análisis de Open Linked Data para identificar los repositorios de datos (las webs) que se pueden usar.
- Implementar la API para obtener las publicaciones a partir de un nombre de investigador, incluyendo nombre de los autores, título de la publicación, año de la publicación y si se tercia datos correspondientes a su edición.
- De manera opcional, el sistema podría identificar la lista de colaboradores del investigador, los investigadores más prolíficos dado un tema (una palabra o conjunto de palabras clave, etc.).

OBJETIVOS ESPECIFICOS DEL TRABAJO

- Desarrollar una API para la búsqueda y gestión de información bibliográfica por autor.
- Entregar una lista de repositorios de datos usados en la exploración de conocimiento
- Entender y comprender SPARSQL y dejar establecida una pequeña base de conocimientos alrededor de esta solución.
- Entender los conceptos acerca de linked data y web semántica

1.3 ENFOQUE Y MÉTODO SEGUIDO

Se ha escogido un enfoque de dos partes principales las cuales a su vez se dividieron en seis sub etapas a fin de facilitar el cumplimiento de los objetivos del proyecto en un marco de tiempo razonable.

Desde el principio se definió que se iba a crear un producto nuevo, cual es la API solicitada en el planteamiento del TFM así como crear un pequeño resumen con los contenidos y teoría detrás de los conceptos de la web semántica y linked data.

Ha sido esta la estrategia apropiada pues permite dividir el trabajo total en segmentos pequeños y de fácil manipulación con lo cual el esfuerzo aplicado se reduce de manera significativa.

1.4 PLANIFICACION DEL TRABAJO.

Los recursos necesarios para desarrollar el proyecto consisten en:

- Computadores estándar
- Acceso a internet
- Recursos estándares definidos por la UOC

El trabajo se dividió en dos etapas y seis actividades a continuación:

- Selección de autor y repositorio
- Revisión preliminar URI/Lectura del modelo RDF de una dirección dada
- Configurar API dentro de la interfaz de JAVA seleccionada
- Crear una mini aplicación y usar la RDF valida
- Ingresar RDF de autor
- Obtener datos del autor. Vía consola

Un detalle del trabajo puede verse en el anexo 1.

1.5 SUMARIO DE LOS PRODUCTOS OBTENIDOS

Se logró crear una API en JAVA usando el framework para web semántica JENA, dicha API permite hacer búsquedas por nombre de autores y de estos cargar el modelo de datos RDF con información relevante a su bibliografía, coautorías y publicaciones.

La API fue creada como consecuencia lógica del proceso de aprendizaje de los conceptos enmarcados en la web semántica y linked data.

1.6 CONTENIDO DE ESTE DOCUMENTO

El presente documento se divide en dos partes principales, la primera es una introducción al objetivo de este trabajo, seguido esto por un compendio de los conceptos de la web semántica como son su definición, la definición de linked data, los conceptos de RDF, SPARQL, OWL y apache JENA.

Continúa entonces el documento presentando un resumen de las actividades desarrolladas en aras de conseguir la elaboración de la API, incluye esto, los diagramas, código fuente y explicaciones asociadas a estos conceptos.

2. DESCRIPCIÓN DEL TRABAJO

El presente trabajo trata de aportar una visión actual del mundo de la web de los datos (conocida como web of data o linked data). Para hacerlo se propone hacer una aplicación en el ámbito de la investigación. En particular, se propone identificar, dado un autor, sus publicaciones sobre investigación realizadas en el campo de la informática. Para hacerlo habrá que desarrollar cierta habilidad en las tecnologías relacionadas con la web semántica, identificar repositorios de datos existentes y hacer una implementación que use esta tecnología para aprovechar esta información.

Hoy en día, uno de los retos en el mundo de la investigación, al igual que en el resto de los procesos donde se utiliza información, es la de gestionar la enorme cantidad de datos referentes a publicaciones que se encuentran en la red a nuestro alcance.

Una solución para gestionar esta gran cantidad de información es la aparición de un conjunto de buenas prácticas para la publicación y la conexión de datos estructurados en la Web conocidas como datos enlazados (linked data). Estas buenas prácticas se basan en tecnologías ya existentes que se conjugan para permitir la automatización del procesamiento de los datos, de forma que la web camine hacia un espacio de datos global que conecte los datos de varios dominios.

Las publicaciones de los actores involucrados en investigación se graban en diferentes webs de referencia. Una de ellas en el ámbito de la informática es el DBLP (http://dblp.uni-trier.de/). Muchas de estas web almacenan sus datos en RDF y ofrecen end points para consultar los datos. En el caso del DBLP tenemos la web con la información del RDF (https://datahub.io/dataset/l3) y el end point (http://dblp.l3s.de/d2r/). Podemos ver un ejemplo de la web del DBLP a http://dblp.uni-trier.de/pers/ y los mismos datos extraídos vía SPARQL del end point (http://dblp.l3s.de/d2r/).

Hay otras webs que se podrían utilizar, como por ejemplo Bibsonomy (http://linkeddatacatalog.dws .). También se puede buscar otros repositorios que se puedan consultar, como por ejemplo

La conexión de datos estructurados en la Web conocida como “linked data” hace parte de lo que se denomina Web Semantica. De la cual del siguiente sitio web se extrae este texto:

De forma similar a como la presentó Tim Berners-Lee en 1998 y adaptando los ejemplos al hipertexto, podemos decir que la World Wide Web, basada en documentos y enlaces de hipertexto, fue diseñada para la lectura humana y no para que la información que contiene pudiera procesarse de forma automática. Si hacemos una búsqueda de documentos, por ejemplo, por el término "hipertexto", la Web no distingue entre los distintos significados o contextos en los que aparece este término (programas para diseñar hipertexto, información docente, empresas que anuncian su web, etc.). La Web actual tampoco permite automatizar procesos, como por ejemplo, buscar un seminario sobre hipertexto, hacer la reserva de plaza, consultar los medios de transporte disponibles hasta la ciudad donde se celebre el evento, reservar billete, y conseguir un plano de dicha ciudad. Aun utilizando un potente buscador, se pierden muchas horas navegando por los resultados obtenidos tras la consulta, para acceder a la información de forma manual, cuando esto lo podría hacer un programa o agente inteligente.

La Web Semántica vendría a ser una extensión de la Web actual dotada de significado, esto es, un espacio donde la información tendría un significado bien definido, de manera que pudiera ser interpretada tanto por agentes humanos como por agentes computerizados.

La Web Semántica ha sido impulsada por Tim Berners-Lee, creador de la WWW, y otras personas relacionados con el W3C (World Wide Web Consortium). El primer avance en este sentido, fue la publicación en septiembre de 1998, por parte de Berners-Lee de 2 documentos denominados Semantic Web Road Map y What the Semantic Web can represent.

En el año 2000, Berners-Lee ofreció una conferencia en el marco del W3C donde propuso: "La nueva información debe ser reunida de forma que un buscador pueda "comprender", en lugar de ponerla simplemente en una "lista". La Web semántica sería una red de documentos "más inteligentes" que permitan, a su vez, búsquedas más inteligentes. La idea sería aumentar la inteligencia de los contenidos de las páginas web dotándolas de contenido semántico. La Web actual posee una gran capacidad para almacenar datos y puede leer y visualizar los contenidos, pero no es capaz de pensar ni de entender todo lo que contiene. Se precisa, por lo tanto, un nueva Web -la Web semántica- que hará posible no sólo almacenar los datos, sino entender e interpretar el sentido de esta información. De esta forma, Berners-Lee presenta la nueva arquitectura en que se basará la Web Semántica, no entendida como una nueva Web, sino como una extensión de la Web existente.

En mayo de 2001, Tim Berners Lee, James Hendler y Ora Lassila popularizan la idea de la Web Semántica al publicar un artículo en la revista Scientific American

titulado "The Semantic Web: a new form of Web content that is meaninful to computers will unleash a revolution of new possibilities", donde explican de forma sencilla su idea de la Web Semántica y los primeros pasos que hay que dar para llevarla a cabo.

2.1 MARCO TEÓRICO

Para desarrollar este trabajo es necesario entender todo lo que rodea el tema en cuestión que es la web semántica y los conceptos asociados de “linked data”.

He por lo tanto de introducir una definición del concepto “web semántica”, luego doy una breve descripción de sus orígenes, su estado actual y su posible desarrollo.

A continuación abordo los temas asociados a esta definición, a saber y no necesariamente en ese orden: “linked data”, RDF, SPARSQL, OWL, Ontologías y algunos otros conceptos que a lo largo del presente marco teórico iré desglosando.

2.2 LA WEB SEMÁNTICA

Existen varias definiciones acerca de lo que es la web semántica además de aquella del consorcio w3c, sin embargo la web semántica se puede definir como un conjunto de estándares y tecnologías que le permiten a una maquina (agente de software) entender el significado y contenido de la información presente en la web, estos estándares y protocolos se construyen o implementan como una capa adicional a la web actual.[CITATION Liy10 \l 9226]

La web, como está construida actualmente no está hecha para que esta información sea comprendida por las maquinas

De acuerdo a la w3c tenemos la siguiente definición acerca de la web semántica:

¿QUÉ ES LA WEB SEMÁNTICA?

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. [CITATION Wor16 \l 9226]

Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla.

Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante.

¿PARA QUÉ SIRVE?

La Web ha cambiado profundamente la forma en la que nos comunicamos, hacemos negocios y realizamos nuestro trabajo. La comunicación prácticamente con todo el mundo en cualquier momento y a bajo coste es posible hoy en día. Podemos realizar transacciones económicas a través de Internet. Tenemos acceso a millones de recursos, independientemente de nuestra situación geográfica e idioma. Todos estos factores han contribuido al éxito de la Web. Sin embargo, al mismo tiempo, estos factores que han propiciado el éxito de la Web, también han originado sus principales problemas: sobrecarga de información y heterogeneidad de fuentes de información con el consiguiente problema de interoperabilidad.

La Web Semántica ayuda a resolver estos dos importantes problemas permitiendo a los usuarios delegar tareas en software. Gracias a la semántica en la Web, el software es capaz de procesar su contenido, razonar con este, combinarlo y realizar deducciones lógicas para resolver problemas cotidianos automáticamente.

El concepto de la semántica está referido a una palabra: sintaxis, en cada lenguaje, la sintaxis se refiere a la forma como se dice algo, en donde la semántica es el significado detrás o inmerso en la frase o palabra dicha. De este modo la web semántica se puede entender como una web de significados.

Según Tim Berners Lee, la estructura de la web semántica es la siguiente:

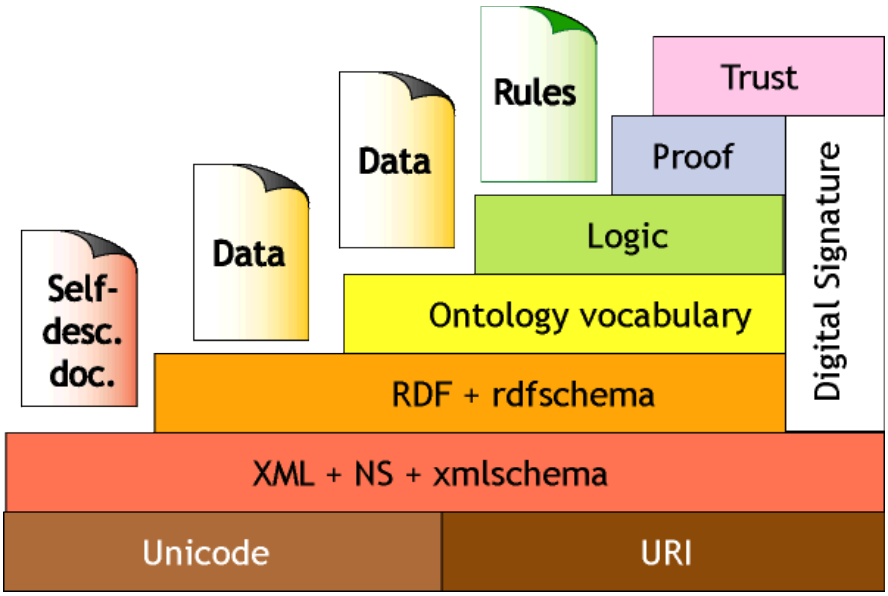


Ilustración 0-1 Semantic Web - XML2000 Architecture [CITATION Tim \l 9226]

Como se puede ver la web semántica está construida sobre los conceptos y modelos de la web actual, en donde una de las capas de la web semántica es la construcción de un vocabulario de ontologías, una superior de lógica y subsecuentes capas de prueba y verdad.

ORIGEN DEL CONCEPTO DE WEB SEMÁNTICA:

La definición anterior de la w3c sobre web semántica, fue acuñada inicialmente por el director de ese organismo Tim Berners Lee en un artículo de la revista “scientific american” en mayo de 2001, el artículo se titula “The Semantic Web”.

ESTADO ACTUAL Y POSIBLES DESARROLLOS:

La web semántica surgió con el objetivo de mejorar la calidad de la información presente en la red, sin embargo y debido a la cantidad de información que existe, la diversidad de la misma y la multiplicidad de formatos para presentar dicha información, el proceso de adaptación de estos documentos al nuevo estándar es costoso y laborioso.

En estos momentos se denomina a la web semántica la web 3.0, la gráfica o imagen siguiente ilustra la evolución de los conceptos de la web 3.0:

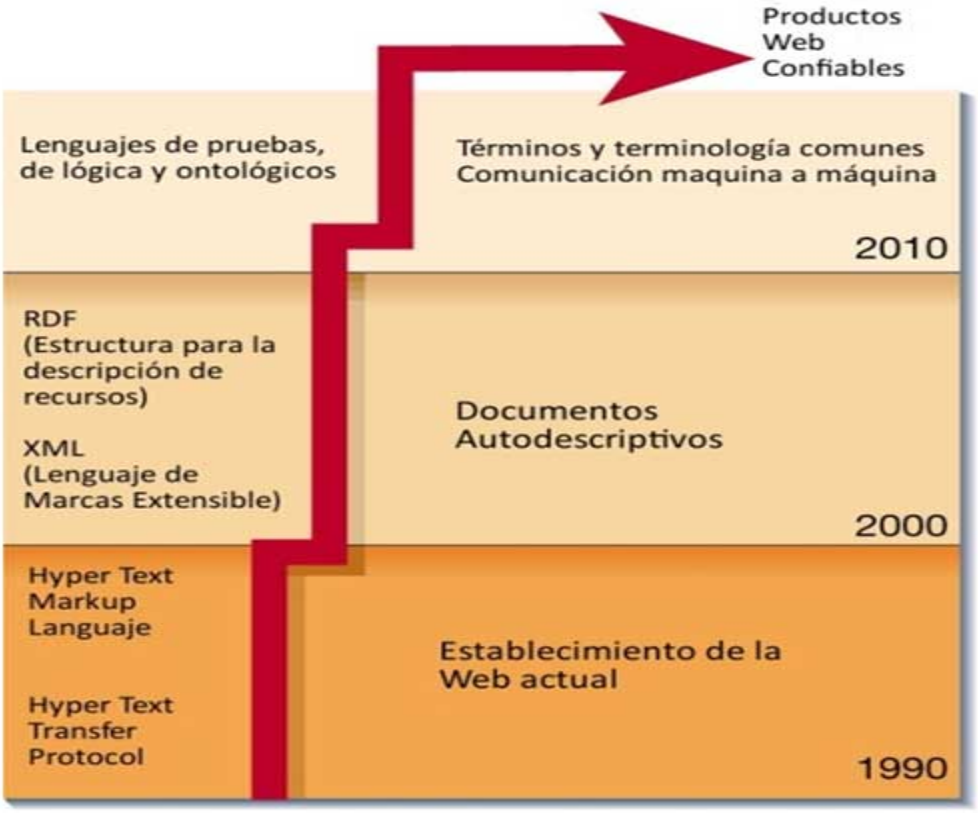


Ilustración 0-2 Capas del nuevo modelo tecnológico propuesto por Tim Berners Lee y James Hendler[CITATION Tim1 \l 9226]

El proyecto o iniciativa de web semántica, tal y como está definido, tiene bastantes similitudes con una biblioteca en donde los usuarios se ayudan con agentes inteligentes y del conocimiento de los mismos (ontologías) para acceder a la información que necesitan.[CITATION Pei \l 1033]

La implementación de toda esta arquitectura y la complexión del proyecto dependen en gran medida de varios factores:

- La facilidad y simplicidad de las herramientas para la creación de esta infraestructura
- La normalización y estandarización de procedimientos, modelos y estructuras.
- La sencillez y fiabilidad de los documentos en la web.

El proyecto a día de hoy avanza de buena manera, teniendo en cuenta que cuenta con bastante soporte y herramientas maduras, al tiempo, la calidad de la información que se entrega y las necesidades de automatización cada vez mayores están impulsando poco a poco la adopción de esta nueva web.

2.3 LINKED DATA

Linked data está en el corazón de todo lo que es la red semántica, integración a gran escala y el razonamiento de datos en la red. Linked data es una colección de conjuntos de datos interrelacionados en la red.[CITATION Wor \l 9226]

Tal y como lo dice Tim Berners Lee en su página personal: la web semántica no se trata simplemente de agregar contenido a la web, se trata de crear relaciones o enlaces, linked data permite acceder a cualquier otro conjunto de datos.

Linked data es parte esencial de la web semántica, sin linked data no hay conexiones en la web semántica.

2.4 RDF

RDF son las siglas de “Resource Definition Framework”, y fue creado a principios de 1999 como una forma estándar de codificar los metadatos. La idea de RDF es definir mecanismos para describir recursos sin hacer presunciones acerca de un dominio de aplicación particular y por lo tanto pueda usarse para describir información acerca de cualquier dominio, como resultado RDF puede ayudar a promover la interoperabilidad entre aplicaciones que intercambian datos en formato comprensible por las maquinas. De hecho actualmente RDF no solo se utiliza para codificar metadatos y recursos web, sino cualquier recurso.[CITATION Liy10 \l 9226]

RDF fue propuesto como un modelo básico para crear y procesar metadatos, el enfoque amplio de RDF para describir y ampliar aún más las capacidades del lenguaje fue definido en iteraciones subsecuentes y definido en las siguientes especificaciones:

Specification	Recommendation
RDF Primer	**10 February 2004**
RDF Test Cases	**10 February 2004**
RDF Concept	**10 February 2004**
RDF Semantics	**10 February 2004**
RDF Schema	**10 February 2004**
RDF Syntax	**10 February 2004**

Tabla 1 RDF W3C Recommendation, Feb 10 2004[CITATION Wor16 \l 9226]

Teniendo en Cuenta las anteriores especificaciones, se puede definir RDF de la siguiente manera**:**

- Es un lenguaje para representar información de recursos en la web
- En un marco de referencia para representar información en la web
- Es un lenguaje de propósito general para representar información en la red

2.5 SPARQL

Es un lenguaje asertivo con la intención de usar proposiciones expresas usando vocabularios formales precisos SPARQL

Es un lenguaje de consultas y protocolo de acceso a datos para la web semántica, fue estandarizado por el grupo de trabajo “SPARQL working group” de la W3C.

Dicho de otra manera, SPARQL es un lenguaje de consultas semántico para bases de datos, capaz de recuperar y manipular los datos almacenados en RDF, SPARQL permite que una consulta consista de patrones triples, conjunciones, disyunciones y patrones opcionales.

Existen múltiples implementaciones para distintos lenguajes de programación.

La ventaja de SPARQL es que permite ejecutar consultas contra datos pobremente estructurados o que siguen la especificación RDF.

SPARQL utiliza un sistema de palabras y frases reservadas muy similares al estándar SQL.

Ejemplo de sentencia SPARQL:

```
PREFIX ex: <http://example.com/exampleOntology#>

SELECT  ?capital ?country WHERE { ?x  ex:cityname       ?capital   ; ex:isCapitalOf    ?y. ?y
ex:countryname    ?country   ; ex:isInContinent  ex:Africa  . }
```

Tabla 2, Ejemplo código SPARQL

Con lo cual la curva de aprendizaje se reduce sustancialmente para aquellos con cierta experiencia en SQL, sin dejar de ser un lenguaje relativamente fácil de aprender.

Un completo tutorial en español se puede conseguir aquí:

Otros ejemplos de SPARQL

a. Búsqueda de científicos en España:

b. SELECT? person WHERE{?person dcterms: subject }

c. Numero de recursos geo localizados en dbpedia:

d. SELECT DISTINCT count (?res) WHERE {?res geo:lat ?v.}

e. Consulta sobre dblp, retorna los sujetos y predicados de un asunto en particular en el repositorio: SELECT DISTINCT * WHERE { ?s ?p ?o} LIMIT 10

2.6 OWL Y ONTOLOGÍAS

A pesar de ser uno de los elementos fundamentales de la web semántica, el concepto de ontología es poco comprendido; una definición de ontología es la que sigue:

“Una ontología define un conjunto común de términos que se utilizan para describir y representar un dominio, una ontología define los términos usados para describir y representar un área del conocimiento.” (Tomado de: W3C's OWL Use Cases and Requirements Documents).[CITATION Wor \l 9226]

De esta definición queda claro que: la ontología es de dominio especifico y es utilizada para representar y describir un área del conocimiento; por otra parte la ontología contiene los términos y las relaciones entre esos términos, los términos a menudo son llamados clases o conceptos y estos términos a menudo se intercambian. De estos existen relaciones entre las clases y estas pueden representarse en una estructura jerárquica en donde se dan súper clases para conceptos de alto nivel y subclases para los detalles, en todo momento las subclases tienen todos los atributos y características de las súper clases. Encima de las relaciones entre las clases, hay otro nivel de relaciones que se expresan por el uso de un grupo especial de términos, estas relaciones se llaman “propiedades”, estas a su vez describen varios atributos y características de los conceptos y pueden usarse para relacionar distintas clases entre sí. De este modo se tienen dos tipos de relaciones, aquellas definidas por la relación de clases y aquellas definidas por las propiedades de dichas clases.

Definiendo estos términos y las relaciones entre estos términos de manera clara la ontología codifica el conocimiento de un dominio de forma tal que el conocimiento puede ser entendido por una máquina.

QUE ES OWL?

Habiendo definido que son las ontologías, conociendo acerca de RDF, podemos entrar a definir que es OWL.

OWL es el acrónimo de “Web Ontology Language”. El propósito de este lenguaje es el de definir ontologías que incluyan clases, propiedades y sus relaciones dentro de una aplicación específica de dominio. Básicamente con OWL se pueden desarrollar ontologías completas, OWL está basado en RDF Schema.

A continuación una completa descripción de OWL tomado de:

El Web Ontology Language OWL está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL puede usarse para representar explícitamente el significado de términos en vocabularios y las relaciones entre aquellos términos. Esta representación de los términos y sus relaciones se denomina una . En realidad, OWL es una extensión del lenguaje y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste.
OWL posee más funcionalidades para expresar el significado y semántica que , , y , pero OWL va más allá que estos lenguajes pues ofrece la posibilidad de representar contenido de la Web interpretable por máquina. OWL es una revisión del lenguaje de ontologías web que incorpora lecciones aprendidas desde el diseño y aplicaciones de DAML+OIL.

El lenguaje OWL tiene 3 sub-lenguajes que incrementan su expresión: OWL Lite, OWL DL, y OWL Full.

La recomendación introductoria de este lenguaje está indicada para quienes pretendan obtener una primera impresión de las capacidades de OWL. La especificación ofrece una introducción a OWL para describir informalmente las características de cada uno de los sub-lenguajes de OWL. Se precisa algún conocimiento de para comprender este documento, pero no es esencial. El documento recomienda que los lectores interesados consulten para descripciones más detalladas y para ejemplos más extensos de las características de OWL. La definición formal normativa de OWL puede encontrarse en .

El Web Ontology Language OWL es, en realidad, un lenguaje de etiquetado semántico para publicar y compartir en la . OWL se ha desarrollado como una extensión del vocabulario de y deriva del lenguaje Web Ontology.

El lenguaje OWL se describe mediante un conjunto de documentos, cada uno de los cuales tienen un propósito diferente y está dedicado a un usuario diferente. A continuación se ofrece un breve mapa para la navegación a través de este conjunto de documentos:

- ofrece una introducción simple a OWL para proveer una lista de características del lenguaje con una breve descripción:
- demuestra el uso del lenguaje OWL para ofrecer un ejemplo extendido. También ofrece un glosario de la terminología usada en estos documentos:
- ofrece una descripción sistemática y resumida (pero todavía informalmente establecida) de todas las primitivas de modelado de OWL, usando la sintaxis de intercambio de RDF/XML para OWL. Este documento sirve como una guía de referencia para usuarios del lenguaje OWL:
- El documento es la normativa final y formal establecida del lenguaje:
- El documento contiene un largo conjunto de casos para el lenguaje:
- El documento contiene un conjunto de casos de uso para un lenguaje de web y compila un conjunto de requerimientos para OWL:

El motivo del desarrollo de este lenguaje ha sido la puesta en marcha de la , en realidad, una visión para el futuro de la en la cual el significado de la información será dado de forma explícita haciendo que las máquinas automaticen de forma más fácil los procesos e integren la información disponible en la Web. La Web Semántica se construirá sobre la sintaxis del lenguaje que se mejorará mediante el uso de para representar el contenido de los datos. El primer nivel sobre RDF requerido para la es un lenguaje de que pueda describir formalmente el significado de la terminología usada en los documentos web. Si las máquinas son capaces de realizar tareas de razonamiento sobre los documentos en los que se utilice una semántica que vaya más lejos que la semántica básica de RDF Schema, la irá por buen camino. El ofrece más detalles sobre ontologías, motiva la necesidad de un Lenguaje de Ontología Web en términos de seis casos de uso, y formula el diseño de objetivos, requerimientos y objetivos para OWL.

OWL ha sido diseñado para conocer las necesidades para un lenguaje de de la y es, pues, parte de las recomendaciones del W3C relacionadas con la .

- provee una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas en el significado de estos documentos.
- es un lenguaje para restringir la estructura de los documentos XML y también extiende XML con tipos de datos.
- es un modelo de datos para objetos ("recursos") y para las relaciones entre ellos, provee una semántica simple para este modelo de datos, a la vez que este modelo de datos puede ser representado en sintaxis XML.

- es un vocabulario para describir propiedades y clases de recursos RDF, con una semántica para generalización de jerarquías de aquellas propiedades y clases.

- OWL añade más vocabulario para describir propiedades y clases: entre otras, relaciones entre clases (ejemplo, inconexas), cardinalidad (ejemplo "exactamente uno"), igualdad, más ricos tipos de propiedades, características de las propiedades (por ejemplo, simetría), y clases enumeradas.

Los tres sub-lenguajes de OWL

OWL ofrece tres sub-lenguajes de expresión incremental diseñados para ser usados por comunidades específicas de desarrolladores y usuarios según el nivel de expresividad que precisen éstos.

- OWL Lite da soporte a aquellos que primordialmente necesitan una jerárquica y restricciones simples. Por ejemplo, soporta restricciones cardinales, pero solamente permite valores cardinales de 0 ó 1. Así pues, es más simple proveer herramientas de soporte para OWL Lite. OWL Lite ofrece una rápida ruta de migración para y otras taxonomías. En resumen, OWL Lite tiene una más baja complejidad formal que OWL DL.

- OWL DL da soporte a aquellos que quieren la máxima expresividad mientras conservan completamente la computacionalidad (todas las conclusiones son garantizadas para ser computables) y resolubilidad (todas las computaciones terminarán en tiempo finito). OWL DL incluye todos los constructos del lenguaje OWL, pero pueden usarse solamente bajo ciertas restricciones (por ejemplo, mientras una clase puede usarse por una subclase de muchas clases, una clase no puede ser una instancia de otra clase). OWL DL se denomina así debido a su correspondencia con , un campo de investigación que han estudiado los lógicos para la fundación formal de OWL.

- OWL Full da soporte a que requieren el máximo de expresividad y la libertad sintáctica de RDF sin garantías computacionales. Por ejemplo, en OWL Full una clase puede ser tratada simultáneamente como una colección de individuos y como un individuo por derecho propio. OWL Full permite a una aumentar el significado del vocabulario predefinido (RDF ó OWL). Es poco probable que algún software racional pueda soportar por completo el razonamiento para cada característica de OWL Full.

Cada uno de estos sub-lenguajes es una extensión de su predecesor más simple, en los que ambos pueden ser expresados legalmente y en los que pueden ser válidamente concluidos. El conjunto siguiente de relaciones es correcto, sin embargo, no sus inversas.

- Cada ontología legal OWL Lite es una ontología legal OWL DL.
- Cada ontología legal OWL DL es una ontología legal OWL Full.
- Cada conclusión válida OWL Lite es una conclusión válida OWL DL.
- Cada conclusión válida OWL DL es una conclusión válida OWL Full.

Los desarrolladores de adoptarán el sub-lenguaje OWL que consideren que mejor responde a sus necesidades. La elección entre OWL Lite y OWL DL dependerá de la mayor o menor extensión que los requieran de los constructos más expresivos provistos por OWL DL. La elección entre OWL DL y OWL Full dependerá, principalmente, de la extensión que los requieran de las posibilidades del metamodelo de RDF Schema (por ejemplo, definir clases de clases, o adjuntar propiedades a clases). Cuando se usa OWL Full en comparación a OWL DL, la razón es menos predictible porque las implementaciones completas de OWL Full no existen actualmente.

OWL Full puede verse como una extensión de , mientras que OWL Lite y OWL DL pueden verse como extensiones de una forma restrictiva de . Cada documento OWL (Lite, DL, Full) es un documento , y cada documento es un documento OWL Full, pero solamente los documentos serán un documento OWL Lite u OWL DL legales. A causa de esto, los tienen que tomar ciertas precauciones si quieren migrar un documento hacia un documento OWL. Cuando la expresividad de OWL DL u OWL Lite se considera apropiada, deben tomarse algunas precauciones para asegurarse de que el documento original cumpla con las restricciones adicionales impuestas por OWL DL y OWL Lite. Entre otras, cada URI que se usa como un nombre de clase debe ser explícitamente asertado para ser del tipo owl:Class (y de igual forma para las propiedades), cada individuo debe ser asertado como perteneciente a una clase (también si solamente es owl:Thing), los URIs usados por las clases, propiedades e individuos deben ser mutuamente inconexos. Los detalles de estas y otras restricciones en OWL DL y OWL Lite se explican en el appendix E de la OWL Reference.

Para hacernos una idea de en qué consiste el lenguaje OWL, nos centraremos en OWL-Lite, puesto que, aunque OWL Lite usa solamente algunas características de los lenguajes OWL y tiene más limitaciones en el uso de las características que OWL DL o OWL Full, con este sub-lenguaje ya se pueden establecer relaciones jerárquicas entre los conceptos que componen una , a la vez que aporta una

menor complejidad formal que sus hermanos mayores. De hecho OWL Lite proporciona una forma rápida de migrar y otras taxonomías al ámbito de la .

Por ejemplo, las clases OWL Lite pueden ser solamente definidas en términos de las llamadas superclases (las superclases no pueden ser expresiones arbitrarias), y solamente pueden usarse ciertas clases de restricciones. La equivalencia entre clases y relaciones entre subclases con las clases solamente están permitidas entre clases de nombres, y no entre expresiones de clases arbitrarias. De igual forma, las restricciones en OWL Lite usan solamente clases de nombres. OWL Lite también tiene una notación limitada de cardinalidad -solamente están permitidos explícitamente para ser establecidos 0 ó 1.
He aquí, pues, algunos ejemplos de los conceptos principales de OWL Lite y sus definiciones:

Características RDF Schema de OWL Lite:

- Class: Una clase define un grupo de individuos que permanecen juntos porque comparten las mismas propiedades. Por ejemplo, Juana y Pedro son ambos miembros de la clase Persona. Las clases pueden ser organizadas en una jerarquía especial, usando subClassOf. Hay una construcción de clase más general llamada Thing que es la clase de todos los individuos y una superclase de todas las clases OWL. Hay también una construcción más específica de clase llamada Nothing que es la clase que no tiene instancias y una subclases de todas las clases OWL.

- rdfs:subClassOf: Las jerarquías de clase deben crearse para hacer una o más declaraciones de que una clase es un subclase de otra clase. Por ejemplo, la clase Persona podría ser establecida para ser una subclase de la clase Mamífero. Desde aquí un razonador puede deducir que si un individuo es una Persona, entonces es también un Mamífero.

- rdf:Property: Las propiedades pueden usarse para state relaciones entre individuos o desde individuos a valores de datos. Ejemplos de propiedades son hasChild, hasRelative, hasSibling, y hasAge. Las tres primeras pueden usarse para relacionar una instancia de una clase Person a otra instancia de la clase Person (y son entonces ocurrencias de ObjectProperty), y el último (hasAge) puede usarse para relacionar una instancia de la clase Persona a una instancia del tipo de datos Entero (y es entonces una ocurrencia de DatatypeProperty). Ambos owl:ObjectProperty y owl:DatatypeProperty son subclasses de la clase RDF rdf:Property.

- rdfs:subPropertyOf: Las jerarquías de propiedades deben crearse para hacer una o más declaraciones que una propiedad es una subpropiedad de una o más other propiedades. Por ejemplo, hasSibling debe ser establecida para ser una subpropiedad de hasRelative. Desde aquí un razonador puede deducir que si un individuo está relacionado a otro por la propiedad hasSibling, entonces éste está también relacionado a otro por la propiedad hasRelative.

- rdfs:domain: Un dominio de una propiedad limita los individuos a los que se aplica la propiedad. Si una propiedad relaciona un individuo a otro individuo, y la propiedad tiene una clase como uno de sus dominios, entonces el individuo debe pertenecer a la clase. Por ejemplo, la propiedad hasChild debe ser establecida para tener el dominio de Mamífero. Desde aquí un razonador puede deducir que si Pedro hasChild Ana, entonces Pedro debe ser un Mamífero. Note que rdfs:domain se denomina una restricción global desde que la restricción está establecida sobre la propiedad y no justamente sobre la propiedad cuando esta está asociada con una clase particular.

- rdfs:range: El rango de una propiedad limita los individuos que la propiedad debe tener como su valor. Si una propiedad relaciona un individuo a otro individuo, y la propiedad tiene una clase como su rango, entonces el otro individuo debe pertenecer a la clase del rango. Por ejemplo, la propiedad hasChild debe ser establecida para tener el rango de Mamífero. Desde aquí un razonador puede deducir que si Luisa está relacionado con Juana por la propiedad hasChild, (por ejemplo, Juana es la hija de Luisa), entonces Juana es un Mamífero. El rango es también una restricción global como lo era el dominio.

- Individual : Los individuos son instancias de clases, y las propiedades deben usarse para relacionar un individuo con otro. Por ejemplo, un individuo llamado Juana debe ser descrito como una instancia de la clase Persona y la propiedad hasEmployer debe ser usada para relacionar el individuo Juana al individuo UniversidadComplutense.

Características que tienen que ver con igualdad o desigualdad en OWL Lite:

- equivalentClass : Dos clases pueden ser establecidas para ser equivalentes. Las clases equivalentes tienen las mismas instancias. La

igualdad puede usarse para crear clases de sinónimos. Por ejemplo, Coche puede ser establecido para ser equivalentClass a Automóvil. Desde aquí, se puede deducir que algún individuo que tiene una instancia para Coche es también una instancia de Automóvil y viceversa.

- equivalentProperty: Dos propiedades pueden ser establecidas como equivalentes. Las propiedades de equivalencia relacionan un individuo al mismo conjunto de otros individuos. La igualdad debe usarse para crear propiedades de sinónimos. Por ejemplo, hasLeader debe establecerse para la equivalentProperty a hasHead. Desde aquí se puede deducir que si X está relacionado con Y por la propiedad hasLeader, X está también relacionado a Y por la propiedad hasHead y viceversa. Un razonador puede también deducir que hasLeader es una subpropiedad de hasHead y hasHead es una subpropiedad de hasLeader.

- sameAs: Dos individuos deben ser establecidos como lo mismo. Estos constructos deben ser usados para crear un número de nombres diferentes para referirse al mismo individuo. Por ejemplo, el individuo Juana puede ser establecido al mismo individuo como JuanaPérez.

- differentFrom: Un individuo puede ser establecido como diferente de otros individuos. Por ejemplo, el individuo Pedro puede ser establecido como diferente de los individuos Juana y Mariano. Entonces, si los individuos Pedro y Juana son ambos valores para una propiedad que es establecida para ser funcional (entonces la propiedad tiene más de un valor), existe una contradicción. Establecer explícitamente que los individuos son diferentes puede ser importante cuando usamos lenguajes tales como OWL (y RDF) que no asumen que los individuos tienen uno y solamente un nombres. Por ejemplo, con información no adicional, un razonador no deducirá que Pedro y Juana se refieren a individuos distintos.

- AllDifferent: Un número de individuos puede ser establecido para ser mutuamente distintos en una declaración AllDifferent. Por ejemplo, Pedro, Juana, y Mariano podrían ser establecidos como distintos usando el constructo AllDifferent. Al contrario que la declaración differentFrom, eta podría también imponer que Mariano y Juana son distintos (pero no que Pedro es distinto de Juana y Pedro es distinto de Mariano. El constructo AllDifferent se usa particularmente cuando hay un conjunto de distintos objetos y cuando los modeladores están interesados en imponer los nombres únicos supuestos dentro de aquel conjunto de objetos. Se usa

en conjunción con distinctMembers para establecer que todos los miembros de una lista son distintos y pares inconexos.

Identificadores especiales en OWL Lite que se usan para proveer información concerniente a las propiedades y valores.

- inverseOf: Una propiedad puede ser establecida para ser la inversa de otra propiedad. Si se establece la propiedad P1 como inversa de la propiedad P2, entonces si X se relaciona a Y por la propiedad P2,Y estará relacionada a X por la propiedad P1. Por ejemplo, si hasChild es la inversa de hasParent y JuanahasParent Luisa, entonces, un razonador deduce que Luisa hasChild Juana.

- TransitiveProperty: Las propiedades pueden establecerse como transitivas. Si una propiedad es transitiva, entonces si el par (x,y) es una instancia de la propiedad transitiva P, y el par (y,z) es una instancia de P, entonces el par (x,z) es también una instancia de P. Por ejemplo, si ancestro se establece para ser transitiva, y si Sara es un ancestor de Luisa (por ejemplo, (Sara,Luisa) es una instancia de la propiedad ancestro) y Luisa es un ancestro de Juana (por ejemplo, (Luisa,Juana) es una instancia de la propiedad ancestro), entonces un razonador puede deducir que Sara es un ancestro de Juana (por ejemplo, (Sara,Juana) es una instancia de la propiedad ancestro). OWL Lite (y OWL DL) imponen la condición de que las propiedades transitivas (y sus superpropiedades) no pueden tener una restricción maxCardinality 1.

- SymmetricProperty: Las propiedades pueden establecerse como simétricas. Si una propiedad es simétrica, entonces si el par (x,y) es una instancia de la propiedad simétrica P, entonces el par (y,x) es también una instancia de P. Por ejemplo, amigo puede ser establecido como una propiedad simétrica. Así, un razonador al que se da que Pedro es amigo de Juan, puede deducir que Juan es amigo de Pedro.

- FunctionalProperty : Las propiedades pueden ser establecidas para tener un valor único. Si una propiedad es una FunctionalProperty, entonces, no tiene mas que un valor para cada individuo (no debe tener valores para un individuo). Esta característica se denomina como una propiedad única. FunctionalPropertyes la forma abreviada para establecer que la cardinalidad mínima de la propiedad es cero y la cardinalidad máxima es 1. Por ejemplo, hasPrimaryEmployer puede ser establecida para ser

una FunctionalProperty. Desde aquí, un razonador puede deducir que no individual puede tener más que un empleado primary. Esto no implica, sin embargo, que cada Persona deba tener al menos un empleado.

- InverseFunctionalProperty: Las propiedades pueden ser establecidas para ser funcionalmente inversas. Si una propiedad es funcionalmente inversa entonces la inversa de la propiedad es funcional. Entonces la inversa de la propiedad tiene más de un valor para cada individuo. Esta característica también se denomina como una propiedad inambigua. Por ejemplo, hasESSocialSecurityNumber (un único identificador para residentes en España) debe establecerse para ser funcional inverso (o inambiguo). La inversa de esta propiedad (que debe ser referida como isTheSocialSecurityNumberFor) tiene más de un valor para cada individuo en la clase de los números de la Seguridad Social. Entonces un número de Seguridad Social de una persona es el único valor para su propiedad isTheSocialSecurityNumberFor. Desde aquí un razonador puede deducir que dos instancias individuales de Persona no tienen idéntico número de Seguridad Social. También un razonador puede deducir que si dos instancias de Persona tienen el mismo número de Seguridad social, entonces aquellas dos instancias se refieren al mismo individuo.

OWL usa los mecanismos para valores de datos. OWL Lite soporta notaciones de inclusión de y relaciones e información adjunta a . También permite notaciones en clases, propiedades, individuos y cabeceras de . El uso de estas notaciones está sujeto a ciertas restricciones. , por ejemplo, ya posee un pequeño vocabulario para describir información sobre la versión de un documento, pero OWL extiende significativamente este vocabulario.
OWL DL y OWL Full usan el mismo vocabulario. Veamos también un ejemplo del vocabulario OWL
DL y OWL Full que extiende las construcciones de OWL Lite:

- oneOf: (clases enumeradas): Las clases pueden describirse por enumeración de individuos que componen la clase. Los miembros de la clases son exactamente el conjunto de individuos enumerados, ni más ni menos. Por ejemplo, la clase de DíasdelaSemana puede describirse por una simple enumeración de individuos Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo. A partir de aquí, un razonador puede deducir el máximo cardinal (7) de alguna propiedad que tiene el DíasdelaSemana como su restricción allValuesFrom.

- hasValue: (valores de propiedad): Una propiedad puede ser requerida para tener un cierto individuo como un valor (también algunas veces para

referirse a un valor de propiedad). Por ejemplo, instancias de la clase de CiudadanosAlemanes pueden ser caracterizados como aquella gente que tiene theNetherlands como un valor de su nacionalidad. (El valor nacionalidad, es una instancia de la clase de Nacionalidades).

- disjointWith: Las clases pueden ser establecidas para ser distintas de otras. Por ejemplo, Hombre y Mujer pueden ser establecidas para ser clases distintas. Desde la declaración de esta disjointWith, un razonador puede deducir una inconsistencia cuando un individuo es establecido para ser una instancia de ambos y de igual forma un razonador pude deducir que si A es una instancia de Hombre, entonces A no es una instancia de Mujer.

- unionOf, complementOf, intersectionOf (combinaciones booleanas): OWL DL y OWL Full permiten combinaciones booleanas arbitrarias de clases y restricciones: unionOf, complementOf, e intersectionOf. Por ejemplo, usando unionOf, podemos establecer que una clase contiene cosas que son eitherCiudadanosEspañoles ó CiudadanosAlemanes. Usando complementOf, podríamos establecer que los niños no son CiudadanosMayoresdeEdad. (por ejemplo, la clase Niños es una subclase del complemento de CiudadanosMayoresdeEdad). Ciudadanos de la Unión Europea podría describirse como la unión de los ciudadanos de todos los estados miembros.

- minCardinality, maxCardinality, cardinality (cardinalidad plena): Mientras que en OWL Lite, la cardinalidad está restringida al máximo, más exactamente a 1 ó 0, OWL Full permite declaraciones cardinales para enteros no negativos. Por ejemplo, la clase de DRNKs ("Doble renta, No Kids") podría restringir la cardinalidad de la propiedad hasIncome a una mínima cardinalidad de dos (mientras la propiedad hasChild podría tener restringida la cardinalidad a 0).

- complex classes : En muchos constructos, OWL Lite restringe la sintaxis a nombres de clase únicos (por ejemplo, en declaraciones subClassOf ó equivalentClass). OWL Full extiende esta restricción para permitir descripciones de clases complejas, que constan de clases enumeradas, restricciones de propiedades, y combinaciones booleanas. También, OWL Full permite que las clases sean usadas como instancias (y OWL DL y OWL Lite no).

Existe un servicio de validación de los distintos tipos de OWL (Lite, DL y Full) que también permite mostrar los constructos utilizados o la forma abstracta, desarrollado dentro del proyecto :

Como ejemplo de una realizada en OWL y para ver la sintaxis de dicho lenguaje, basado en sobre , podemos consultar la Ontología sobre Derechos de Autor desarrollada por Distributed Multimedia Applications Group (DMAG) de la Universidad Pompeu Fabra . En podemos visualizar dicha : IPROnto Intellectual Property Rights Ontology en su versión actual OWL y en la antigua versión para DAML-OIL. [CITATION Hip \l 9226]

2.7 QUE ES APACHE JENA

es un framework Java para construir aplicaciones basadas en ontologías.
Jena se desarrolló en HP Labs en el 2000, en 2009 HP cedió el proyecto a la fundación Apache que decidió adoptarlo en noviembre de 2010. [CITATION Lui12 \l 9226]
Su Arquitectura incluye:

- API para trabajar (leer, procesar, escribir) ontologías RDF y OWL
- Motor de inferencia para razonar sobre ontologías RDF y OWL
- Estrategias de almacenamiento flexible para almacenar tripletas RDF en memoria o fichero
- Motor de queries compatible con especificación SPARQL

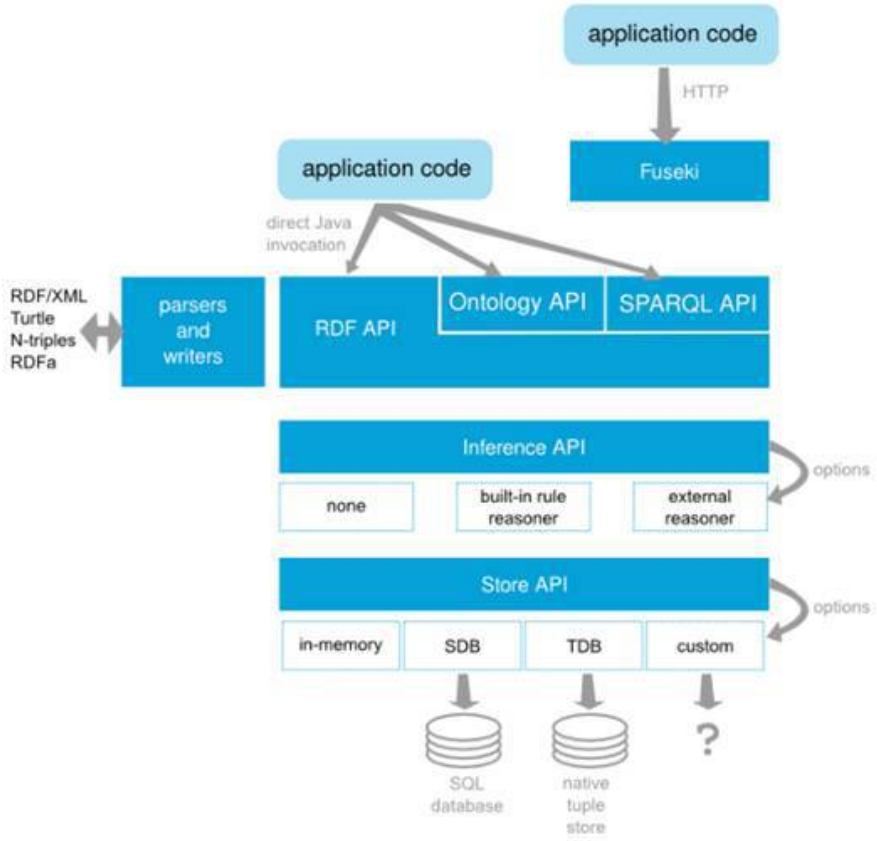


Ilustración 0-3, Que es apache JENA[CITATION Lui12 \l 9226]

Asociados con Jena hay un gran número de proyectos interesantes:
· : generador de JavaBeans desde OWL
· : Jena para dispositivos móviles
· : interfaz para trabajar con queries SPARQL
· : permite trabajar con bases de datos relacionales no RDF con grafos Jena RDF
· : colección de utilidades y ejemplos para usar MapReduce, Pig yHBase para procesar datos en formato RDF
· : añade predicados geoespaciales y capacidades de razonamiento a Jena.

```
String queryString = "PREFIX : <http://example.org/>"
     + "PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>\n "
     + "PREFIX ext: <java:org.geospatialweb.arqext.>\n\n "
     + "SELECT ?subject ?c"
     + "WHERE { ?c a :City . ?subject ext:nearby( ?c 20 )}";
Query query = QueryFactory.create(queryString);
QueryExecution qexec = QueryExecutionFactory.create(query, m);
Geo.setContext(qexec, i);
try {
   ResultSet results = qexec.execSelect();
   for (; results.hasNext();) {...
```

Tabla 3, Ejemplo de codigo en JENA. [CITATION Lui12 \l 9226]

2.8 DESARROLLO DEL TRABAJO

BORRADORES Y SOLUCIÓN INICIAL PLANTEADA.

La solución planteada inicialmente en las sucesivas entregas y borradores iniciales fue la siguiente:

No hay más caso de uso para la API que el de leer una URI proporcionada por el usuario y desglosarla en sus distintos componentes, esto queda expresado en el diagrama a continuación>

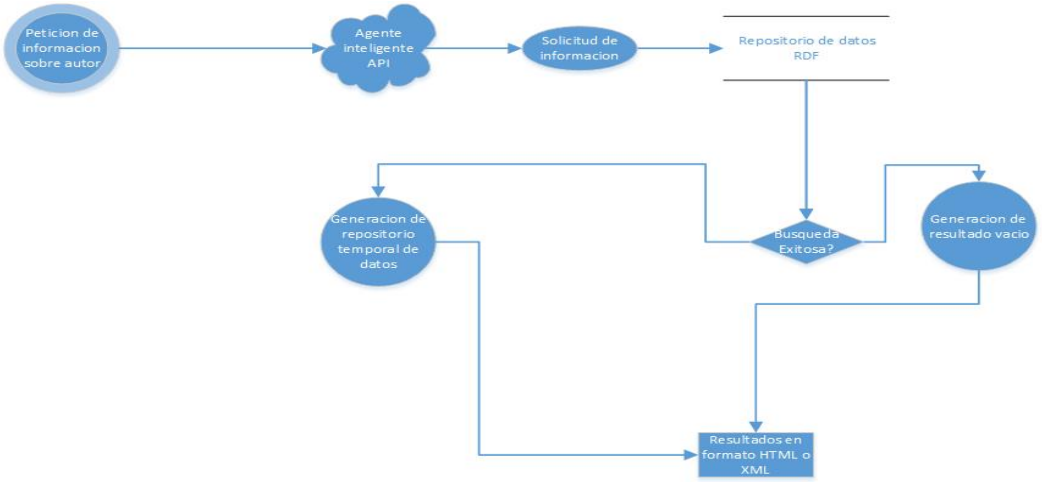


Ilustración 0-4, Diagrama casos de uso o procesos iniciales.[CITATION Bre \l 9226]

Los casos de uso considerados en este desarrollo solo involucran un agente inteligente que se encarga de buscar y gestionar la información solicitada para un autor aun repositorio de datos RDF, si dicho repositorio tiene los datos o información en el formato y cantidad adecuados, entonces se genera un reporte o informe de resultados ya sea en html o xml.

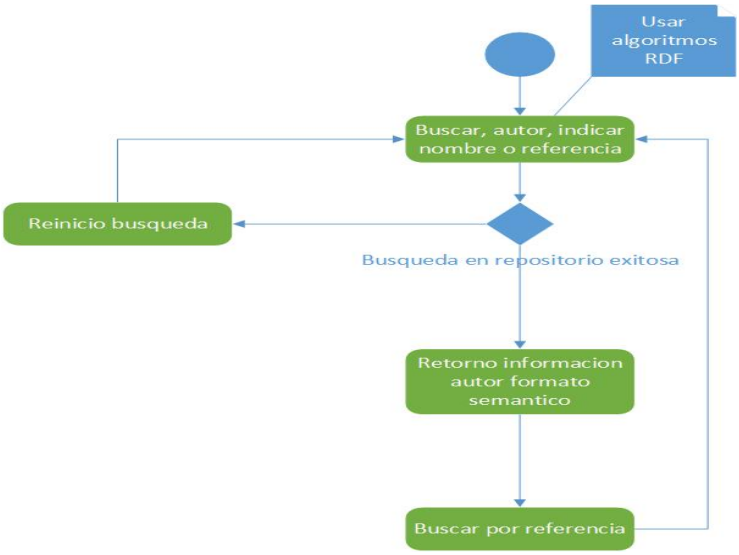


Ilustración 0-5, Diagrama de procesos.[CITATION Bre \l 9226]

En este diagrama es fácilmente visible la simplicidad el modelo de la API, solo se trata de una función de búsqueda con dos puntos de retornos a la búsqueda inicial, sea que se encuentre o no la información requerida, en todo caso la decisión de volver o no, al punto inicial está en manos del usuario.

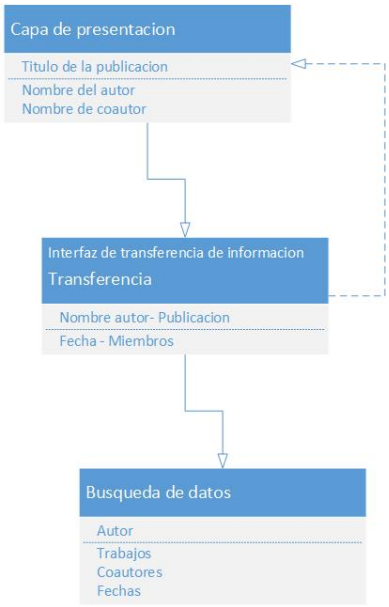


Ilustración 0-6, Diagrama de clases. [CITATION Bre \l 9226]

Se ha planteado de forma inicial una API de tres clases, la de presentación que gestionaría la estructura y formado de visualización de los resultados. La de transferencia de información, que gestionaría el paso de los datos con un formato y estructura a otro formato y estructura a la clase o capa de presentación. Finalizando con la clase de búsqueda y gestión de los datos, en esta clase se gestionan las tareas de búsqueda para luego ser enviadas a la transferencia de

información. Las dos primeras clases heredan atributos de la clase de búsqueda de datos.

ANALISIS Y COMPARATIVA DE LOS REPOSITORIOS VISITADOS

Durante el proceso de análisis y estudio de la web semántica y “linked data”, se visitaron varios repositorios y se hicieron varias consultas a los mencionados repositorios.

Los repositorios RDF visitados fueron:

DBLP: repositorio dedicado al ámbito de la informática, su web RDF es: , su “endpoint” es: ; funciona el “endpoint” sobre un servidor D2R.

Al principio fue algo confuso entender como operaba el sitio dada la gran cantidad de información que genera, sin embargo no pasa mucho tiempo antes de determinar qué significa cada sección.

La estructura del sitio web: punto de partida del sitio con información enlazada es la siguiente:

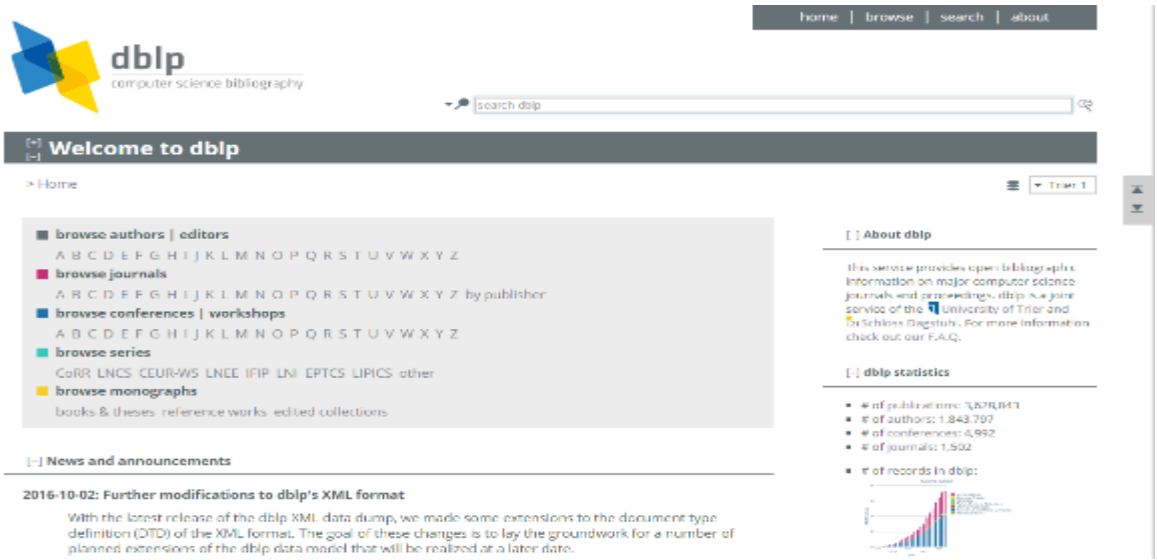


Ilustración 0-7, Pagina inicial DBLP,[CITATION Bre \l 9226]

- Sección dedicada a explorar el sitio por autores, journals, conferencias, series, etc.
- Sección con noticias.
- Menú de estadísticas y un acerca de.
- Sección de búsqueda, motor de búsqueda en el sitio.
- Menú principal.

Lo primero que se hizo fue buscar un autor al azar usando la opción de exploración, por ejemplo:

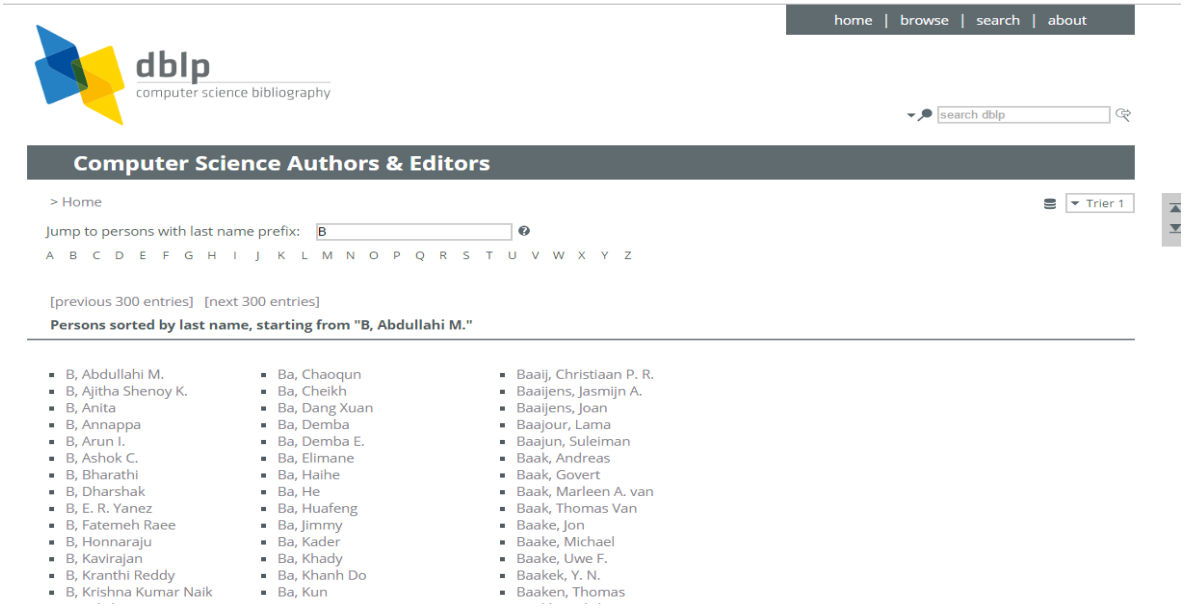


Ilustración 0-8, Listado autores, [CITATION Bre \l 9226]

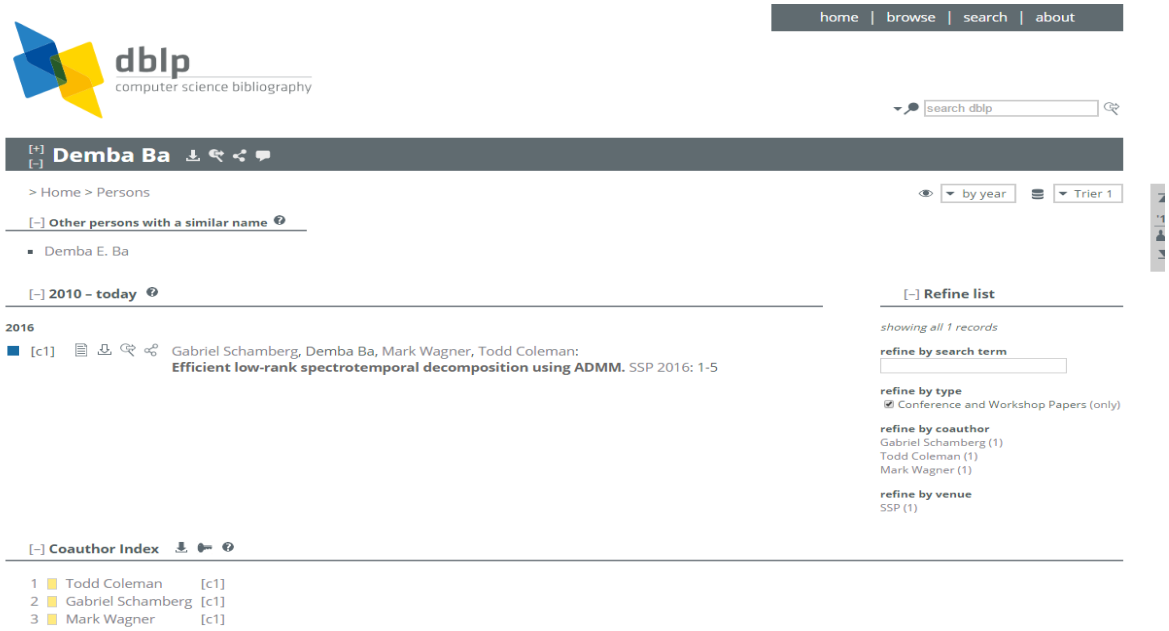


Ilustración 0-9, búsqueda por autor, [CITATION Bre \l 9226]

En esta última imagen ilustro la página de resultados por autor, nótese las opciones que ofrece, a la izquierda en el menú lateral es posible ver opciones para refinar la búsqueda, así como opciones para filtrar por coautor y filtros por año.

El punto que nos interesa de esta página es la información RDF, es decir dónde están los enlaces y como obtener esos enlaces para el motor de búsqueda.

Resulta que después de revisar las distintas opciones que ofrece esta página de resultados, la información en el formato deseado se encuentra dentro de este menú emergente:

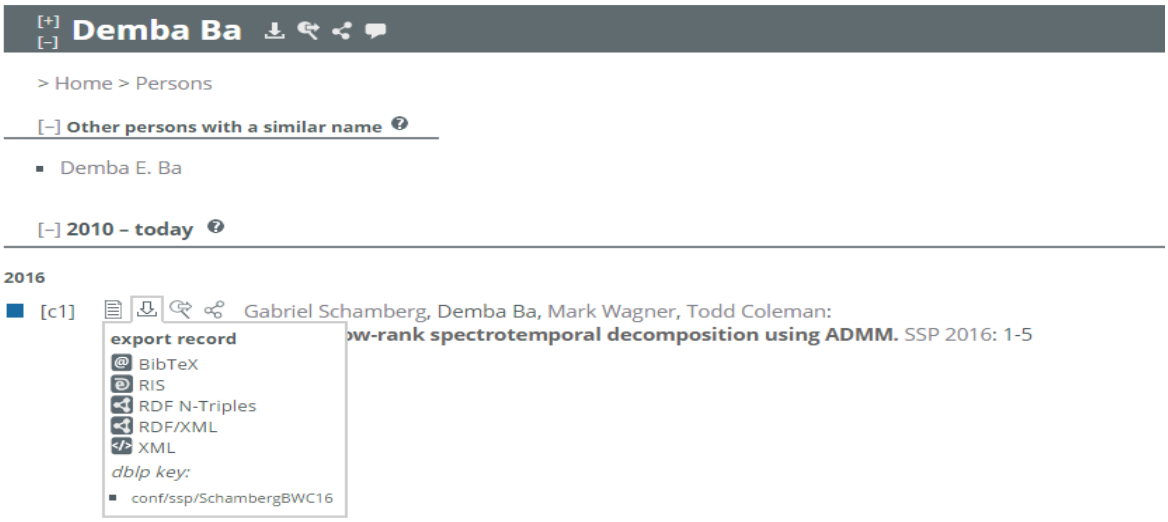


Ilustración 0-10, resultados por autor, [CITATION Bre \l 9226]

Como se puede ver en la última imagen, esta página ofrece el formato RDF como una opción de exportación, de aquí se puede tomar el enlace indicado y pasarlo a la clase analizadora de estructura RDF a fin de determinar si contiene la información que se requiere.

En la imagen se puede observar cómo se captura la dirección IP con la referencia a la URI con datos RDF.

Luego esta dirección se carga en el analizador de formato RDF y su resultado determina si este repositorio contiene los datos requeridos.

Cabe aclarar que hasta ahora no se ha utilizado el “front-end” sparql para hacer las consultas, estamos de momento usando JENA y una serie de direcciones URI entregadas por el motor de búsqueda del repositorio.

Capturando la URI y enviándola a un programa en JAVA me permite saber la estructura y las distintas propiedades del repositorio.

Para la dirección en mención, la información requerida es presentada en consola, la imagen muestra dichos resultados:

```
1  import org.apache.jena.rdf.model.Model;
2
3
4
5  public class ReadRDFModel
6     {
7
8     public static final String STRING ="http://dblp.uni-trier.de/rec/rdf/conf/ssp/SchambergBWC16.rdf";
9
10    public static void main(String[] args)
11    {
12
13        Model model = ModelFactory.createDefaultModel();
14        model.read(STRING, "N3");
15        model.write(System.out,"N3");
16    }
17   }
18
```

Console

```
<terminated> ReadRDFModel [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (18/01/2017, 10:31:09 a.m.)
log4j:WARN No appenders could be found for logger (Jena).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix bibtex: <http://data.bibbase.org/ontology/#> .
@prefix dblp:  <http://dblp.uni-trier.de/rdf/schema-2016-11-11#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1> .

<http://dblp.uni-trier.de/rec/conf/ssp/SchambergBWC16>
        a                       dblp:Publication ;
        dblp:authoredBy         <http://dblp.org/pers/s/Schamberg:Gabriel> , <http://dblp.org/pers/b/Ba:Demba> , <http://dblp.org/pers/c/Coleman:Todd> , <http://dblp.org/pers/w/Wagner:Mark> ;
        dblp:bibtexType         bibtex:Inproceedings ;
        dblp:pageNumbers        "1-5" ;
        dblp:primaryElectronicEdition  <http://dx.doi.org/10.1109/SSP.2016.7551797> ;
        dblp:publicationType    dblp:Inproceedings ;
        dblp:publishedAsPartOf  <http://dblp.org/rec/conf/ssp/2016> ;
        dblp:publishedInBook    "SSP" ;
        dblp:title              "Efficient low-rank spectrotemporal decomposition using ADMM." ;
        dblp:yearOfPublication  "2016" ;
        owl:sameAs              <http://dblp.org/rec/conf/ssp/SchambergBWC16> , <http://dx.doi.org/10.1109/SSP.2016.7551797> .

<http://dblp.uni-trier.de/rec/conf/ssp/SchambergBWC16.rdf>
        rdfs:label      "provenance information for RDF data of dblp record 'conf/ssp/SchambergBWC16'" ;
        dcterms:creator   <http://dblp.org> ;
        dcterms:isPartOf  <http://dblp.org/rdf/dblp.rdf> ;
        dcterms:license   <http://www.opendatacommons.org/licenses/by/> .
```

Ilustración 0-11, resultado por consola, [CITATION Bre \l 9226]

Una vez obtenido estos resultados, procedemos con la consulta SPARQL en el "front-end" del repositorio usando el mismo autor y los resultados son los mismos.

Con esta consulta Sparql, valido por segunda vez la información obtenida y al mismo tiempo se hace uso del lenguaje de consultas para conocer su función.

Para este caso se inicia con una función "Describe" que muestra la información que requerimos de forma más completa.

Snorql: Exploring http://dblp.l3s.de/d2r/sparql

SPARQL:

```
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
DESCRIBE ?author
WHERE { ?author foaf:name "Demba E. Ba" }
```

Results: Browse Go! Reset

SPARQL results:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:map="file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#"
    xmlns:swrc="http://swrc.ontoware.org/ontology#"
    xmlns:d2r="http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <rdf:Description rdf:about="http://dblp.l3s.de/d2r/resource/publications/conf/icmcs/BaFZ07">
    <foaf:maker>
      <foaf:Agent rdf:about="http://dblp.l3s.de/d2r/resource/authors/Demba_E._Ba">
        <rdfs:seeAlso>
          <rdf:Description rdf:about="http://www.bibsonomy.org/uri/author/Demba+E.+Ba">
            <rdfs:seeAlso rdf:resource="http://dblp.l3s.de/d2r/sparql?query=DESCRIBE+%3Chttp%3A%2F%2Fwww.bibsonomy.org%2Furi%2Fauthor%2FDemba%2BE.%2BBa%3E"/>
          </rdf:Description>
        </rdfs:seeAlso>
        <rdfs:seeAlso>
          <rdf:Description rdf:about="http://dblp.l3s.de/Authors/Demba+E.+Ba">
            <rdfs:seeAlso rdf:resource="http://dblp.l3s.de/d2r/sparql?query=DESCRIBE+%3Chttp%3A%2F%2Fdblp.l3s.de%2FAuthors%2FDemba%2BE.%2BBa%3E"/>
          </rdf:Description>
        </rdfs:seeAlso>
        <rdfs:label>Demba E. Ba</rdfs:label>
        <foaf:name>Demba E. Ba</foaf:name>
      </foaf:Agent>
    </foaf:maker>
    <dc:creator rdf:resource="http://dblp.l3s.de/d2r/resource/authors/Demba_E._Ba"/>
  </rdf:Description>
```

Ilustración 0-12, Búsqueda SPARQL en DBLP [CITATION Bre \l 9226]

Usamos ahora la función “SELECT” para obtener el compendio de los enlaces que referencian al autor:

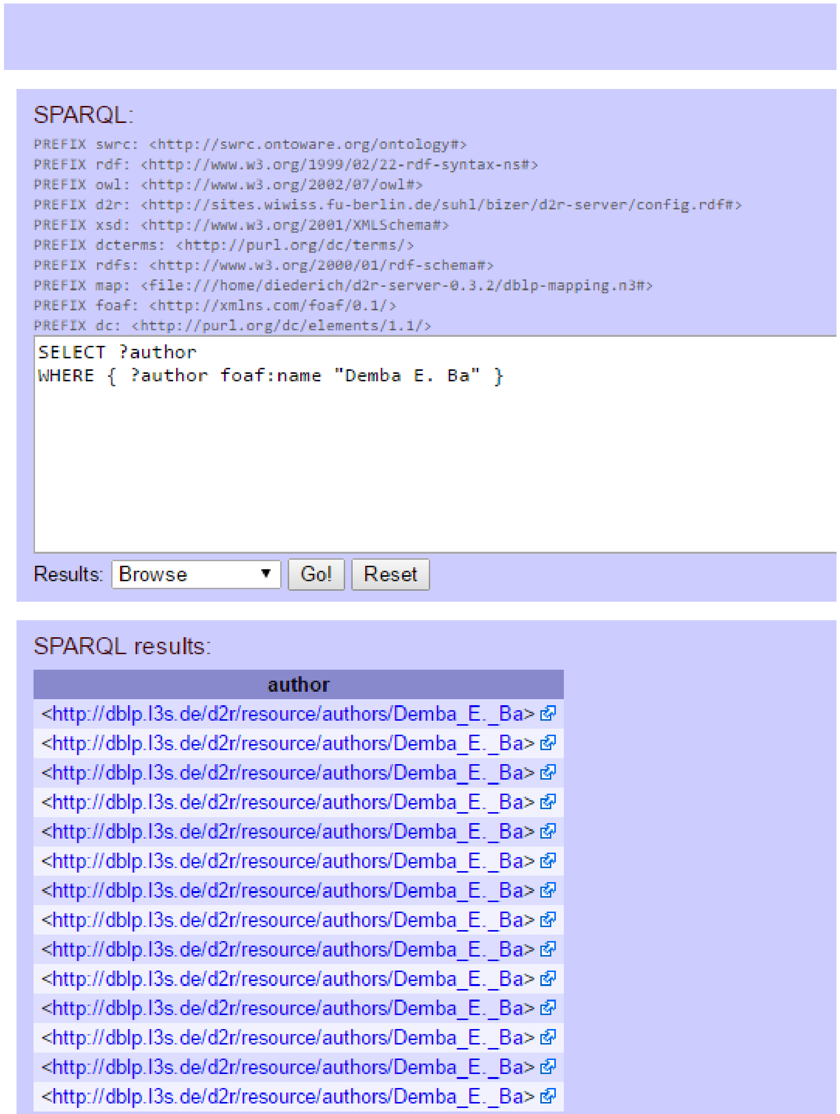


Ilustración 0-13, Busqueda en front end SPARQL de DBLP [CITATION Bre \l 9226]

El segundo repositorio visitado es ResearchGate:

En este caso el repositorio exigía un registro previo para poder usar la plataforma, a la fecha aún no he recibido el correo de verificación de registro con lo cual este repositorio es descartado.

El siguiente apartado es Bibsonomy.

Bibsonomy, similar a DBLP es un repositorio principalmente académico, su interfaz y la descripción de su página de inicio es clara en este aspecto:

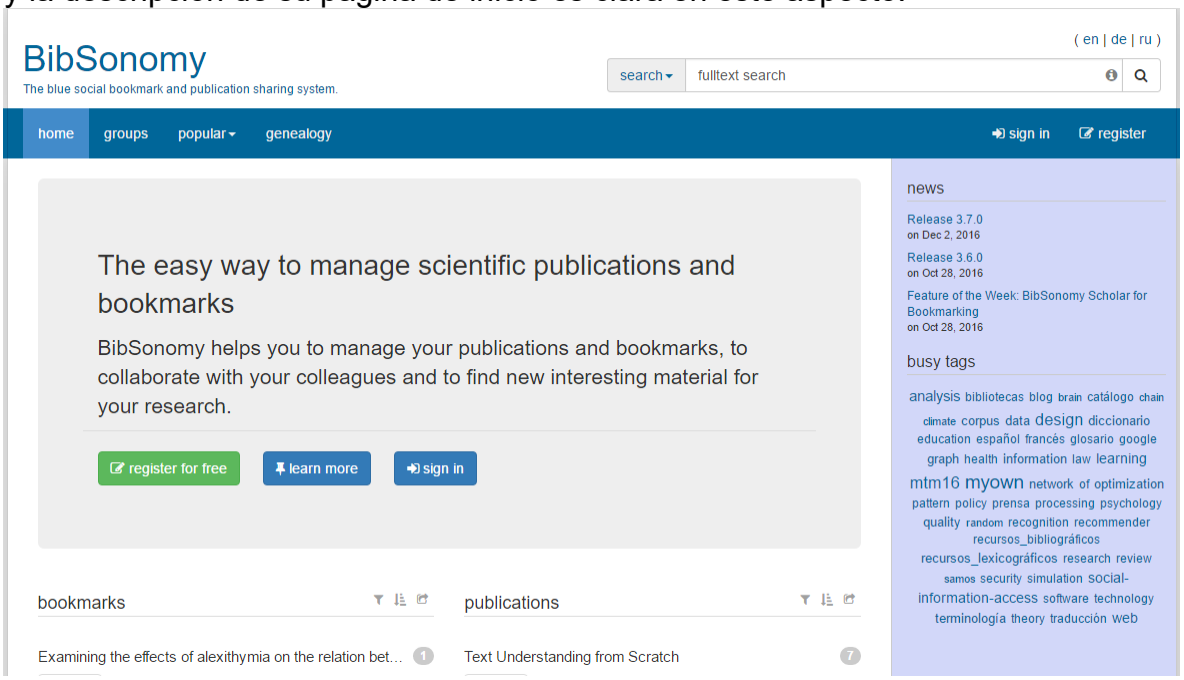


Ilustración 0-14, página inicial bibsonomy, [CITATION Bre \l 9226]

Al igual que DBLP, este repositorio ofrece una interfaz con opciones de búsqueda, cambio de lenguaje, exploración por autores o por grupos y un menú lateral con noticias y “tags” relacionados.

Para buscar en bibsonomy, me dirijo a la exploración por autor dentro del menú “popular”. A diferencia de dblp, este repositorio ofrece una opción de “nube” de tags en las que es posible ver el listado de autores en orden de popularidad resaltando el tamaño del apellido.

Se hace la búsqueda con el mismo autor en este repositorio y se examina qué información arroja el motor de búsqueda, sobre todo que datos en formato RDF entrega este motor.

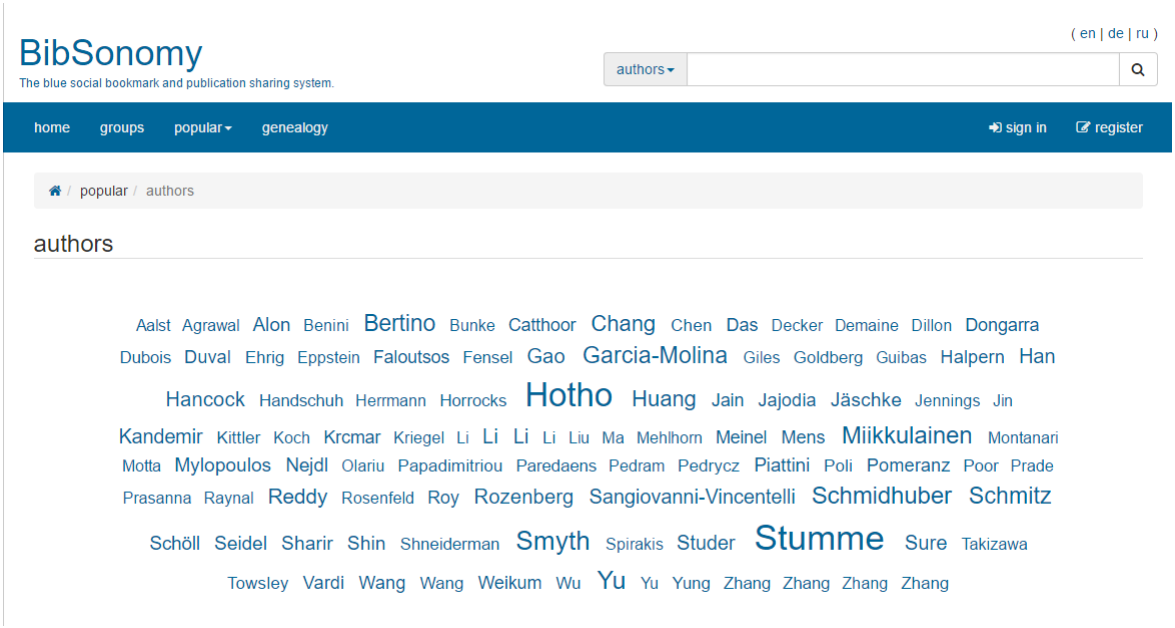


Ilustración 0-15, nube de autores bibsonomy,[CITATION Bre \l 9226]

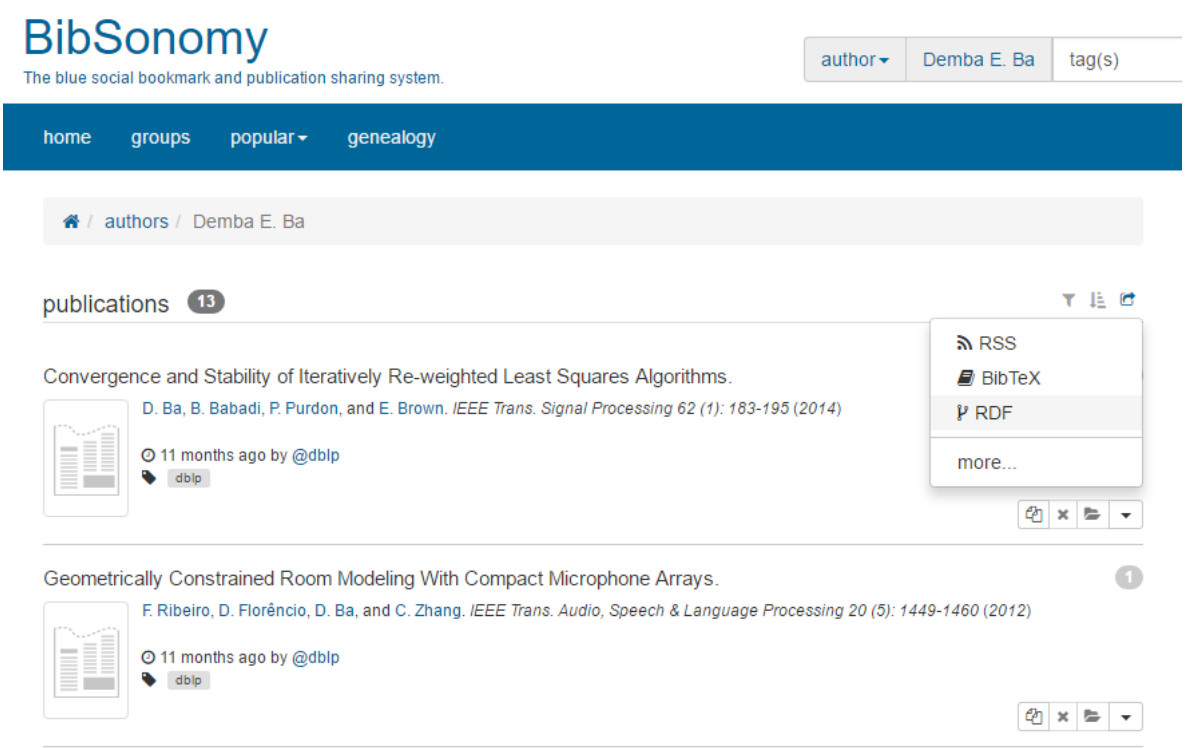


Ilustración 0-16, resultados búsqueda por publicación[CITATION Bre \l 9226]

Los resultados obtenidos son un poco más numerosos para ese autor en este sitio web y tal como lo muestra el menú emergente, también es posible generarlos en formato RDF.

Una vez determinado que el sitio ofrece la posibilidad de cargar la URI con formato rdf, se captura esa dirección y se carga en java para probar el modelo y determinar la información contenida:

```
import org.apache.jena.rdf.model.Model;


public class ReadRDFModel
   {

    public static final String STRING ="https://www.bibsonomy.org/layout/bibordf/author/Demba%20E.%20Ba?items=100";

    public static void main(String[] args)
    {

        Model model = ModelFactory.createDefaultModel();
        model.read(STRING, "N3");
        model.write(System.out,"N3");
    }
   }
```

Console

```
<terminated> ReadRDFModel [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (18/01/2017, 11:43:29 a.m.)
log4j:WARN No appenders could be found for logger (Jena).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix bibo:  <http://purl.org/ontology/biblio/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .

<http://dx.doi.org/>  a    bibo:Document , bibo:Article ;
        dc:date           "2013" , "2008" , "2012" , "2006" , "2014" , "2007" , "2010" ;
        dc:title          "Exact and Stable Recovery of Sequences of Signals with Sparse Increments via Differential _1-Minimization." , "Turning enemies
        dcterms:isPartOf  [ a         bibo:Journal ;
                            dc:title  "IEEE Trans. Multimedia"
                          ] ;
        dcterms:isPartOf  [ a         bibo:Journal ;
                            dc:title  "Neural Computation"
                          ] ;
        dcterms:isPartOf  [ a         bibo:Journal ;
                            dc:title  "IEEE Trans. Audio, Speech &#x0026; Language Processing"
                          ] ;
        dcterms:isPartOf  [ a         bibo:Journal ;
                            dc:title  "IEEE Trans. Signal Processing"
                          ] ;
        dcterms:isPartOf  [ a         bibo:Journal ;
                            dc:title  "Front. Comput. Neurosci."
                          ] ;
        dcterms:isPartOf  [ a         bibo:Journal ;
                            dc:title  "IEEE Trans. Audio, Speech &#x0026; Language Processing"
                          ] ;
        dcterms:isPartOf  [ a         bibo:Journal ;
                            dc:title  "Front. Comput. Neurosci."
                          ] ;
```

Ilustración 0-17, Resultados de la validación del modelo rdf en java, [CITATION Bre \l 9226]

Tal y como se observa en la imagen, los resultados del modelo fueron satisfactorios y la información es perfectamente válida. Ahora procedo a revisar el "front end" del repositorio.

Para este repositorio no pude hallar el "front end" para consultas usando SPARQL directamente en el sitio.

Es un repositorio que les permite a los usuarios compartir y anotar marcadores y publicaciones, la plataforma está basada en JAVA y con MySQL como base de datos de soporte y motor de búsqueda provisto por ElasticSearch.

El objetivo del sitio es darle a cada usuario la posibilidad de crear colecciones de marcadores y publicaciones, creando bibliografías e insertándolas en sus propios trabajos.
La plataforma entrega una API para efectos programáticos que tiene una lista de métodos que se pueden hallar en la siguiente url:

Entre estos métodos se destaca la posibilidad de búsqueda a texto completo.
No posee el sitio un “end point” que de la posibilidad al usuario de crear sus propias consultas similar al hallado en dblp.
La documentación referente al formato de intercambio de datos en formato XML esta contenida en este documento: ,
Entre los elementos destacables están la existencia de grupos, posts, tags y resourcehash, la definición de los tipos de usuario y los tipos de grupo.

Los siguientes métodos están disponibles dentro para la API del repositorio:
Una descripción completa de su base de datos esta disponible en esta dirección: ,
Cabe destacar que bibsonomy basa gran parte de su información en DBLP.

Al carecer de un “end point” bibsonomy ya presenta una desventaja en relación a dblp, pero el enfoque de bibsonomy es algo diferente ya que trabaja más como una colección de enlaces, libros y referencias de libros que cada usuario crea. Por lo tanto son modos distintos de acceder a la misma información.

El siguiente repositorio fue DBPEDIA.

DBPedia es un repositorio de datos mucho más amplio que los anteriormente señalados, no quiere decir el término “amplio” que también incluya aquellos datos académicos existentes en los dos anteriores, me refiero a la variedad de contenido que posee, mucho de ese contenido derivado de Wikipedia.

Dbpedia también tiene un “front end” en sparql, muy similar a dblp. Se realizaron búsquedas en ese front end para el mismo autor que anteriormente había buscado en los otros dos repositorios.

Después de escoger y probar infructuosamente con varios autores, determino que la base de dato académica de este repositorio es bastante más reducida que las anteriores, para uno de los autores que encontré en este repositorio la búsqueda me arroja la descripción del autor, más no sus trabajos en detalle.

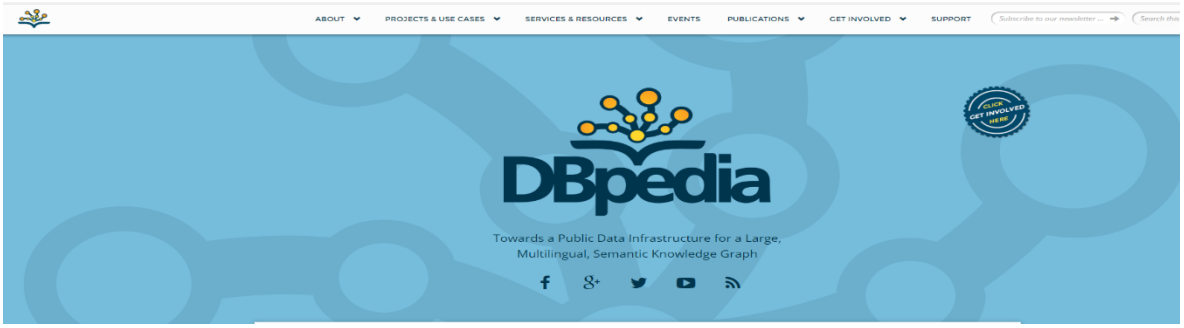


Fig. wiki de dbpedia

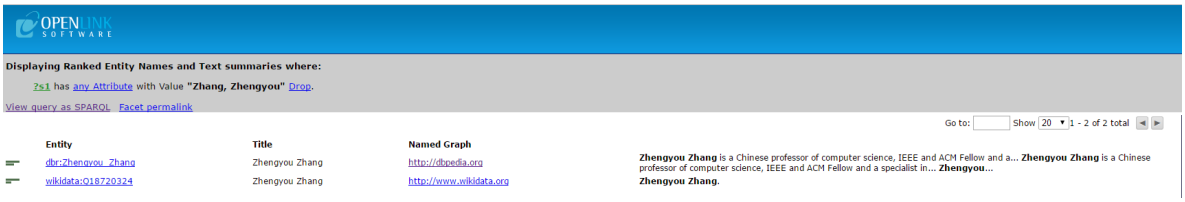


Ilustración 0-18, Resultados búsqueda de un autor académico, [CITATION Bre \l 9226]

La búsqueda de un autor en el ramo académico, arroja un listado de afiliaciones del investigador, este motor antes de enfocarse en las publicaciones, muestra entre otras cosas aparte de las afiliaciones, el lugar de trabajo, los premios, los reconocimientos y enlaces a través de la etiqueta “sameAs” a otras páginas o recursos donde complementar la información del autor.

Los datos que arroja la página de resultados de la búsqueda por autor, ya contienen información RDF tal y como se puede apreciar en la imagen siguiente.

Dbpedia comparte un “front end” sparql similar a dblp, sin embargo su sintaxis es un poco distinta así como los resultados que arroja

c1	c2	sc	rank	g
http://dbpedia.org/resource/Zhengyou_Zhang	Zhengyou Zhang.	8.4	6.64381	http://dbpedia.org
http://dbpedia.org/resource/Zhengyou_Zhang	Zhengyou Zhang is a Chinese professor of computer science, IEEE and ACM Fellow and a specialist in...	8.4	6.64381	http://dbpedia.org
http://dbpedia.org/resource/Zhengyou_Zhang	Zhengyou Zhang is a Chinese professor of computer science, IEEE and ACM Fellow and a specialist in...	8.4	6.64381	http://dbpedia.org
http://dbpedia.org/resource/Zhengyou_Zhang	Zhengyou Zhang.	8.4	6.64381	http://dbpedia.org
http://dbpedia.org/resource/Zhengyou_Zhang	Zhang, Zhengyou.	8.4	6.64381	http://dbpedia.org
http://dbpedia.org/resource/Zhengyou_Zhang	Zhang, Zhengyou.	8.4	6.64381	http://dbpedia.org
http://www.wikidata.org/entity/Q18720324	Zhengyou Zhang.	8.4	4.00032	http://www.wikidata.org

Ilustración 0-19, resultados búsqueda usando SPARQl, [CITATION Bre \l 9226]

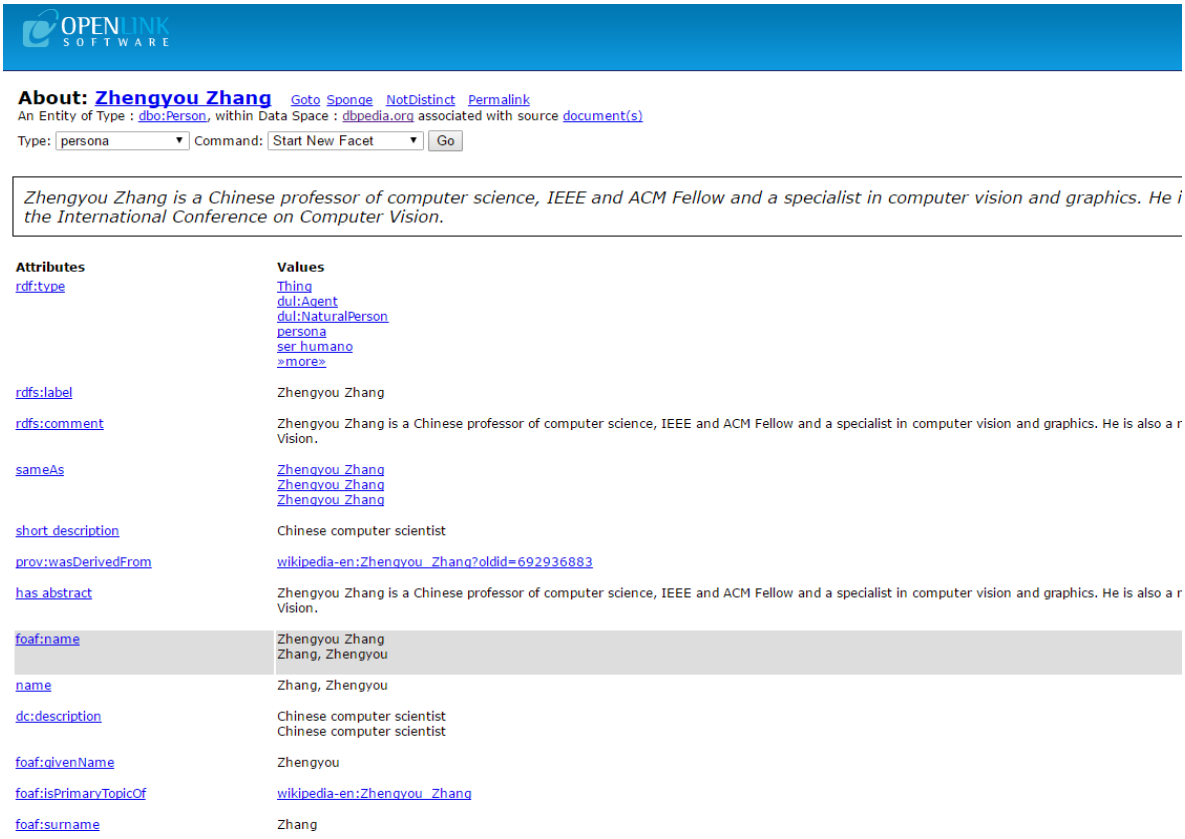


Ilustración 0-20, resultados búsqueda usando motor de dbpedia, [CITATION Bre \l 9226]

Ilustración 0-21, "Front end" sparql dbpedia [CITATION Bre \l 9226]

DBPedia es un buen repositorio pero su enfoque más general lo hace poco adecuado para el ámbito académico, su uso no es tan intuitivo como dblp y su "front end" sparql, aunque similar a dblp, es algo más complicado de manipular.

ANÁLISIS DEL REPOSITORIO ESCOGIDO

En razón del análisis de los repositorios anteriormente presentado se ha escogido el repositorio DBLP, enumero algunas de las razones de dicha selección:

Es el sitio de referencia, en línea, para las más importantes publicaciones en ciencias de la computación. La información contenida en el repositorio es de tipo enteramente académico, sus principales características son:

Más de 1 millón de autores con cerca de 3 millones de publicaciones en 32 mil volúmenes de journals, 31 mil conferencias y 23 mil monografías.
La composición del repositorio viene dada así (según las estadísticas del sitio):
53% del contenido está relacionado con conferencias y talleres
Cerca de un 40% son artículos de journals
El resto, un 7% se distribuye entre libros, tesis, trabajos referenciados, partes de libros o colecciones, publicaciones informales y trabajos de edición.
La mayor parte de las publicaciones están elaboradas entre 2 y 4 autores con un promedio de 2 a 3 publicaciones por autor.

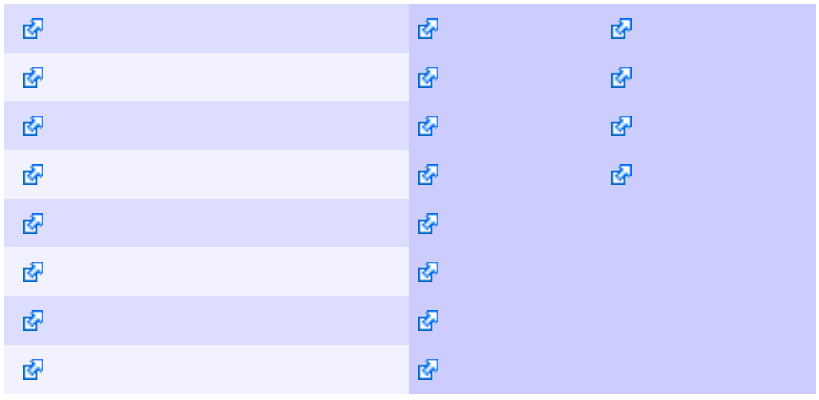
Tan solo estas estadísticas garantizan una buena fuente de información y recursos en formato RDF para probar y lograr hacer funcionar el programa de búsqueda y carga de información.

La lista de campos disponible en el repositorio es la siguiente:

"author|editor|title|booktitle|pages|year|address|journal|volume|number|month|url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter|publnr"

En donde cada campo posee al menos dos atributos principales: key y mdate, en donde key es la llave única del registro y mdate es la fecha de la última modificación del registro con formato ISO 8601.

Las propiedades RDF de la información son las siguientes (haciendo una consulta sobre el endpoint de dblp):



Adicional a esto, este sitio presenta un “end point” sparql en esta dirección: , a través del cual los investigadores pueden explorar todo el contenido semántico creando sus propias consultas SPARQL.

La ventaja de este repositorio precisamente es la variedad y calidad de la información contenida, esta estructuración de los datos y la gran cantidad de los mismos, hacen relativamente fácil buscar y administrar el contenido en relación a otros repositorios que carecen de tanta información o presentan menos formas y herramientas de acceder a la información.

Aunque no se utilizaron sentencias SPARQL en el programa JAVA, si se utilizó el “front end” de este repositorio para fines de verificar los resultados de algunas consultas hechas con la aplicación.

Se ejecutan las consultas sobre el repositorio DBLP, cada búsqueda de autor, como es lógico, genera un conjunto distinto de resultados, estos resultados se ilustran a continuación y algunos de esos resultados se explican.

Resultados del ejercicio con distintos autores sobre el repositorio DBLP:

Autor: Yasumasa Baba

Total resultados para busqueda dada: 1 enlaces relevantes

Para cada resultado se obtiene la siguiente información:

```
.....
log4j:WARN No appenders could be found for logger (Jena).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Informacion para: <http://dblp.uni-trier.de/rec/rdf/journals/lalc/IyeiriYB11.rdf>:
 - <http://purl.org/dc/terms/modified> : <2011-09-30T13:15:48+0200>
 - <http://purl.org/dc/terms/license> : <http://www.opendatacommons.org/licenses/by/>
 - <http://purl.org/dc/terms/isPartOf> : <http://dblp.org/rdf/dblp.rdf>
 - <http://purl.org/dc/terms/creator> : <http://dblp.org>
 - <http://www.w3.org/2000/01/rdf-schema#label> : <provenance information for RDF data of dblp
record 'journals/lalc/IyeiriYB11'>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> : <http://dblp.org/pers/i/Iyeiri:Yoko>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#primaryElectronicEdition> :
<http://dx.doi.org/10.1093/llc/fqr005>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#yearOfPublication> : <2011>
 - <http://www.w3.org/2002/07/owl#sameAs> : <http://dblp.org/rec/journals/lalc/IyeiriYB11>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publishedInJournalVolumeIssue> : <2>
```

Tabla 4, [CITATION Bre \l 9226]

El esquema de información presentada por el aplicativo sigue la misma estructura para todas las búsquedas; se presenta primero un título indicando cuantos resultados han sido obtenidos con las palabras ingresadas por parámetros al momento de ejecutar la aplicación, luego se indica que por cada enlace visitado se obtiene una información en formato RDF (aunque puede ser ninguno), dichos datos tienen una estructura similar compuesta por lo general de los siguientes términos:
una línea con información de los autor(es) del documento encontrado
Una línea o varias líneas con los datos de publicación
Una o varias líneas con información del autor en otros enlaces.

Para el caso del autor anterior (Yasumasa Baba), aparece una publicación en el año 2011 en conjunto con Yoko Iyeiri, el título de la publicación es: provenance information for RDF data of dblp record, y fue publicado en el volumen 2 del año 2011.

Autor: Jorge de-la-Peña-Sordo

Total resultados para busqueda dada: 7 enlaces relevantes

Para cada resultado se obtiene la siguiente información:

```
.....
log4j:WARN No appenders could be found for logger (Jena).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Informacion para: <http://dblp.uni-trier.de/rec/rdf/journals/igpl/BringasdPHQC16.rdf>:
 - <http://purl.org/dc/terms/modified> : <2016-12-19T16:08:36+0100>
 - <http://purl.org/dc/terms/license> : <http://www.opendatacommons.org/licenses/by/>
 - <http://purl.org/dc/terms/isPartOf> : <http://dblp.org/rdf/dblp.rdf>
 - <http://purl.org/dc/terms/creator> : <http://dblp.org>
 - <http://www.w3.org/2000/01/rdf-schema#label> : <provenance information for RDF data of dblp
record 'journals/igpl/BringasdPHQC16'>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publishedInJournal> : <Logic Journal of the
IGPL>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> :
<http://dblp.org/pers/p/Puerta:Jos=eacute=_Gaviria_de_la>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> :
<http://dblp.org/pers/d/de=la=Pe=ntilde=a=Sordo:Jorge>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> :
<http://dblp.org/pers/h/Herrero:=Aacute=lvaro>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publishedInJournalVolumeIssue> : <6>

.....
Informacion para: <http://dblp.uni-trier.de/rec/rdf/conf/softcomp/de-la-Pena-SordoSPB13.rdf>:
 - <http://purl.org/dc/terms/modified> : <2016-07-28T16:35:20+0200>
 - <http://purl.org/dc/terms/license> : <http://www.opendatacommons.org/licenses/by/>
 - <http://purl.org/dc/terms/isPartOf> : <http://dblp.org/rdf/dblp.rdf>
 - <http://purl.org/dc/terms/creator> : <http://dblp.org>
```

```
 - <http://www.w3.org/2000/01/rdf-schema#label> : <provenance information for RDF data of dblp
record 'conf/softcomp/de-la-Pena-SordoSPB13'>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publishedAsPartOf> :
<http://dblp.org/rec/conf/softcomp/2013>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> :
<http://dblp.org/pers/d/de=la=Pe=ntilde=a=Sordo:Jorge>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> : <http://dblp.org/pers/s/Santos:Igor>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> :
<http://dblp.org/pers/b/Bringas:Pablo_Garc=iacute=a>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#bibtexType> :
<http://data.bibbase.org/ontology/#Inproceedings>

.....
```

Tabla 5,[CITATION Bre \l 9226]

Para este autor, la búsqueda arrojó un total de 7 resultados relevantes, de esos enlaces analizamos el último mostrado, de este resultado se concluye:

- Tuvo coautoría con Igor Santos.
- Fue un artículo publicado en el año 2016 en los proceedings de ontología.
- Tiene título: provenance information for RDF data of dblp record

Autor: M. Mbarawa

Total resultados para busqueda dada: 1 enlaces relevantes

Para cada resultado se obtiene la siguiente información:

```
.....
log4j:WARN No appenders could be found for logger (Jena).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Informacion para: <http://dblp.uni-trier.de/rec/rdf/journals/engl/OgengaLM09.rdf>:
 - <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> : <http://dblp.uni-trier.de/rdf/schema-2016-11-
11#Publication>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publicationType> : <http://dblp.uni-
trier.de/rdf/schema-2016-11-11#Article>
 - <http://www.w3.org/2002/07/owl#sameAs> : <http://dblp.org/rec/journals/engl/OgengaLM09>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publishedInJournal> : <Engineering Letters>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#primaryElectronicEdition> :
<http://www.engineeringletters.com/issues_v17/issue_3/EL_17_3_06.pdf>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publishedInJournalVolumeIssue> : <3>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#yearOfPublication> : <2009>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#title> : <Sulphur Dioxide Abetment Using
Synthesized South African Limestone/Siliceous Sorbents.>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#bibtexType> :
<http://data.bibbase.org/ontology/#Article>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> :
<http://dblp.org/pers/m/Mbarawa:M=>
```

Tabla 6,[CITATION Bre \l 9226]

Del señor M. MBarawa se tiene que:

- El título del ensayo es: Sulphur Dioxide Abetment Using Synthesized South African Limestone/Siliceous Sorbents.
- Fue publicado en la revista “Engineering Letters” en el 2009.

Tal y cómo se indica al principio, cada autor posee un numero diferente de “hits” o resultados de búsqueda y si existe información en formato RDF de cada uno de esos resultados, esta información se presenta en consola. Claro que si no existe nada que mostrar, es decir los resultados son 0, entonces nada aparece en la consola.

2.9 DESARROLLO FINAL DE LA API.

Finalmente la solución propuesta ha tomado un camino ligeramente distinto en razón de la dinámica de desarrollo, iniciando con un diseño nuevo de los procesos o eventos para el análisis de una URI:

DISEÑO DE PROCESOS O EVENTOS

A pesar de la propuesta inicial y debido a la dinámica de desarrollo, el proceso de análisis de una dirección RDF para determinar de un autor su artículo varió con respecto a los estimados iniciales y ahora se basa en el diagrama siguiente:

Diagrama de eventos o pasos para leer la información de un autor.

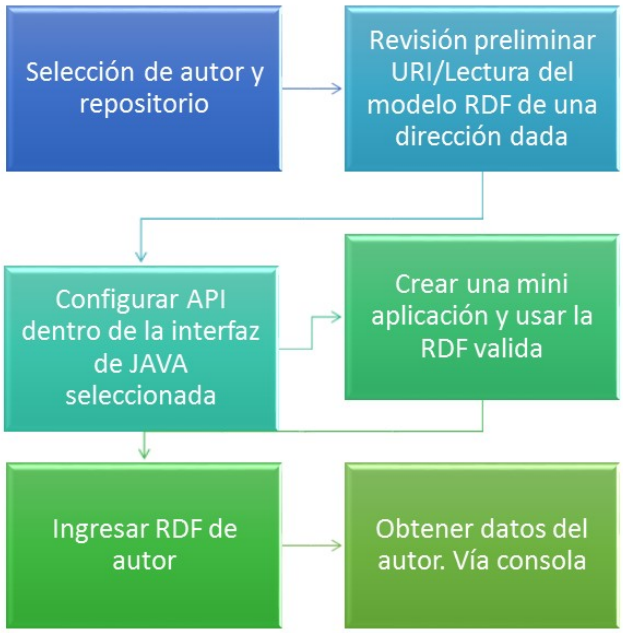


Ilustración 0-22, Diagrama de pasos para leer la información de un autor.

Es así como el proceso de la solución tiene 6 etapas:

1. Escoger un repositorio y la correspondiente URI de un autor determinado, esta selección de repositorios se basó en la información proporcionada por el enunciado del TFM asi como diversas fuentes de internet, la elección del autor fue meramente aleatoria.
2. URI ha de entregarse a la API en formato RDF, se hace una revisión preliminar de esta URI con una clase/API muy sencilla que solo toma la URI y carga un modelo en memoria, mostrando los contenidos de la URI en la

consola JAVA. También se hizo una revisión de la URI utilizando los front-end de cada dirección.

El repositorio escogido es uno de los planteados en la propuesta de proyecto: , este es el repositorio de artículos académicos DBLP.
Un autor, Olivier van der A fue escogido inicialmente para las pruebas de los algoritmos, primero se hace una búsqueda en un motor de búsqueda tradicional (google), nótese la variedad y dispersión de los resultados.

Cabe aclarar que estos resultados tienen tal variedad debido a varios factores:

- Se usa la versión estándar del motor de búsqueda.
- No se usaron filtros para refinar la búsqueda a sitios académicos
- No se enfocó la búsqueda en un sitio web específico.

El autor seleccionado es: Olivier van der Aa

El primer articulo es: ***Interoperation of world-wide production e-Science infrastructures.***
Concurrency and Computation: Practice and Experience 21(8): 961-990 (2009)

Resultados manuales:
Google web search para el tema (muestra de los resultados, no se muestran todos):
About 3,170,000 results (0.50 seconds)

Interoperation of world-wide production e-Science infrastructures

*of **world** - **wide production e** - **Science infrastructures** - Riedel - Cited by 57*

Search Results
[PDF]
https://www.ogf.org/OGF_Special_Issue/cpeginpaper_riedel.pdf
*Many **production** Grid and **e** - **Science infrastructures** have begun to offer services to end- ... Grid **infrastructures** to reach the goal of a **world** - **wide** Grid vision on a ...*
onlinelibrary.wiley.com/doi/10.1002/cpe.1402/full
by M Riedel - 2009 - -
*Mar 24, 2009 - Abstract. Many **production** Grid and **e** - **Science infrastructures** have begun to offer services to end-users during the past several years with an ...*
cds.cern.ch/record/1706673
by M Riedel - -
*Jun 5, 2014 - Many **production** Grid and **e** - **Science infrastructures** have begun to offer ... **production** Grid **infrastructures** to reach the goal of a **world** - **wide** Grid ...*
juser.fz-juelich.de/record/861/
by M Riedel - 2009 - -
*Nov 13, 2012 - Many **production** Grid and **e** - **Science infrastructures** have begun to offer services to end-users during the past several years with an increasing ...*
*www.academia.edu/.../ **Interoperation_of_world** - **wide** _ **production** _ **e** - **Science** _ **infrastru** ...*
*Many **production** Grid and **e** - **Science infrastructures** have begun to offer services to end-users during the past several years with an increasing number of ...*
*www.academia.edu/.../ **Interoperation** _scenarios_of_ **production** _ **e** - **science** _ **infrastructu** ...*

*... continuing **interoperation** efforts to reach the goal of a **world** - **wide** Grid vision on .... There are a lot of **production e** - **Science infrastructures** that all support wide ...* *https://books.google.com.co/books?isbn=3893368612* *2013* *... standardization process still ongoing within the **Production** Grid **Infrastructure** (PGI) ... of **World** — **wide** Concurrency **Interoperation Production e** — **Science** and ...* *dl.acm.org/citation.cfm?id=1542055.1542056* *by M Riedel - 2009 - -* *Jun 1, 2009 - **Interoperation of world-wide production e-Science infrastructures** ... E. Laure, Open Grid Forum—Grid Interoperation Now (GIN)—Community ...* *eprints.wmin.ac.uk/11070/* *by M Riedel - 2009 - -* *Aug 7, 2012 - **Interoperation of world-wide production e-science infrastructures** ... of Electronics and Computer Science at the University of Southampton.* *https://experts.illinois.edu/.../ **interoperation-of-world** - **wide** - **production** - **e** - **science** - **infra** ...* *Jun 10, 2009 - **Interoperation of world-wide production e-Science infrastructures** . M. Riedel ; E. Laure ; Th Soddemann ; L. Field ; J. P. Navarro ; J. Casey ; M.* *Resultados para búsqueda por Autor, google search:* *About 1,720,000 results (0.68 seconds)*

Tabla 7, Ejemplo de busqueda de informacion en web no semantica

Se puede observar el problema clásico de la búsqueda en motores de búsqueda tradicionales, el exceso de información en muchos casos desconectada entre sí o información de poca relevancia.

3. Determinada la viabilidad de la URI, se configura la interfaz de programación, en este caso ECLIPSE en combinación con los repositorios JENA, el conjunto de librerías JENA posee clases que permiten manipular conjuntos de URIs y obtener la información contenida o relacionada a las URIs.

   Para programar una solución en ECLIPSE-JENA, se requiere un diseño de casos de uso, diagramas de proceso y diagramas de clase:

CASOS DE USO.

No hay más caso de uso para la API que el de leer una URI proporcionada por el usuario y desglosarla en sus distintos componentes, esto queda expresado en el diagrama a continuación:

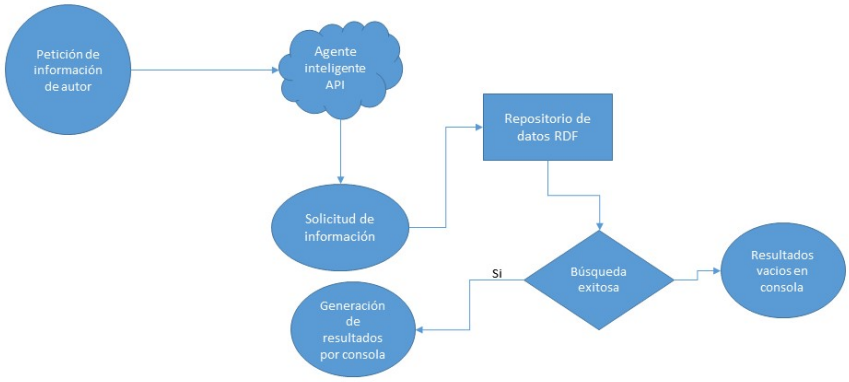


Ilustración 0-23, Diagrama de caso de uso. [CITATION Bre \l 9226]

Este diagrama en su versión final, excluye el paso de representación de los resultados en formato html o xml ya que al ser un api de tipo académico, la representación mencionada no es necesaria.

FLUJO DE PROCESOS

El flujo de procesos de la API es bastante simple; se envía una URI, esta es URI es expandida por JENA al ser incorporada dentro de un “modelo” que se encarga de cargar en memoria las tripletas que componen la estructura de la dirección de recurso o URI.

Una vez desglosada la estructura, la interfaz acude a la dirección para identificar los elementos RDF, y genera una consulta al repositorio de datos.

Resulta la consulta, la información es almacenada en memoria para luego ser mostrada en la consola JAVA.

De esta consulta se obtiene, tanto el título del trabajo o trabajos del autor, como sus coautores y fechas y lugar de publicación de los trabajos, siempre y cuando esta información este contenida en las tripletas RDF.

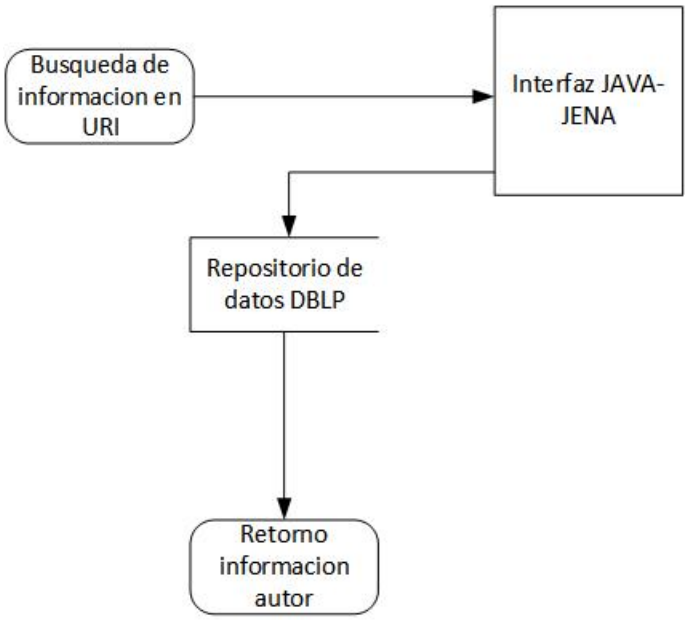


Ilustración 0-24, Diagrama de flujo de procesos.[CITATION Bre \l 9226]

CLASES IMPLEMENTADAS.

La solución propuesta reduce a dos las clases principales, una que se encarga de todo el trabajo de análisis del modelo en formato RDF del sitio o URI, y otra clase cuya principal función es la de almacenar todas aquellas direcciones complementarias a la dirección inicial y que sean objeto de escrutinio por parte del código.

Existe una tercera clase encargada de hacer el análisis independiente de una URI a fin de determinar si el contenido es relevante, dicha clase es anexa y opcional y puede ser obviada o no considerada como parte de la solución.

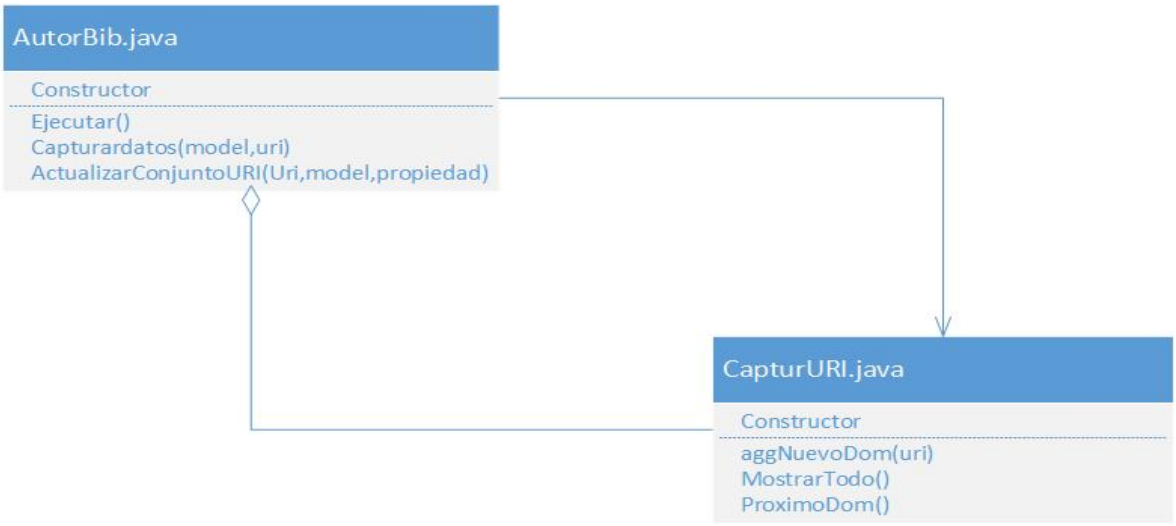


Ilustración 0-25, Diagrama de clases. [CITATION Bre \l 9226]

4. Se crea una mini aplicación, en este caso una clase que utilice la API desarrollada para analizar URIs, esta clase llama a los constructores de la API y genera resultados por consola con el detalle del contenido de la URI.

   La aplicación se desarrolla usando la plataforma de desarrollo Eclipse y la API de lectura y escritura para RDF y SPARQL - JENA, el código para la clase “AutorBib” se resume a continuación.

   ### CODIGO FUENTE

   Para elaborar este código se partió de una base de codificación presente en el documento: A Developer’s Guide to the Semantic Web, Liyang Yu

```
public class AutorBib
{
	//esta clase hace la captura de la URI en formato RDF y crea un modelo en memoria
	private CapturURI MismaURI = null;
	public AutorBib(String uri) {
	MismaURI = new CapturURI();
	MismaURI.aggNuevoDom(uri);
}

public void ejecutar()
{
	//este procedimiento hace los llamados, crea el modelo y busca las siguientes URI
	String URIActual = MismaURI.ProximoDom();
	if (URIActual == null)
	{
		return;
	}

	try
	{

	 //tomando como base la URI en formato RDF, estos pasos capturan el modelo y lo llevan a memoria
	 Model model = ModelFactory.createDefaultModel();
	 model.read(URIActual);

	 // captura del conjunto de datos
	 capturarDatos(model,URIActual);

	 // en caso de existir, busca enlaces adicionales determinados por el ultimo parametro de la funcion.
	 actualizarConjuntoURI(MismaURI,URIActual,model,OWL.sameAs);

	 }
	catch (Exception e)
	{
		System.out.println("Se han presentado errores al acceder a: (" + URIActual + ")" +
e.toString());
	}

	System.out.println("\n Los siguientes enlaces se pueden explorar: " );
	MismaURI.Mostrartodo();



	 // following our nose
	ejecutar();

	}
```

```
private void capturarDatos(Model model, String uri)
{
	if ( uri == null )
{
	return;
}
	//este procedimiento se encarga de capturar los datos el modelo y presentarlo por medio de consola

	int factCounter = 0;

	System.out.println("Informacion para: <" + uri + ">:");
	// procedimientos de control y verificacion

	for (StmtIterator si = model.listStatements(); si.hasNext();)
	{
		Statement statement = si.nextStatement();

		if ( uri.equalsIgnoreCase(statement.getSubject().getURI()) != true)
		{
			factCounter ++;
			//System.out.println("dentro de for");
			System.out.print(" - <" +statement.getPredicate().toString() + "> : <");
			System.out.println(statement.getObject().toString()		+ ">");
			if ( factCounter >= 10 )
			{
				return;
			}
		}
	}
}

private void actualizarConjuntoURI(CapturURI uriCollection,String uri, Model model,Property property)
{
	if ( uri == null )
	{
		return;
	}

	 // verificacion del objeto, aqui este procedimiento se encarga de abrir las URI con la propiedad definida en el enunciado
	 Resource resource = model.getResource(uri);
	 //busqueda por nodos, propiedades definidas
	 NodeIterator objects=model.listObjectsOfProperty(property);

	 //System.out.println("antes del while");
	 while ( objects.hasNext() )
	 {
		 //System.out.println("dentro del while");
		 RDFNode object = objects.next();
		 if ( object.isResource() )
		 {
			 //System.out.println("dentro del if");
			 Resource tmpResource = (Resource)object;
			 uriCollection.aggNuevoDom(tmpResource.getURI());
		 }
	 }
	 }
```

Tabla 8, Código fuente clase 1. [CITATION Bre \l 9226]

Correspondientemente el código para la clase CapturURI se ilustra a continuación:

```
public class CapturURI {
	//Esta clase permite capturar las URI relacionadas con la direccion inicial y crear una lista en memoria de estas URIs
	private Stack URI = null;
```

```
        private HashSet dominios = null;

        public CapturURI()
        {
                URI = new Stack();
                dominios = new HashSet();
        }

        public void aggNuevoDom(String uri)
        {
                if ( uri == null ) {
                        return;
        }
        try
        {
                //Este procedimiento adiciona las nuevas URI a la lista en memoria, tener en cuenta que si la
URI ya fue adicionada, este "if" evita que se adicione dos veces
                URI thisURI = new URI(uri);
                if ( dominios.contains(thisURI.getHost())           == false )
                {
                        dominios.add(thisURI.getHost());
                        URI.push(uri);
                }
        }
        catch(Exception e) {};
        }

        public String ProximoDom()
        {
                //Salta al proximo dominio
                if ( URI.empty() == true )
                {
                        return null;
                }
                return (String)(URI.pop());
        }

        public void Mostrartodo()
        {
                //Muestra en consola todas las URI almacenadas
                for ( int i = 0; i < URI.size(); i ++ )
                {
                        System.out.println(URI.elementAt(i).toString());

                }
        }


}
```

Tabla 9, Código fuente clase 2.[CITATION Bre \l 9226]

5. Para hacer una búsqueda por autor en el repositorio seleccionado, se crea una clase que por paso de argumentos al ejecutar el código JAVA se indica el nombre del autor cuyo formato de ingreso debe ser: nombre1+nombre2+nombre3. Luego esta clase ejecuta la búsqueda en el repositorio y para cada resultado valido (URI en formato RDF), hace los llamados respectivos a las dos clases anteriores para obtener la información del autor indicado. Es decir se ha creado un “motor de búsqueda” que usa el motor de búsqueda propio del repositorio para obtener los resultados solicitados.
   La clase que se elabora se presenta a continuación:

```
public class AutorBus
```

```
{
	public static final String startURI ="http://dblp.uni-trier.de/search?q=";

	public static void main(String[] args)  throws IOException
	{


	Document doc;
	try {


			doc = Jsoup.connect(startURI + args[0]).get();
			Elements links = doc.select("a[href ~=\\.(rdf)]");

				System.out.println("\n Total resultados para busqueda dada: " + links.size() + "
enlaces relevantes");
				System.out.println("\n Para cada resultado se obtiene la siguiente información: ");
			for (Element link : links) {
				//System.out.println("\nsrc : " + link.attr("href"));
				System.out.println("\n.....");
				AutorBib fyn = new AutorBib(link.attr("href"));
				fyn.ejecutar();
			}

	} catch (IOException e) {
			e.printStackTrace();
	}

    }
```

Tabla 10, Código fuente clase 3. [CITATION Bre \l 9226]

6. Se observan los resultados por consola JAVA, se obtienen datos por cada uno de los resultados de la búsqueda, por cuanto el autor determinado se busca no solo como autor sino como co-autor. (X representa un numero de resultados de consulta dependiendo de cada autor):

```
Total resultados para búsqueda dada: X enlaces relevantes

 Para cada resultado se obtiene la siguiente información:

Información para: <http://dblp.uni-trier.de/rec/nt/conf/iciap/PirloRVI15.nt>:
 - <http://purl.org/dc/terms/modified> : <2015-08-21T11:37:23+0200>
 - <http://purl.org/dc/terms/license> : <http://www.opendatacommons.org/licenses/by/>
 - <http://purl.org/dc/terms/isPartOf> : <http://dblp.org/rdf/dblp.nt>
 - <http://purl.org/dc/terms/creator> : <http://dblp.org>
 - <http://www.w3.org/2000/01/rdf-schema#label> : <provenance information for RDF data of dblp record
'conf/iciap/PirloRVI15'>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#publishedAsPartOf> : <http://dblp.org/rec/conf/iciap/2015w>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#primaryElectronicEdition> : <http://dx.doi.org/10.1007/978-3-319-
23222-5_30>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> : <http://dblp.org/pers/r/Rizzi:Fabrizio>
 - <http://www.w3.org/2002/07/owl#sameAs> : <http://dblp.org/rec/conf/iciap/PirloRVI15>
 - <http://dblp.uni-trier.de/rdf/schema-2016-11-11#authoredBy> : <http://dblp.org/pers/p/Pirlo:Giuseppe>
```

Tabla 11, Resultados proceso inspección URI.[CITATION Bre \l 9226]

3. CONCLUSIONES

Entendiendo los objetivos planteados, estas son las conclusiones al trabajo:

- Se logró una comprensión adecuada del concepto de “linked data” y la web semántica, el estudio de las numerosas fuentes en internet, las páginas de estos conceptos disponibles en la red y en la W3C, me permiten entender que es lo que Tim Berners Lee deseaba cuando creo lo que hoy conocemos como HTML. A destacar de este proceso de aprendizaje es el relativo estado de maduración por el que atraviesa la web semántica; aún la mayoría de sitios no poseen una estructura de tripletas y todavía la mayor parte de la red está diseñada para ser explorada por personas, muchos de los sitios que existen están dedicados a la academia o la investigación (fin último de Tim Berners Lee) y están por lo tanto lejos de aquellos que no se encuentran específicamente en este segmento de la informática. El futuro de una red totalmente semántica está ligada a los esfuerzos y apoyo que puedan brindar los grandes actores de la computación cuyos nombres omito. Para que este tipo de redes pueda avanzar a buen ritmo se requiere de herramientas más fáciles de manipular, métodos de construcción de páginas más rápidos y software con baja curva de aprendizaje.

- Uno de los éxitos de las tecnologías móviles es la relativa facilidad con la que se pueden elaborar contenidos y publicarlos, esta facilidad fue la que impulso el tremendo florecimiento de aplicaciones móviles de todo tipo. Algo similar se requiere para la web semántica, quizás el advenimiento de los asistentes al estilo “alexa” o “amazon eco” le den el impulso que necesita esta nueva variación de la red, estos asistentes se beneficiarían grandemente pues se convertirían en verdaderos mayordomos digitales al acceder a contenidos completos y claros en la red.

- Aunque se analizaron distintos repositorios de datos, entre ellos DBPEDIA, finalmente y dada la naturaleza netamente académica de este trabajo, se enfocó en el repositorio http://dblp.uni-trier.de/.

- Habiendo definido el repositorio dblp como sitio de pruebas y ejecución del código, se hicieron ejercicios con el front end del mismo: http://dblp.l3s.de/d2r/snorql.

- Al principio y ante la ausencia de conocimiento en el tema de la web semántica, fue complicado entender cómo elaborar una consulta para acceder a los contenidos de un repositorio, no tenía conocimientos claros al respecto de que herramienta o software usar ni mucho menos como

implementarlo. Fue satisfactorio entender casi al final como operaban las herramientas y ante todo volver a escribir algo de código en esta plataforma como lo es JAVA, espero que el código presentado y las API escritas ayuden a futuros estudiantes a entender de manera más fácil los conceptos de programación de aplicaciones en la web semántica. Debo recomendar sin duda alguna el libro “A developer’s guide to the semantic web, de Liyang Yu”, fue este el documento que me despejo muchas dudas y me hizo comprender un poco más de este interesante tema. Mucho del contenido de este trabajo está basado en el contenido de dicho documento.

- Las API desarrolladas lograron generar un listado completo de las propiedades de una URI dada, estas propiedades incluyen entre otras (dependiendo del contenido de la URI), el título del trabajo, los autores, la fecha de publicación.
- Se ha creado un “meta” motor de búsqueda usando librerías JSoup, que haciendo uso del motor propio del repositorio, recupera los datos de la búsqueda y nos permite desglosar la página resultante para obtener la información requerida

4. GLOSARIO

Basado en: [CITATION eva1 \l 9226].

- FOAF (Friend Of A Friend): Proyecto dentro de la Web Semántica para describir relaciones mediante RDF que puedan ser procesadas fácilmente por máquinas.
- Ontologías: Colecciones de enunciados redactados en un lenguaje, como el RDF, que define el vocabulario de un área mediante un conjunto de términos básicos y las relaciones entre dichos conceptos y especifica reglas lógicas que combinan términos y relaciones. De esta manera, los ordenadores "comprenden" el significado de los datos semánticos de una página de la red siguiendo vínculos con ontologías especificadas. Pueden representar relaciones complejas entre los objetos, e incluyen las reglas y los axiomas que faltan en los tesauros
- OWL: Vocabulario para describir propiedades y clases, tales como relaciones entre clases, igualdad, tipologías de propiedades más complejas, caracterización de propiedades o clases enumeradas.
- RDF: Modelo de datos para los recursos y las relaciones que se puedan establecer entre ellos. Aporta una semántica básica para este modelo de datos que puede representarse mediante XML. Describe la información para que sea procesada por las máquinas.
- RDF Schema: Vocabulario para describir las propiedades y las clases de los recursos RDF, con una semántica para establecer jerarquías de generalización entre dichas propiedades y clases.
- Web 2.0: segunda generación en la historia de la Web basada en comunidades de usuarios y una gama especial de servicios, como las redes sociales, los blogs , los wikis o las folksonomías, que fomentan la colaboración y el intercambio ágil de información entre los usuarios.
- Web 3.0: Neologismo utilizado para describir la evolución en el uso y la interación en la red, pasando por la transformación de la red en una base de datos y el empuje de tecnologías de inteligencia artificial o la Web semántica

- World Wide Web: Sistema de documentos de hipertexto enlazados y accesibles a través de Internet.

- XML: Lenguaje que permite a los usuarios añadir una estructura arbitraria a sus documentos, pero no dice nada acerca del significado de dicha estructura. Aporta la sintaxis superficial para los documentos estructurados, pero sin dotarles de ninguna restricción sobre el significado. En realidad, es un conjunto de tecnologías y lenguajes agrupados, que se presenta como el sustituto de HTML.
- XML Schema: Lenguaje para definir la estructura de los documentos XML.

5. BIBLIOGRAFÍA

Brewer, C. (s.f.). Diagramas.

Consortium, W. W. (s.f.). Obtenido de https://www.w3.org/standards/semanticweb/data

Consortium, W. W. (2016). *W3C*. Obtenido de W3C: https://www.w3.org/2013/data/

evaluci-actualitynews. (s.f.). *evaluci-actualitynews*. Obtenido de evaluci-actualitynews: http://evaluci-actualitynews.blogspot.com.co/

evaluci-actualitynews. (s.f.). *http://evaluci-actualitynews.blogspot.com.co/*. Obtenido de evaluci-actualitynews.

Gracia, L. M. (2012). *Un Poco de JAVA*. Obtenido de https://unpocodejava.wordpress.com/2012/07/27/que-es-apache-jena/

Hendler, T. B. (s.f.). *Capas del nuevo modelo tecnologico*. Obtenido de http://www.nature.com/nature/journal/v410/n6832/full/4101023b0.html

Hipertexto. (s.f.). Obtenido de http://www.hipertexto.info/documentos/owl.htm

Lee, T. B. (s.f.). *Tim Berners-Lee. Semantic Web -XML2000.* Obtenido de http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide11-0.html

Peis, .., Herrera-Viedma, E., & Herrera, Y. H. (s.f.). *Analisis de la Web Semantica.*

yu, L. (2010). *A Developer's Guide to the Semantic Web.* Atlanta: Springer.

6. ANEXO 1

CRONOGRAMA DE ACTIVIDADES

Doble click para abrir el documento.

7. Anexo 2

CÓDIGOS FUENTE

Clase AutorBib:

```
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.NodeIterator;
import org.apache.jena.rdf.model.Property;
import org.apache.jena.rdf.model.RDFNode;
import org.apache.jena.rdf.model.ResIterator;
import org.apache.jena.rdf.model.Resource;
import org.apache.jena.rdf.model.Statement;
import org.apache.jena.rdf.model.StmtIterator;
import org.apache.jena.vocabulary.*;

public class AutorBib
{
	//esta clase hace la captura de la URI en formato RDF y crea un modelo en
memoria
	private CapturURI MismaURI = null;
	public AutorBib(String uri) {
	MismaURI = new CapturURI();
	MismaURI.aggNuevoDom(uri);
}

public void ejecutar()
{
	//este procedimiento hace los llamados, crea el modelo y busca las
siguientes URI
	String URIActual = MismaURI.ProximoDom();
	if (URIActual == null)
	{
		return;
	}

	try
	{

	 //tomando como base la URI en formato RDF, estos pasos capturan el
modelo y lo llevan a memoria
	 Model model = ModelFactory.createDefaultModel();
	 model.read(URIActual, "TURTLE");

	 // captura del conjunto de datos
	 capturarDatos(model,URIActual);

	 // en caso de existir, busca enlaces adicionales determinados por el
ultimo parametro de la funcion.
	 actualizarConjuntoURI(MismaURI,URIActual,model,OWL.sameAs);
```

```
        }
       catch (Exception e)
       {
              System.out.println("Se han presentado errores al acceder a: (" +
URIActual + ")" + e.toString());
       }

       //System.out.println("\n Enlaces complementarios susceptibles de
exploracion: " );
       MismaURI.Mostrartodo();



        // following our nose
       ejecutar();

       }

private void capturarDatos(Model model, String uri)
{
       if ( uri == null )
       {
       return;
       }
       //este procedimiento se encarga de capturar los datos el modelo y
presentarlo por medio de consola
       int factCounter = 0;

       System.out.println("Informacion para: <" + uri + ">:");
       // procedimientos de control y verificacion


       for (StmtIterator si = model.listStatements(); si.hasNext();)
       {

              Statement statement = si.nextStatement();

              if ( uri.equalsIgnoreCase(statement.getSubject().getURI()) != true)
              {
                     factCounter ++;

       System.out.print(statement.getPredicate().getLocalName().toUpperCase() +
": ");
                     if(statement.getObject().isLiteral())
                     {
                            System.out.println(statement.getLiteral().toString());
                     }
                     else
                     {
                            String
text=statement.getObject().asResource().toString();
                            String subtext=text.substring(text.lastIndexOf("/")
+1);
                            System.out.println(subtext);
                     }
```

```
                        if ( factCounter >= 10 )
                        {
                                return;
                        }
                }
        }

private void actualizarConjuntoURI(    CapturURI uriCollection,String uri,Model
model,Property property)
        {
                if ( uri == null )
                {
                        return;
                }

                Resource resource = model.getResource(uri);

                NodeIterator objects = model.listObjectsOfProperty(resource,
property);

                //System.out.println();

                while ( objects.hasNext() )
                {

                        //System.out.print(tmpResource.getURI().toString());

                        RDFNode object = objects.next();

                        if ( object.isResource() )
                        {
                                //System.out.println(uri);
                                Resource tmpResource = (Resource)object;
                                uriCollection.aggNuevoDom(tmpResource.getURI());
                        }
                }

                ResIterator subjects
=model.listSubjectsWithProperty(property,resource);
                while ( subjects.hasNext() ) {
                        Resource subject = subjects.nextResource();
                        uriCollection.aggNuevoDom(subject.getURI());
                }


        }
```

Clase CapturURI:

```
import java.net.URI;
import java.util.HashSet;
import java.util.Stack;
```

```
public class CapturURI {
	//Esta clase permite capturar las URI relacionadas con la direccion
inicial y crear una lista en memoria de estas URIs
	private Stack URI = null;
	 private HashSet dominios = null;

	 public CapturURI()
	 {
		URI = new Stack();
		dominios = new HashSet();
	 }

	 public void aggNuevoDom(String uri)
	 {
		if ( uri == null ) {
			return;
	 }
	 try
	 {
		//Este procedimiento adiciona las nuevas URI a la lista en
memoria, tener en cuenta que si la URI ya fue adcionada, este "if" evita que se
adicione dos veces
		URI thisURI = new URI(uri);
		if ( dominios.contains(thisURI.getHost())    == false )
		{
			dominios.add(thisURI.getHost());
			URI.push(uri);
		}
	 }
	 catch(Exception e) {};
	 }

	 public String ProximoDom()
	 {
		//Salta al proximo dominio
		if ( URI.empty() == true )
		{
			return null;
		}
		return (String)(URI.pop());
	 }

	 public void Mostrartodo()
	 {
		//Muestra en consola todas las URI almacenadas
		for ( int i = 0; i < URI.size(); i ++ )
		{
			System.out.println(URI.elementAt(i).toString());

		}
	 }


}
```

Clase AutorBus:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.URL;
import java.net.URLEncoder;
import java.util.List;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;



public class AutorBus
{
	public static final String startURI ="http://dblp.uni-trier.de/search?
q=";

	public static void main(String[] args)  throws IOException
	{


	Document doc;
	try {


		doc = Jsoup.connect(startURI + args[0]).get();
		Elements links = doc.select("a[href ~=\\.(rdf)]");

			System.out.println("\n Total resultados para busqueda dada: "
+ links.size() + " enlaces relevantes");
			System.out.println("\n Para cada resultado se obtiene la
siguiente información: ");
		for (Element link : links) {
			//System.out.println("\nsrc : " + link.attr("href"));


System.out.println("\n.....................................................
.....................................");
			AutorBib fyn = new AutorBib(link.attr("href"));
			fyn.ejecutar();
		}

	} catch (IOException e) {
		e.printStackTrace();
	}

	}
```

```
}
```