SEMANTIC WEB SERVICES FOR BLOG ARTICLES

by

NIKOLAOS MARIDAKIS

B.Sc. Computer Science, University of Crete, 2013

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF APPLIED INFORMATICS
AND MULTIMEDIA

SCHOOL OF APPLIED TECHNOLOGY

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

2016

Approved by:

Major Professor
Dr. Nikolaos Papadakis

Copyright

Abstract

The appearance of new web technologies and the massive available information online led to an information overload within the web. Considering this revolution, blogs have become a new way of providing and distributing news as a new media. Although there are several arguments about their validity and quality of content, the huge amount of blogs currently available require the usage of novel techniques for the mining, analysis, collection and efficient querying of the available information. To this direction, this work presents a novel platform which provides aggregating, indexing and searching within blog articles. The information is modelled using an RDF/S Ontology named “Blogs Ontology” and is also published as Linked Open Data. In addition, APIs are developed and provided for inserting, updating and searching information. The platform also offers graphical user interfaces (GUIs) for searching and inserting information. During the time that this work was done, the presented platform is the only one currently available publishing blog articles as Linked Open Data and simultaneously providing API’s and GUIs for aggregating, inserting and searching articles.

Key words: Blogs, Semantic Web, Search, Ontologies

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to publicly thank my parents and my five sisters and one brother who helped me get to this level, Nikolaos Papadakis and Theocharis Kondilakis for their valuable scientific assistance, Ioannis Komporakis for contribution to the formatting of the text, the patience and interest in the hearing of my work and all research participants from the third generation of graduate programs M.Sc in Informatics and Multimedia.

Dedication

To my big family, my mentor and friends, I couldn’t have done this without you. Thanks you for all of your support along the way.

Chapter 1 - Introduction

New survey over the last years reveals the rapid rise of Blogs. Searching the internet we find the definition given by the Wikipedia [1], blog is a discussion or informational site published on the World Wide Web and consists of discrete entities named “posts” typically displayed in reverse chronological order.

Nowadays blogs influence public opinion affect reliability and policies as the means news reporting. Blogs can provide important information about the "public opinion" in a wide variety of topics, with basic products, political opinions, and entertainment, through aggregating, monitoring and analysis of information [2]. The blogging started slowly but now it has quickly gained in popularity and now exist more than 248 million blogs (as of February 2014) [1].

1.1 Weblogs

The concept weblog began taking shape in the mid-90s from the community of internet users who maintained websites to describe an innovative practice of renewal of such sites with dated and short records usually contain comments and links to interesting sites something like the "news" sections in modern websites say. Note that the websites at that time, was quite difficult to be refreshed on a regular basis. The term itself was first coined on December 17, 1997 by John Varger [3], owner of website "robot wisdom" and comes from the shortening of words ‘web’ and ‘log’ (term used to describe an individual record dated usually in logbooks), referring precisely to this practice: logging the web, ie keep diary of my navigation on the web.

According to Blood [4], who was somehow early ethnographer of that practice, at the time, maintaining a weblog was tightly related to people already were maintaining sites, since anyway they constituted a collateral activity in already existent websites. Then as this practice began to become popular, in the foreground, began to appear sites created exclusively for use as weblogs and yet again there were people who had relevant experience and knew quite well the html language to the set up from the beginning to the end. In early 1999, there were only 23 websites of this kind (exclusively weblogs), but later on their number increased rapidly. At the same time, as early as October 1998, with the launch of Open Diary[1], the ancestors of social

[1] http://www.opendiary.com/

networking sites began to appear, creating social networking platforms that made use of many elements of what would later they characterized the weblogs as the comments. In early 1999, Peter Merholz[2] announced that from now on will they be called the wee-blogs (in a reference that the community ties that already had been formed) and this is how they came to be referred to as "blogs", their authors as "bloggers" and the practice of maintaining them as "blogging".

1.1.1 Definition

The definitions and conceptualizations of blogs vary as they grow along with the scientific vision of the web as a whole. Below is a wide range of definitions in the theory:

“The blogs are multimedia and easy to use websites that through chronological structure and their potential function as archive abilities, personalized and interconnected filters the web by creating a new online Public sphere that turned the web back into the world” [5]

"Blog is a series of archived records of internet usually characterized by brief texts displayed in reverse chronological order and in general contain hypertext links to other websites the author proposes " [6]

"Blog is a website which is designed to be updated with a chronologically defined linear fashion, just as with a personal calendar but with the difference that the content is aimed at public consumption. Often manageable by a special software, the blogs contain articles or entries are grouped mainly by the date and time published " [7]

"A blog is typically comprised of usually dated records vary in size and frequency of replacement, is organized in reverse dating so that readers can always see the most recent first publication " [8]

"Blog is a calendar based on the web in which the entries are posted in reverse chronological order " [9]

[2] http://www.peterme.com/

“Blog is a website which often contains publications refreshed with the most recent at the top and past appear in reverse chronological order” [10]

The unique feature in which all definitions have been converging It constitutes the definition this work adopts namely that every blog is often renewed site with reverse chronologically arranged entries (ie, from the newest to the oldest).

1.1.2 Anatomy of a blog

In 1999, Cam Barret wrote an article trying to describe the different elements consisting a weblog [5]. He recognizes five aspects that are commonly associated with weblogs: i) daily updates, ii) easy-to-use user interface design, iii) theme (topic), iv) user interaction and v) emerging community.

A blog can be private or used within group, public or controlled accessed, but has one and only specific Internet address (url). Everyone is able to intervene in a greater or lesser extent (depending on the technical skills) in looks and functions however there are three components that make up a typical blog:

Title: the name of the blog, is normally at the top of the page very often replaced or accompanied by a banner that is a picture or a graphical title or a combination of them.

The main part consists of individual, labeled and dated entries (posts / articles) or posts posted by reverse chronological order (the most recent words appear first). Each such record may be multimedia that include text, video, photos, audio and of course the structural architectural element of the web: the hyperlink or link.

The side column (sidebar) usually contains information about the blog in logic menu (just like the category “about” the typical websites), for the administrators, the history to which you can refer to previous records listed chronologically categorized as a links (blogroll) to other sites and other blogs that mainly proposed as valuable and that a degree of their choice determines deliberately the identity of the reference blog [6] [7].

1.1.3 Common Blogging Phrases

Like most new technologies, the blogosphere (blogging world) is full of new words, terms, and slang used to describe blogs and the act of blogging. To get you started on knowing the lingo, here are some of the many blog-related terms you'll find written online today.

- blog: Short for Web log, a blog is a Web page that serves as a publicly accessible personal journal for an individual. Typically updated daily, blogs often reflect the personality of the author.
- blogger: A person who blogs.
- blogging: The act of writing or updating your blog.
- blogosphere: Meaning all blogs, it is an expression used to describe the 'world of blogs'.
- blogroll: Found on blogs it is a list of links to other blogs and Web sites that the blog author commonly references or is affiliated with. Blogrolls help blog authors to establish and build upon a their blogger community.
- blogsnob: (1) A slang term used to describe a blogger who doesn't respond to blog comments left by people outside his or her own circle of blogger friends.

  (2) Written as BlogSnob, a free advertising exchange for blogs and personal sites.
- b-blog: Short for business blog, a blog used by a business to promote itself.
- klog: Short for knowledge blog, klog is a type of blog usually used as an internal / Intranet blog that is not accessible to the general public and that serves as a knowledge management system. The term klog is also being used to describe a blog that is technical content oriented.
- moblog: Acronym used to combine the terms "mobile" and "Web log". Where a Web log (also called a blog) is a Web page that serves as a publicly accessible personal journal for an individual, a moblog is a blog which has been posted to the Internet from a mobile device such as a mobile phone or PDA.
- tagging: Commonly used in blogs, site authors attach keyword descriptions (called tags) to identify images or text within their site as a categories or topic. Web pages and blogs with identical tags can then be linked together allowing users to search for similar or related content. If the tags are made public, online pages that act as a Web-based bookmark service are able to index them. tags can be created using words, acronyms or numbers. Tags are also called tagging, blog tagging, folksonomies (short for folks and taxonomy), or social bookmarking.
- Blog and Ping: An online marketing term applied to a system that utilizes blogs and pings (short for pingback) to deliver content and /or sites for indexing in search engines with the ultimate aim of profit. Also called blog ping.

- vlog: Short for video blog, it is the term used to describe a blog that includes or consists of video clips. Typically updated daily (or with regular frequency) vlogs often reflect the personality or cause of the author. Also called vog.

1.2 Semantic Web Blog Search Engines

"The Semantic Web is the representation of data on the World Wide Web. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming." – W3C Semantic Web. The Semantic Web is a framework that allows publishing, sharing, and reusing data and knowledge on the Web and across applications, enterprises, and community boundaries [14]. Currently, the Semantic Web, consisting of Semantic Web documents typically encoded in the languages RDF and OWL, is essentially a Web universe parallel to the Web of HTML documents [15]. Knowledge encoded in Semantic Web languages such as RDF differs from both the largely unstructured free text found on most Web pages and the highly structured information found in databases. Such semi-structured information requires using a combination of techniques for effective indexing and retrieval. RDF and the Web Ontology Language (OWL) which are ontology based procedures or representing knowledge on the Web, introduce aspects beyond those used in ordinary XML, allowing users to define terms (for example, classes and properties), express relationships among them, and assert constraints and axioms that hold for well-formed data. An application of the emerging Semantic Web is a Semantic Web search engine which searches the Semantic Web documents against a user query for accurate results. Our work uses RDF encoded Semantic Web documents which are searched in response to a user query for exact results.

Blog search engines are search engines for the blogosphere. They can only index and provide search results from weblogs. There are several examples of blog search engines like Google Blog Search and Technorati, that make use of the traditional ways of searching and indexing. In our work, we focus on Semantic Web search engines that enable more efficient searching and aggregating entries within the blogosphere.

Accessing the right blog entries on the Web is often a difficult task. While searching the Web for information, users usually face huge amounts of irrelevant results that mislead them into losing the way to the initial target. According to Guha et al. [8], semantic search attempts to augment and improve traditional search results (based on Information Retrieval technology) by using data from the Semantic Web. Traditional Information Retrieval (IR) technology is based almost purely on the occurrence of words in documents. Search engines like Google, augment this in the context of the Web with information about the hyperlink structure of the Web. The availability of large amounts of structured, machine understandable information about a wide range of objects on the Semantic Web offers some opportunities for improving on traditional search.

Before getting into the details of how the Semantic Web can contribute to search, we need to distinguish between two very different kinds of searches.

- **Navigational Searches:** In this type of searches, the user enters within the appropriate field of the search engine a phrase or combination of characters or words which expects to find in the results. There is no straight, reasonable interpretation of these words as denoting a concept. In such cases, the user is using the search engine, to serve him as a navigation tool to a specific document he is interested into. In this work, we are not interested in this class of searches and we will not focus into them.
- **Research Searches**: In many other cases, the user provides the search engine with a phrase which is intended to denote an object about which the user is trying to gather/research information. There is no particular document which the user knows about that s/he is trying to get to. Rather, the user is trying to locate a number of documents which together will give him/her the information s/he is trying to find. This is the class of searches we are interested in, and in our case, the documents are blog entries.

In a traditional programming environment the developers would write queries based on knowledge of the search criteria, data structure (e.g. relational database) and the query language. A truly optimized semantic search however requires a search engine that can write itself queries to be fed back into the query engine. This means that a semantic search tool needs to both link meanings to search keywords and be able to efficiently retrieve information. [9].

Given the fact that the trend is all waiting to go on, the techniques are very important in order to make the collection, access and effective questioning blogs. For example there are several engines like Regator [10] and the IceRocket [11] trying to show and allow search in different blog posts. However, although some of the above search engines provide current information on both popular searches and tags used to categorize posting blog, they lack a systematic approach to increase the available semantic data and expose as Linked Open Data.

The purpose of the Semantic Web is to create data with meaning effectively, making it easier to exchange data between systems and to support the integration of information from various sources [12]. To this direction, and to make this possible, relationships between the data need to be available is the basis of Linked Data - a group practice for publishing structured data on the web. The contributions of this paper are in this direction and are the following:

- A new ontology, called Blogs Ontology to represent all relevant information and relationships between blog posts. The ontology links existing relevant ontologies and expand by adding the social dimension and extra-potential properties.
- A repository using existing Semantic Web technologies to publish all information as Linked Open Data, which allows for the import, aggregation and integration of new information.
- Even though this repository provides native query functionality through a SPARQL endpoint move further to provide our own APIs that use the REST services. There are two versions of the API that are available, namely a) the legacy API that allows the exchange of JSON data and b) the linked data API using JSON-LD.
- Besides allowing users to interact using programmatically with the repository, a nice graphical user interface (GUI) is provided to the owners Blog, to acquaint all relative information.
- We provide a search engine allows searching the content index. The search engine is more advanced and beyond the simple text search in order to a) enable the faceted search in different categories of information b) make semantic annotations of text available for the classification of returned results.
- The evaluation effected displayed the benefits of our approach and the great advantages gained.

During the examination of the current literature, it is noted that Blogsearch is a unique platform that combines an ontology which is able to represent all the information in its field, with the new APIs that enable the publication of information in a central repository and provide an advanced engine search with a user-friendly GUI interface to enable insertion and search available information. The general notion is that the individual sites will use the API to propel their sites to our central repository. Then they will be available as Linked Open Data ready to be consumed either programmatically or by Blogsearch our semantic search engine.

1.3 Research goals and questions

Semantic Web and Semantic Search are two different terms that share a common word, ‘semantic’. Fortunately, understanding semantics in relation to the web is actually quite simple, and for many these is already a part of your daily routine. It isn’t a new concept, just one that has recently gained attention.

But, how the terms Semantic Web and Semantic Search differ and why it matters? Being able to understand how these terms differ is important because it can help you better understand how search works and how you can make sure your information is getting in front of a relevant audience. Below explains the differences between these two terms that are often mistakenly meshed into one:

In Semantic Web, The whole idea is to teach searchers about understanding the whole content of data as opposed to just the structure of search engines like Google. The Semantic Web is a set of technologies for representing, storing, and querying information. Although these technologies can be used to store textual data, they typically are used to store smaller bits of data.

Essentially, the semantic web will include things like numbers and dates in order to be able to answer a very complex question. Semantic search focuses on the text, but the semantic web focuses on pulling data from multiple sources and multiple formats.

Another way to look at it is that the semantic web is not going to store one page as just one page. Instead, it works to take each small detail on the page and retrieve those small details off every page to find one cohesive answer.

On the other hand, Semantic Search, offers more relevant results without limiting searches to just keywords (traditional Google search would be called “keyword search” as opposed to a semantic search). The below definition puts it into simple terms:

“Semantic search is the process of typing something into a search engine and getting more results than just those that feature the exact keyword you typed into the search box.”

Semantic search will take into account the context and meaning of your search terms. It’s about understanding the assumptions that the searcher is making when typing in that search query.

In this direction, this work aims to answer the following question:

How does Semantic Web Search in blogs perform in comparison to existing conventional search methods? To answer this question the following sub questions need to be answered.

1. What information should a blog entry contain in order to be easily retrieved?

2. How to design and build a Semantic Web Search engine for blogs?

1.4 Structure

This master thesis is structured in five chapters as follows:

Chapter 2, reviews and reports the related work.

Chapter 3, the implementation of the whole work is presented, analyzing the architecture, the different components of the platform and the tools and technologies used for its construction.

Chapter 4, presents usability evaluation results and finally

Chapter 5, concludes this work and presents directions for further work.

Chapter 2 - Related Work

To model information on news, there are several approaches by this moment. In journalism, for example, the BBC channel has recently published a new ontology named the History News [13]. This ontology consists of broad categories that describe news from various publishers and interconnected writers, publishers, etc. Nevertheless, the ontology is not primarily targeted to articles blog. But then, SiOC [14] is a collective effort to interconnect online communities, offering a general ontology for describing the information resides in message boards, forums, etc. If their work is focused on creating new connections between content and community objects ontology does not consider the explosion of existing social media platforms and cannot be adequately modeled current interactions social media. There is another approach with similar problems is Schema.Org offers shapes for modeling Blogs and blog postings. Our approach is based on both Schema.Org and FOAF ontology connection and expansion, aiming to additional social dimension of the media and their enrichment with more categories and properties.

There were many attempts to model approach for relevant information as much as there were attempts to provide the blog search engines but stopped like Bloglines [15], the BlogScope [2] and Technorati [16]. These machines have a problem that they considered the blogs as static web pages using simple text search (and corresponding classification systems) ignoring personal, time, and social dimensions of these articles. In addition to the above machines there are many more that exist today, such as Regator [10] and Ice-Rocket [11]. These systems can detect, analyze and to index the blogs. Nevertheless, the results provided are not semantically high, the metadata provided is very little and the statement by the lack of semantic technologies is not possible.

Additionally, the increasing use of micro-blogging platforms like Twitter, has created a huge number of tools that try to index and to search the content which created there. Nevertheless, both the focus as much and methods (get smaller message size, etc.) that there are different and our solution can be considered supplementary.

On the other hand, there is the Swoogle [17] which is a popular semantic search engine that detects Semantic Web documents and elicits metadata which found. Their work focuses on give results on appropriate ontologies based on the search term, or to characterize the Semantic Web with the collection of interconnected relations document. Their approach is different from standard search engines like semantic analyze documents instead of simple web sites, displaying the results in RDF format. In the same way, Sindice [18] crawls and indexes the web RDF collection of documents and provides an API so that developers can consolidate data in applications of third manufacturers. Nevertheless, the approach our indexing and aggregating blog posts them then presented as RDF / S. Furthermore the data APIs there are all the necessary GUIs available for active users to insert and search for related information.

In another paper, Page et al. [19] sought to identify if the REST and Linked Data should be complementary architectural style or have cross purposes. In their paper, they have concluded that the two have similarities and differences. This will result founding various authorities in order to better serve the region's development-oriented applications together. The Blogsearch platform concurs with these principles for reconciling RESTful API to Linked Data technologies.

2.1 Searching in Blogs

In the last decade, the exponential rise in the number of blogs has created a need for effective access and retrieval services. Today, there is a broad range of search and discovery tools for blogs, offered by a variety of players; some focus exclusively on accessing the blogs, while well-established web search engines such as Google[3], Yahoo![4] and Bing[5] offer specialized blog services or provide, among others, results from blogs also.

Before we analyze the implementation of the system, we have to review the modern blog search engines.

2.1.1 IceRocket

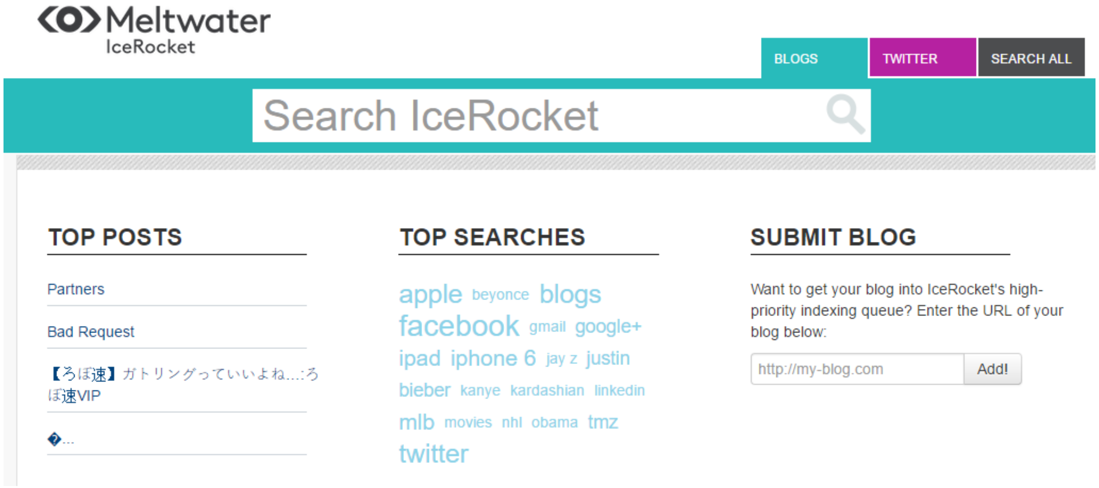
IceRocket is an Internet search engine which specializes in real-time search. Based in Dallas, Texas, it launched in 2004 hoping to market itself solely through “word of mouth”. It was originally launched with features designed to facilitate web searches on mobile devices such as

[3] https://www.google.com/

[4] https://yahoosearch.tumblr.com/

[5] https://blogs.bing.com/search/

PDA much easier, allowing users, for example, to email a query to the engine and receive their results back in response. In August 2011, it was announced that IceRocket had been acquired by the Meltwater Group.



Want to get your blog into IceRocket's high-priority indexing queue? Enter the URL of your blog below:

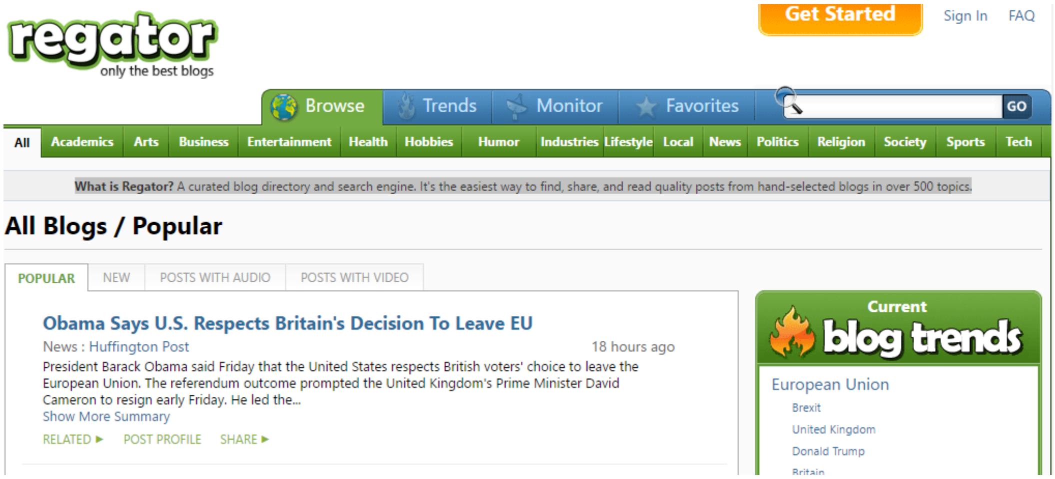
Picture 1: IceRocket Homepage

IceRocket is generally for blog searches but has expanded into searching the popular social networking websites Twitter and Facebook as well as allowing searching of news and the world wide web. “Big Buzz” feature allows users to search Blogs, Tweets, news, images etc. all from one page. Finally, it provides an API that it licenses to social media monitoring firms as well as PR agencies.

2.1.2 Regator

“What is Regator? A curated blog directory and search engine. It's the easiest way to find, share, and read quality posts from hand-selected blogs in over 500 topics.”. This is what is written in the top of the home page[6] of Regator blog search engine. It was founded in 2007 by Scott Lockhart, Chris Turner, and Kimberly Turner and it got in production in August 2008 [20]. It provides an API platform that allows retrieving and tracking detailed trend and analyzing text.

[6] http://regator.com/



What is Regator? A curated blog directory and search engine. It's the easiest way to find, share, and read quality posts from hand-selected blogs in over 500 topics.

All Blogs / Popular

Obama Says U.S. Respects Britain's Decision To Leave EU

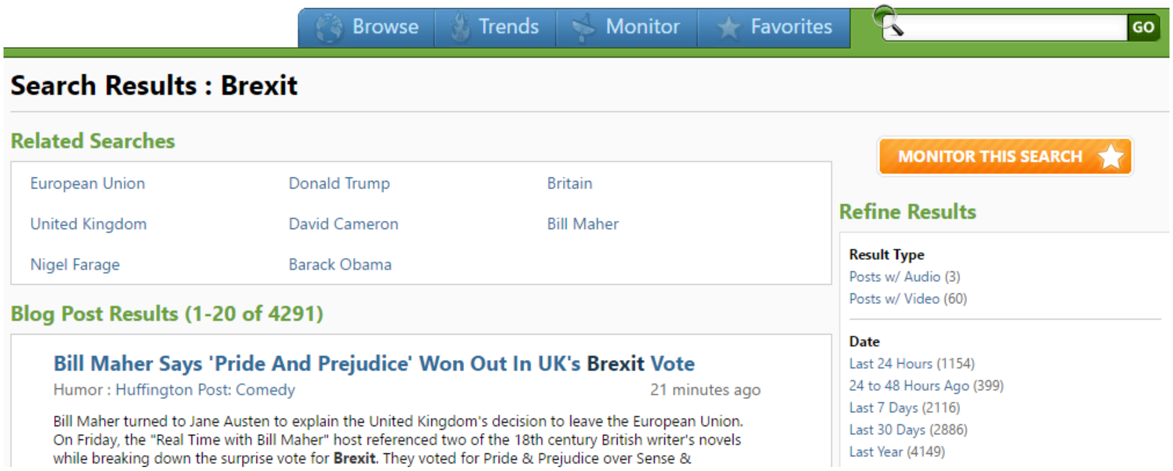
News : Huffington Post 18 hours ago

President Barack Obama said Friday that the United States respects British voters' choice to leave the European Union. The referendum outcome prompted the United Kingdom's Prime Minister David Cameron to resign early Friday. He led the...

European Union
Brexit
United Kingdom
Donald Trump
Britain

Picture 2: Regator homepage

In the homepage of Regator, the visitor is exposed to the most popular blog posts by default, while he is able to filter them by date (new posts), or posts that include audio or video. It is also feasible to filter posts by trends, choose topics such as academics, arts, business, entertainment, health, humor, news etc.



Search Results : Brexit

Related Searches

European Union	Donald Trump	Britain
United Kingdom	David Cameron	Bill Maher
Nigel Farage	Barack Obama	

Blog Post Results (1-20 of 4291)

Bill Maher Says 'Pride And Prejudice' Won Out In UK's Brexit Vote

Humor : Huffington Post: Comedy 21 minutes ago

Bill Maher turned to Jane Austen to explain the United Kingdom's decision to leave the European Union. On Friday, the "Real Time with Bill Maher" host referenced two of the 18th century British writer's novels while breaking down the surprise vote for **Brexit**. They voted for Pride & Prejudice over Sense &

Refine Results

Result Type
Posts w/ Audio (3)
Posts w/ Video (60)

Date
Last 24 Hours (1154)
24 to 48 Hours Ago (399)
Last 7 Days (2116)
Last 30 Days (2886)
Last Year (4149)

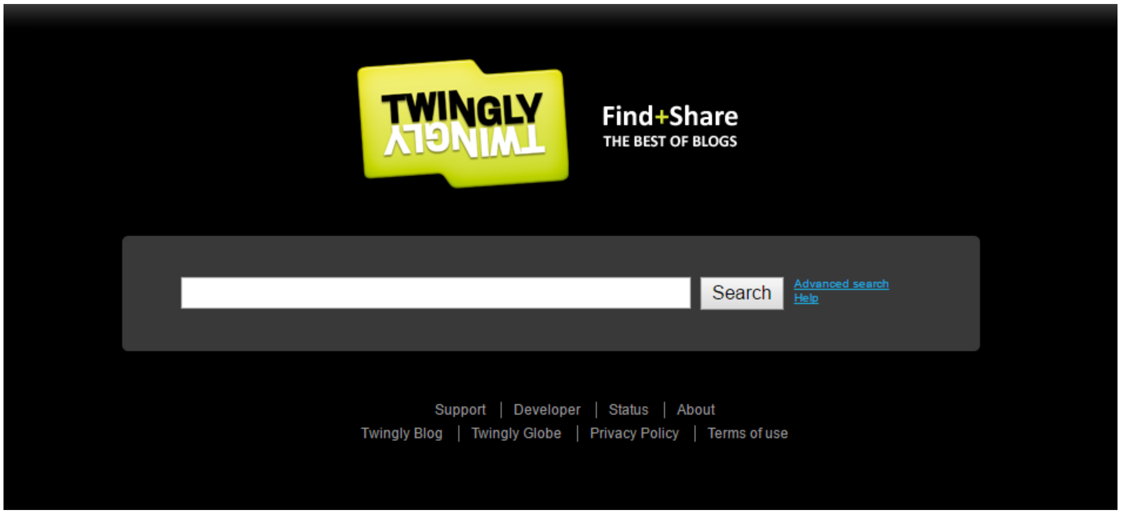
Picture 3: Searching in Regator

Searching in Regator, offers search result by string matching (conventional search), while it also offers tools for searching in related terms, refine results by type or by date and a mechanism to monitor the search by sending updates around it periodically.

2.1.3 Twingly

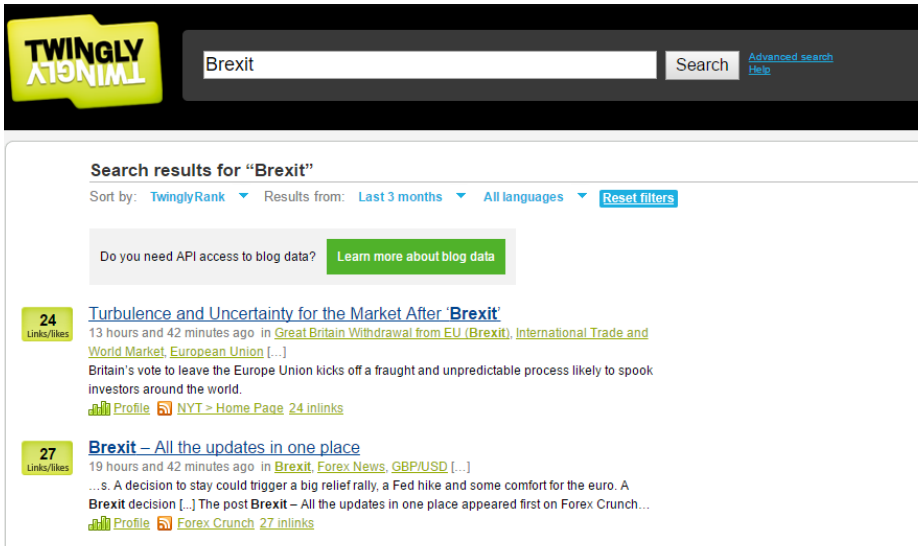
Twingly is an advanced blog search service that offers instant results in XML, history up to 12 months, search across one or multiple languages simultaneously, querys with a custom query language, and search blog posts by tag. According to their blog [21], Twingly is *"a* *leading supplier of global blog data, monitoring more than 6 million active blogs.* Its vision is to provide solutions for media and e-Commerce companies to improve their relations with bloggers, to have them blog more about their content, brands and products. The solutions are based on their blog data mining where so far have indexed more than 85 million blogs.

Twingly offers a traditional blog search engine[7] that the user is able to find results that exist within the blogosphere.


TWINGLY
Find+Share
THE BEST OF BLOGS
Search
Advanced search
Help
Support | Developer | Status | About
Twingly Blog | Twingly Globe | Privacy Policy | Terms of use

Picture 4: Twingly Search Engine

As it is shown in Picture 5, the results of the search can be filtered by rank, date and languages.



Search results for "Brexit"

Sort by: TwinglyRank Results from: Last 3 months All languages Reset filters

Do you need API access to blog data? Learn more about blog data

24 Links/likes

Turbulence and Uncertainty for the Market After '**Brexit**'

13 hours and 42 minutes ago in Great Britain Withdrawal from EU (**Brexit**), International Trade and World Market, European Union [...]

Britain's vote to leave the Europe Union kicks off a fraught and unpredictable process likely to spook investors around the world.

Profile NYT > Home Page 24 inlinks

27 Links/likes

Brexit – All the updates in one place

19 hours and 42 minutes ago in **Brexit**, Forex News, GBP/USD [...]

...s. A decision to stay could trigger a big relief rally, a Fed hike and some comfort for the euro. A **Brexit** decision [...] The post **Brexit** – All the updates in one place appeared first on Forex Crunch...

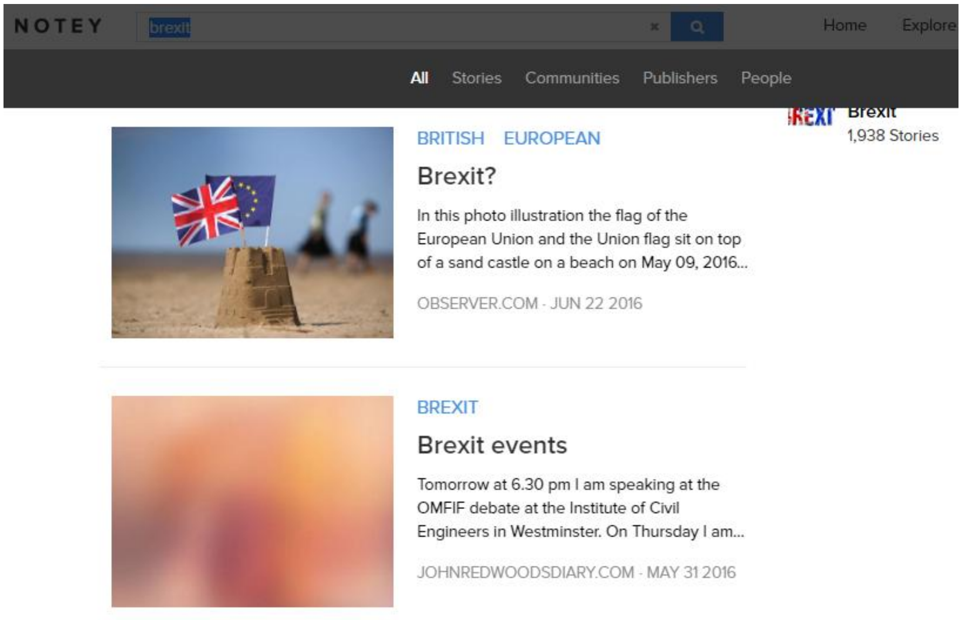
Profile Forex Crunch 27 inlinks

Picture 5: Twingly search results

[7] https://www.twingly.com/search

2.1.4 Notey

Another blog search engine is Notey. It offers search between different interests such us trip planning, a TV show or a favorite football team. The difference between the other blog search engines that act as simple RSS readers, Notey relies on word of mouth to find interesting blogs. It started as a traveling assistance search engine, but it proceed to streamline the process and expand to many other topics (around 500,000 according to their website) and now is the largest blog search and discovery platform.



BRITISH EUROPEAN

Brexit?

In this photo illustration the flag of the European Union and the Union flag sit on top of a sand castle on a beach on May 09, 2016...

OBSERVER.COM · JUN 22 2016

BREXIT

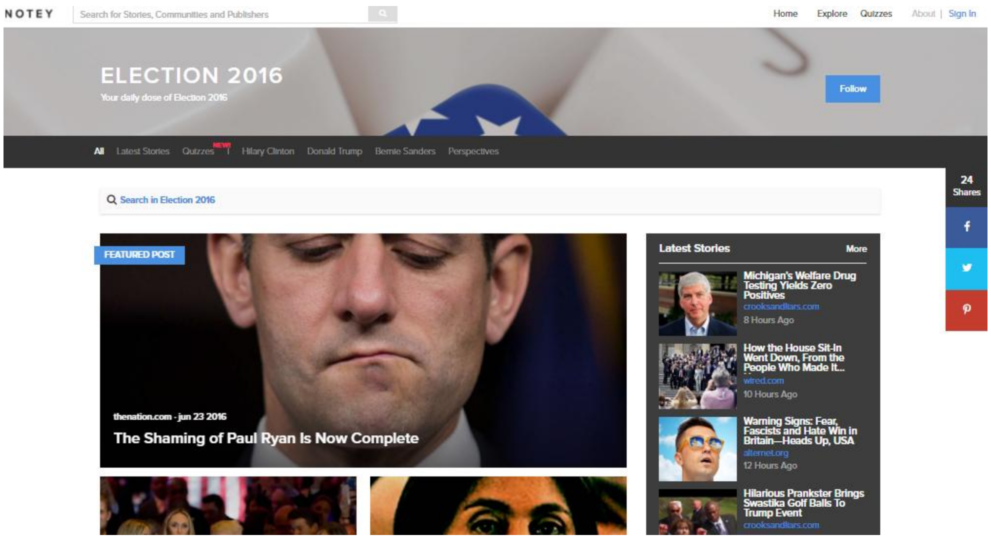
Brexit events

Tomorrow at 6.30 pm I am speaking at the OMFIF debate at the Institute of Civil Engineers in Westminster. On Thursday I am...

JOHNREDWOODSDIARY.COM · MAY 31 2016

Picture 6: Notey search results

In the home page, the top stories of the day are provided in the main part of the page, while the most popular categories are presented on the top for rapid navigation. In the page of each category, there are featured posts, must read stories, latest additions of the blogosphere, and community tools such as friends that follow the same topic. Also, the top publishers of the content are listed.



ELECTION 2016

Your daily dose of Election 2016

thenation.com - jun 23 2016

The Shaming of Paul Ryan Is Now Complete

Michigan's Welfare Drug Testing Yields Zero Positives
crooksandliars.com
8 Hours Ago

How the House Sit-In Went Down, From the People Who Made It...
wired.com
10 Hours Ago

Warning Signs: Fear, Fascists and Hate Win in Britain—Heads Up, USA
alternet.org
12 Hours Ago

Hilarious Prankster Brings Swastika Golf Balls To Trump Event
crooksandliars.com

Picture 7: Notey category page

Chapter 3 - System Architecture and Use Cases

For the design of the platform a three - tier architecture was used, shown in Figure 1.

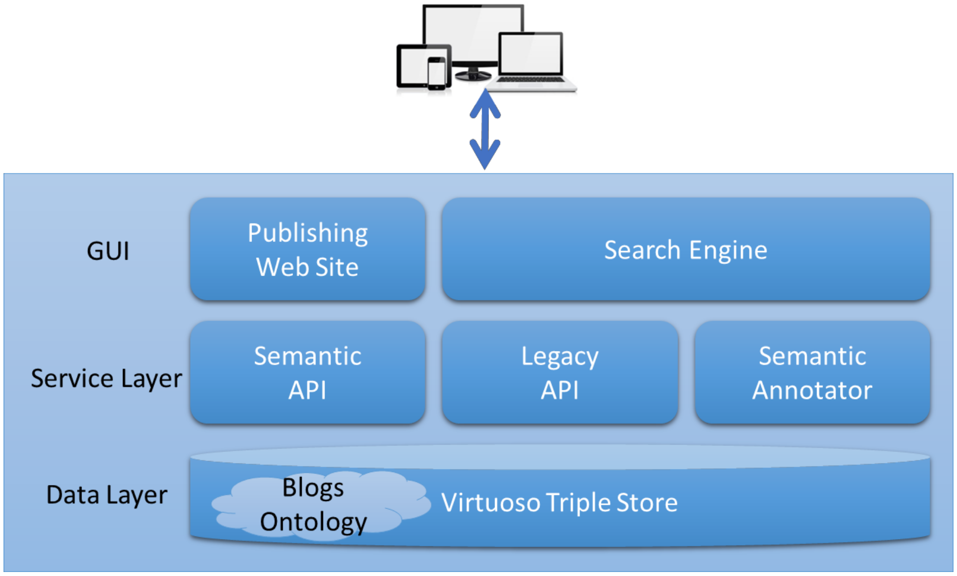


Figure 1: Platform Architecture

The three layers are the following:

- The GUI Layer: This layer consists of a search engine and a publishing client that allow the manipulation of the information in relevance.
- The API Layer: Is the middle layer, which is composed of web services required to communicate with the Linked Data Layer and a SPARQL endpoint which provides access to all information. External tools may use this layer to insert, update or search for relevant information using the appropriate HTTP Request calls.
- The Linked Data Layer: In the lowest layer and the backbone of the platform, the Virtuoso triple store is used to store and provide access to the data. Virtuoso Universal Server is a middleware database hybrid engine that combines functionalities of typical RDMS and file server facilities. In our work the open source version of virtuoso was used. Among others, Virtuoso and OWLIM were considered [22] for the semantic backbone of our platform. Finally, Virtuoso was selected since OWLIM discontinued the open source version (it is now called GraphDB).

We have developed an ontology called Blogs ontology to form the information related to the posts. Moreover, it is noted that the web services implemented by following REST [23], an architectural style for building large-scale distributed hypermedia systems; evolving into a model for the development of web applications. REST based information over to abstract entities, declared as "resources", to be manipulated by a single interface via a URI. The RESTful web services have been used because of their advantages compared to standard RPC - style services (SOAP, WSDL, etc.) such as scalability, flexibility and safety [24], a rather natural response to prohibitive complexity of development and describes these services. There is a detailed description of the ontology below, the GUI and the API layers.

3.1 Ontology

Ontology is a typical, clear description of a domain of interest [25]. In our case we produced an RDF / S ontology which describes blogs and their posts by analyzing a variety of blog sites available on the Internet. The requirements for our ontology were to create a robust ontology covering the use, the users and the information available at various web blog sites. Is based on a detailed analysis of the available information on various blog sites and the above conditions, the following main categories of generic produced to store various data in the blog:

- Blog: This class describes individual blog sites. A blog site can have a name, a thumbnail, and a url. In addition a blog can have many members and many articles.
- Article: An article is an individual post in a blog. It has an author, comments, dates for the creation and the update of the content, a rating, a thumbnail, a title and a url.
- Author: An author is a person who produced a blog article. The person can have a name, an email, a nickname, a thumbnail etc.
- Comment: Finally, a post might have many comments, produced on specific dates by specific persons.

Apart from the main resource types we saw above, necessary attributes introduced to interconnect individual classes and literals. The main categories and their properties are shown in

Figure 2, while the ontology can be found online. To create the ontology used the Protégé tool, which is free and open-source ontology editor.

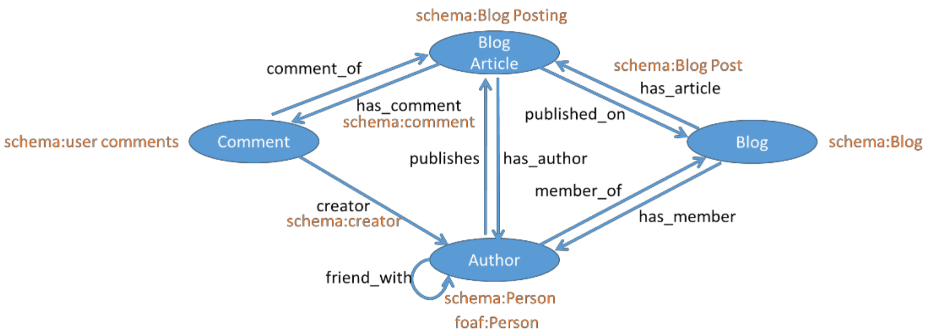


Figure 2: The core classes and properties of the BLOGS Ontology and equivalences with the FOAF and Schema.Org ontologies.

The developed ontology is also connected to the FOAF and ontologies Schema.Org to extend it by adding more metadata about the social dimension and properties and annotations on Articles blog, comments and authors. In Fig. 2, the matches of the main categories of BLOGS Ontology with FOAF and Schema.Org ontologies are shown (equivalences shown in orange).

3.2 The Service Layer

In the service layer, the legacy and the linked data APIs have been implemented and the semantic annotator web service too. The APIs allow an external application to introduce new information to our website and to allow programmatic update and search. On the other hand, the semantic annotator allows searching beyond simple text matching also utilizing ontology terms.

3.2.1 API

As it is presented in the work, Blogsearch API allows users to have direct access to semantic data to the blog via the Hyper-text Transfer Protocol (HTTP), and provides an efficient way for users to search for programming article information. Among other authorities is an important step in any RESTful design is the identification of resources enforced basis of the data model. The Blogsearch API is designed in a Resource-Oriented architecture and includes four types of resource: Blogs, Authors, Articles and Comments. As the article informs [23], each object must be represented, are assigned to a URIs, and should be asked in a uniform matter. In our work we have established a base URL for each of the types of our resources, ie http: // [...] /

rest / blogs, http: // [...] / rest / articles etc. followed by an identifier allowing simple and easy access. On our web services someone can find information online[8].

Broadly speaking, RESTful APIs require the use of HTTP GET method for read-only queries and HTTP PUT method to create new resources. Most of these actions return structured data representing the object or effect of an act by the object. For encoding structured data, JSON was selected considering that it is the current default and popular option for web service interfaces and maintain the analysis support in many programming languages.

Taking into account that we handle semantic data, each request for data retrieval or storage should be analyzed from triple JSON format display or conversely in case of a request for data storage. In our project we have created mechanisms to effectively analyze the required data, using Jena API. Included in other features, Jena API provides methods and support the creation or mapping of RDF graphs to Java Objects for data processing. The Blogsearch API uses Jena, not only for the analysis but additionally for communicating with the Linked Data Layer and particular Virtuoso triple store.

Over and above the legacy API we also create a more advanced version of the API using JSON-LD, instead of a simple JSON. JSON-LD [33] is a lightweight Linked Data format. It is easy to use for people to read and write. Based on pre-existing successful JSON format and provides a way to help JSON data cooperate to Web-scale. JSON-LD is an ideal data format for programming environments and all implemented web services can also be found online connection. A simple example of JSON-LD of a single article is shown in Figure 3.

Figure 3: Example JSON-LD

```
  [
    {
        "article_date_created":"2015-01-19T12:00:00Z",
        "article_uuid":"4abad6f9-d696-4556-8ff3-cb84049d780e",
        "author":"b57e067a-59f0-4668-867f-b5b8675b0940",
        "author_nickname":"andy3",
        "article_date_updated":"2015-01-19T12:00:00Z",
        "article_thumb_url":"http:\/\/i.kinja-img.com\/gawker-
media\/image\/upload\/s--aOU9-C5_--
\/c_fit,fl_progressive,q_80,w_636\/dm3vjbsclmbbywztlm2n.jpg",
        "blog_title":"LifeHacker",
```

[8] http://83.212.124.52/?p=ontology

```
      "article_url":"http:\/\/lifehacker.com\/ask-an-expert-all-about- custom-
pc-building-1680403516",
      "blog":"cff98839-a7e1-4402-b4b3-fc10a8cfff92ad4fa3f-fe40-44ff-
954d-2ee3917ac2cb",
      "@context":{
         "article_date_created":"sw:article_date_created",
         "sw":"http:\/\/sw.owncloud.gr\/articles#",
         "article_uuid":"sw:article_uuid",
         "author":{ "@type":"@id",
            "@id":"sw:author"
         }, "author_nickname":"sw:author_nickname",
         "article_date_updated":"sw:article_date_updated",
         "article_thumb_url":"sw:article_thump_url",
         "blog_title":"sw:blog_name",
         "article_url":"sw:article_url",
         "blog":{ "@type":"@id",
            "@id":"sw:blog"
         }, "article_categories":"sw:article_categories",
         "article_title":"sw:article_title",
         "article_comments_count":"sw:has_comment",
         "article_rating":"sw:article_rating",
         "article_body":"sw:article_body",
         "article_tags":"sw:article_tags"
      },
      "article_categories":"Technology",
      "article_title":"Ask an Expert: All About Custom PC Building",
      "article_comments_count":2,
      "article_rating":2,
      "@id":"article_4abad6f9-d696-4556-8ff3-cb84049d780e",
      "article_body":"No one takes….", "article_tags":"Technology,
      PC"
   }
]
```

3.2.2 Semantic Annotator

To increase the quality of search, unstructured text documents go through a semantic lifting and indexing phase. During this phase, the Semantic Annotator Service called at regular intervals, once each week, to comment on the recently introduced documents on the internet in terms from the BLOGS, Schema.Org and FOAF ontologies. The entire process is asynchronous since the time required to comment each article can be significant. Each document is analyzed using

Natural Language Processing (NLP) algorithms for tokenization, lemmatization and part of speech. [26] The Semantic NLP service is based on the XMedLan tool developed by XEROX company [27] and fit and exports ontology terms, concepts and semantic types of the aforementioned ontologies. All the extracted terms are stored in Virtuoso Triple Store and they are used to match documents to queries. To attain the purposes of this match the commentator is used at the runtime, and comment input queries in terms of ontology and then the cosine similarity of these comments with the article annotations is calculated as we will be seen below.

3.3 The Graphical User Interface Layer

The User Interface is composed by two separate application. First, the Search Engine BLOGS and second, the Publishing Wizard. Both of the above applications have been developed using of the Java language and are based on the API developed.

3.3.1 The BlogSearch Engine

The traditional search engines usually query keywords based on the webpages where the search term appears to have detected and indexed in accordance with the ranking. There are however, semantic search engines looking for ontological concepts transporting the meaning of the term. Until now, many semantic search engines have been created [28] [29] [30], but then either focus on the use of annotations to find the documents, in order to formulate queries with regard ontologies, or the lack of a user friendly interface.

Our search engine:

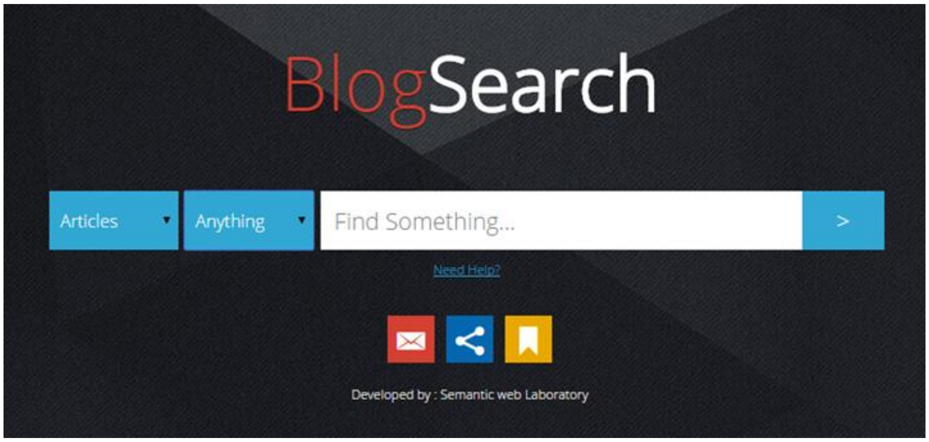
a) provides a nice, useful interface GUI and allows faceted search using the main categories of ontology (articles, blogs, authors) and their properties (anything, post’s title, text, keyword, blog’s title, author’s nickname, date, rating)

b) is based potentials questions formed by user interaction generated by the system;

c) exploits semantic annotations to go beyond the simple string matching

d) an advanced level algorithm used to rank results

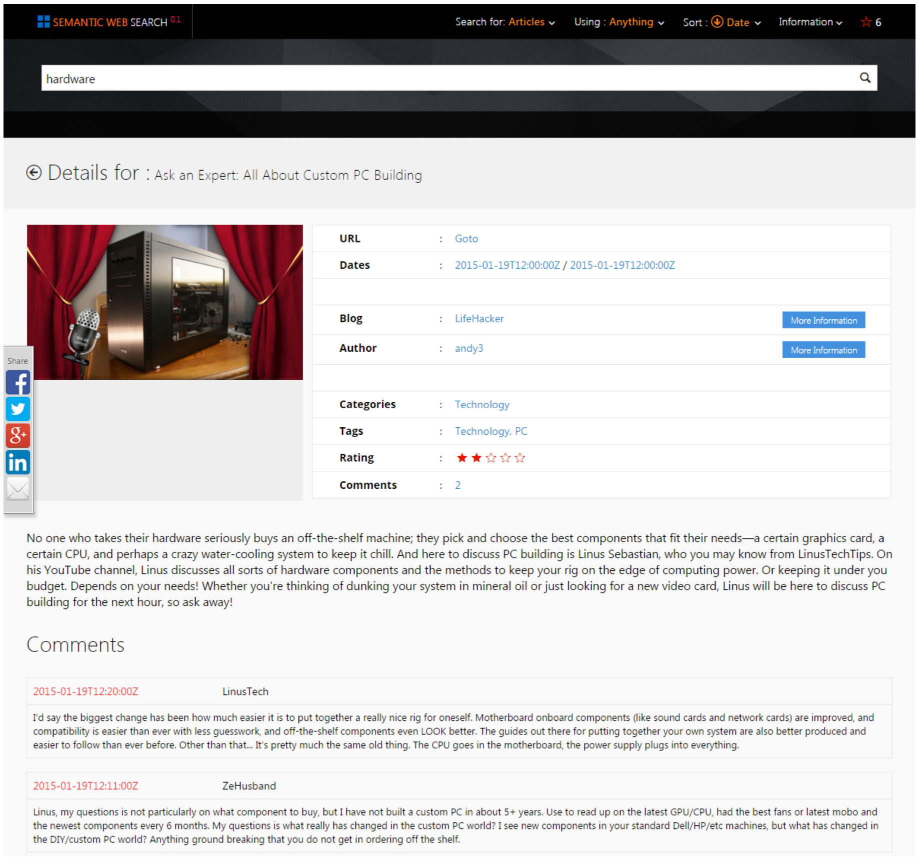
Figure 4: The BlogSearch engine



Especially, a user of the platform can search for a term in whole ontology or execute a progressive, for example, search in a specific domain of the ontology. The user has the possibility of searching for articles, sites or authors using a variety set of properties. Additionally, the results can be ordered in many different ways. A beta version of our search engine is available online[9]. The landing page of our search engine is shown in Figure 4, and a screenshot of reading an article shown in Figure 5. Users may like the articles, to reflect, to read or to comment on them in the search engine conservation, however connections to their original source. Moreover, articles can be shared using the social networks like Facebook, Twitter, LinkedIn, and Google+ or sent directly from our website. The content comes from Virtuoso triple store using appropriately the available APIs.

Figure 5: Reading an Article

[9] http://83.212.124.52/



hardware

Details for : Ask an Expert: All About Custom PC Building

URL	:	Goto
Dates	:	2015-01-19T12:00:00Z / 2015-01-19T12:00:00Z
Blog	:	LifeHacker
Author	:	andy3
Categories	:	Technology
Tags	:	Technology, PC
Rating	:	★★☆☆☆
Comments	:	2

No one who takes their hardware seriously buys an off-the-shelf machine; they pick and choose the best components that fit their needs—a certain graphics card, a certain CPU, and perhaps a crazy water-cooling system to keep it chill. And here to discuss PC building is Linus Sebastian, who you may know from LinusTechTips. On his YouTube channel, Linus discusses all sorts of hardware components and the methods to keep your rig on the edge of computing power. Or keeping it under you budget. Depends on your needs! Whether you're thinking of dunking your system in mineral oil or just looking for a new video card, Linus will be here to discuss PC building for the next hour, so ask away!

Comments

2015-01-19T12:20:00Z LinusTech

I'd say the biggest change has been how much easier it is to put together a really nice rig for oneself. Motherboard onboard components (like sound cards and network cards) are improved, and compatibility is easier than ever with less guesswork, and off-the-shelf components even LOOK better. The guides out there for putting together your own system are also better produced and easier to follow than ever before. Other than that... It's pretty much the same old thing. The CPU goes in the motherboard, the power supply plugs into everything.

2015-01-19T12:11:00Z ZeHusband

Linus, my questions is not particularly on what component to buy, but I have not built a custom PC in about 5+ years. Use to read up on the latest GPU/CPU, had the best fans or latest mobo and the newest components every 6 months. My questions is what really has changed in the custom PC world? I see new components in your standard Dell/HP/etc machines, but what has changed in the DIY/custom PC world? Anything ground breaking that you do not get in ordering off the shelf.

Our search engine goes beyond from simply matching string, to matching ontology annotations. The algorithm used for semantic search, extensively used in information retrieval and is an extension of the vector space model [31]. In sum, pursuantly to this model, two letters d and q queries are represented as vectors

$$d_i = (w_{1,d}, w_{2,d}, \dots, w_{n,d})$$

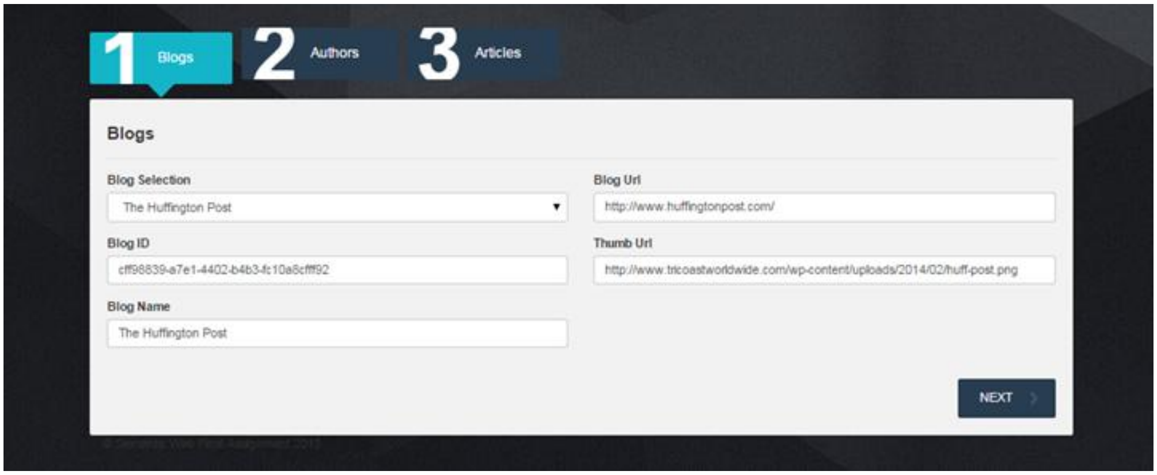
$$q = (w_{1,q}, w_{2,q}, \dots, w_{t,q})$$

Each $w_{j,i}$ is a weight for the annotation j in article i, and reflects the importance of that term. Those weights are computed on the basis of the frequency of the terms in the article, the query or the collection and are calculated based on the annotations produced by the Semantic Annotator. At retrieval time, the documents are ranked by the cosine of the angle between the document vectors and the query vector.

The vector of the articles is computed based on the following metric $w_{t,d} = idf_t * tf_{t,d}$ where $tf_{t,d}$ is the frequency of the term in the specific articles d and idf_t is the inverse document frequency of the term t in all documents in the collection. All those weights are computed before and stored in our database to avoid the costly recalculation at runtime. The approach has already been proven to have fertile results in the medical domain [32][39]. The novelty of our search engine lies in the fact that since the data are linked, using the appropriate SPARQL queries, complex queries over the social graph and the time dimension can be exploited (e.g. complex queries such as, give all articles published by the coauthors of “Jim Morison” published during November and December 2015 can be answered). Moreover, we uniquely combine text search with search based on ontology annotations for ranking the results.

3.3.2 The Publishing Wizard

Except to searching for information in the triple store an additional web application is provided to manually register articles in our data repository. The application is created as a wizard to enable the uninterrupted and easy insertion of relevant information. A screenshot is shown in Figure 6Figure 6 and is also available online[10].



Blogs

Blog Selection: The Huffington Post ▾

Blog ID: cff98839-a7e1-4402-b4b3-fc10a8cfff92

Blog Name: The Huffington Post

Blog Url: http://www.huffingtonpost.com/

Thumb Url: http://www.tricoastworldwide.com/wp-content/uploads/2014/02/huff-post.png

Figure 6: The Publishing Wizard

[10] http://83.212.124.52/publish

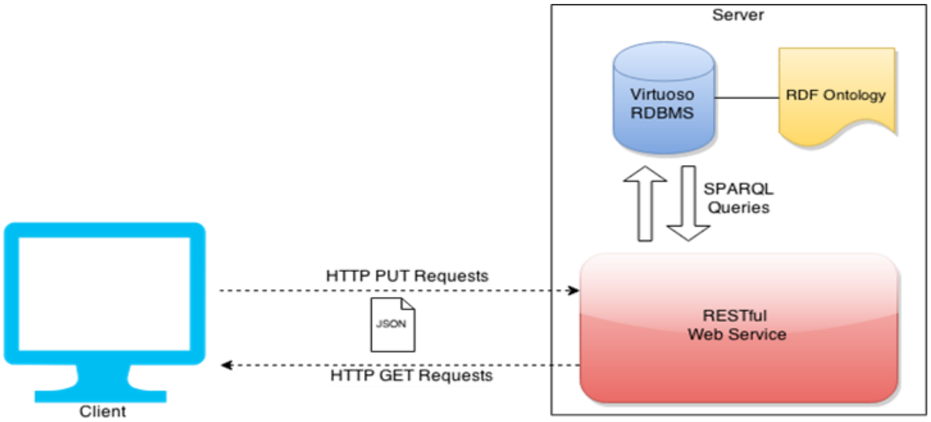
Chapter 4 - Implementation and Tools

For the purposes of this work, a Blog's RDF Ontology was implemented, adopting a client-server architecture. The client-server model is adopted in order to enable clients to ubiquitously access (send and receive) to the ontology's database stored to a Web application server.

In this direction, the "OpenLink Virtuoso" was used, an Open-Source SQL-ORDBMS and Web Application Server hybrid that provides SQL, XML, and RDF data management in a single multithreaded server process. Triple Store access is available via SPARQL, SIMILE Semantic Bank API, ODBC, GRDDL, JDBC, ADO.NET, XMLA, WebDAV, and Virtuoso/PL (SQL Stored Procedure Language).

Above this, a RESTful Web service is made to provide the appropriate intercommunicate between the client and Virtuoso's RDF Ontology. The web service accepts HTTP PUT requests sent by the client from the available web form and forwards them to ontology's database. It also forwards Virtuoso's replies to the client's HTTP GET requests. SPARQL queries are used in both, send and receive transactions, between web service and Virtuoso (Figure 7).

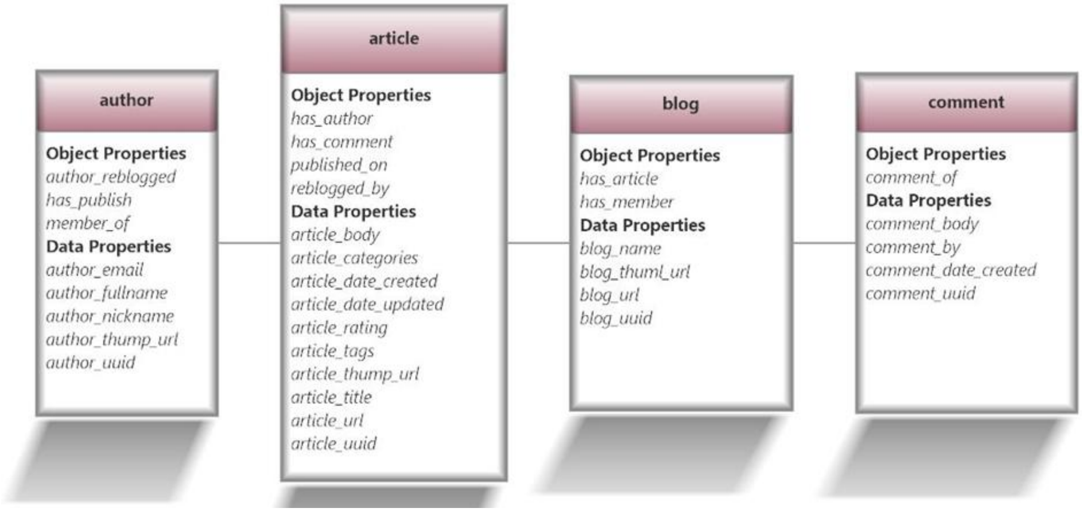
Figure 7: Web Service Architecture



4.1 Ontology Schema

An ontology design pattern is a reusable solution to a recurring ontology modeling problem. Our problem was to identify and describe the elements of a blog that is posted in the Internet. So we created and fully described the ontologies that a blog can be described when it is posted in the Internet. The ontologies were built incrementally according to the current state of the blogs posted in the Internet. Based on the analysis of the requirements, we decided to use the following classes to store the persistent data for the blog elements. We used Protégé to create the model of the blog.

The following diagram (created by smart draw) shows the classes and the object and data properties of each one.



4.1.1 Classes

Below are the classes used to describe the blog.

article

The article ontology was used to describe all the articles of a blog.

author

The author ontology was used to describe all the authors of the articles of a blog, or the authors of a comment or the authors that reblogged an article of a blog.

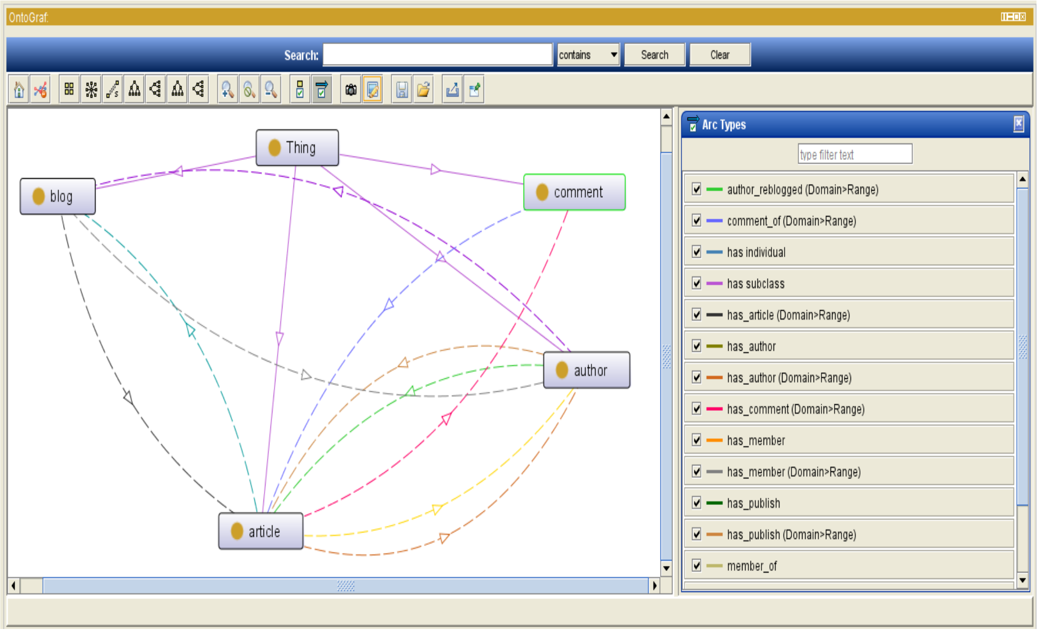
blog

The blog ontology was used to describe the blog itself with all of its characteristics.

comment

The comment ontology was used to describe the comments that are been made in an article of a blog. The classes can be seen in Figure 8.

Figure 8: Classes of the Ontology



☑ author_reblogged (Domain>Range)
☑ comment_of (Domain>Range)
☑ has individual
☑ has subclass
☑ has_article (Domain>Range)
☑ has_author
☑ has_author (Domain>Range)
☑ has_comment (Domain>Range)
☑ has_member
☑ has_member (Domain>Range)
☑ has_publish
☑ has_publish (Domain>Range)
☑ member_of

4.1.2 Object Properties

The following object properties were used to describe the relations between the classes.

comment_of

This property describes the relation “A comment has been made on an article”.

author_reblogged

This property describes the relation “An author reblogged an article”.

has_article

This property describes the relation “A blog has an article”.

has_author

This property describes the relation “An article has an author”.

has_comment

This property describes the relation “An article has comment”.

has_member

This property describes the relation “A blog has an author”.

has_publish

This property describes the relation “An author has published an article”.

member_of

This property describes the relation “An author has a blog”.

published_on

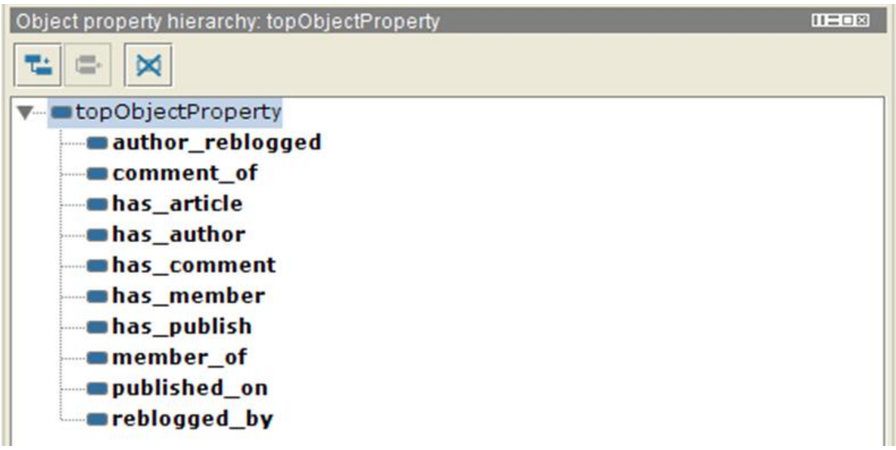
This property describes the relation “An article has been published in a blog”.

reblogged_by

This property describes the relation “An article reblogged by an author”.

The object properties can be seen in Figure 9 as shown in Protégé.

Figure 9: Object Properties



- topObjectProperty
  - author_reblogged
  - comment_of
  - has_article
  - has_author
  - has_comment
  - has_member
  - has_publish
  - member_of
  - published_on
  - reblogged_by

4.1.3 Data Properties

Data Properties

The following data properties were used to describe the relations between instances of the classes of a blog and the Xml schema datatypes.

article_body

The datatype of the article body text was declared as Literal.

article_categories

The datatype of the name of the article category text was declared as string.

article_date_created

The datatype of the date the article created was declared as dateTime.

article_date_updated

The datatype of the date the article updated was declared as dateTime.

article_rating

The datatype of the rating of the article was declared as integer.

article_tags

The datatype of the tags of the article was declared as string.

article_thump_url

The datatype of the thumbnail Url of the article was declared as anyURI.

article_title

The datatype of the title of the article was declared as string.

article_url

The datatype of the article Url was declared as anyURI.

article_uuid

The datatype of the article unique ID was declared as string.

author_email

The datatype of the author email was declared as Literal.

author_fullname

The datatype of the author fullname was declared as string.

author_nickname

The datatype of the author nickname was declared as string.

author_thump_url

The datatype of the author thumbnail Url was declared as anyURI.

author_uuid

The datatype of the author unique ID was declared as string.

blog_name

The datatype of the blog name was declared as string.

blog_thump_url

The datatype of the blog thumbnail Url was declared as anyURI.

blog_url

The datatype of the blog Url was declared as anyURI.

blog_uuid

The datatype of the blog unique ID was declared as string.

comment_body

The datatype of the comment text body was declared as Literal.

comment_by

The datatype of the comment author was declared as string.

comment_date_created

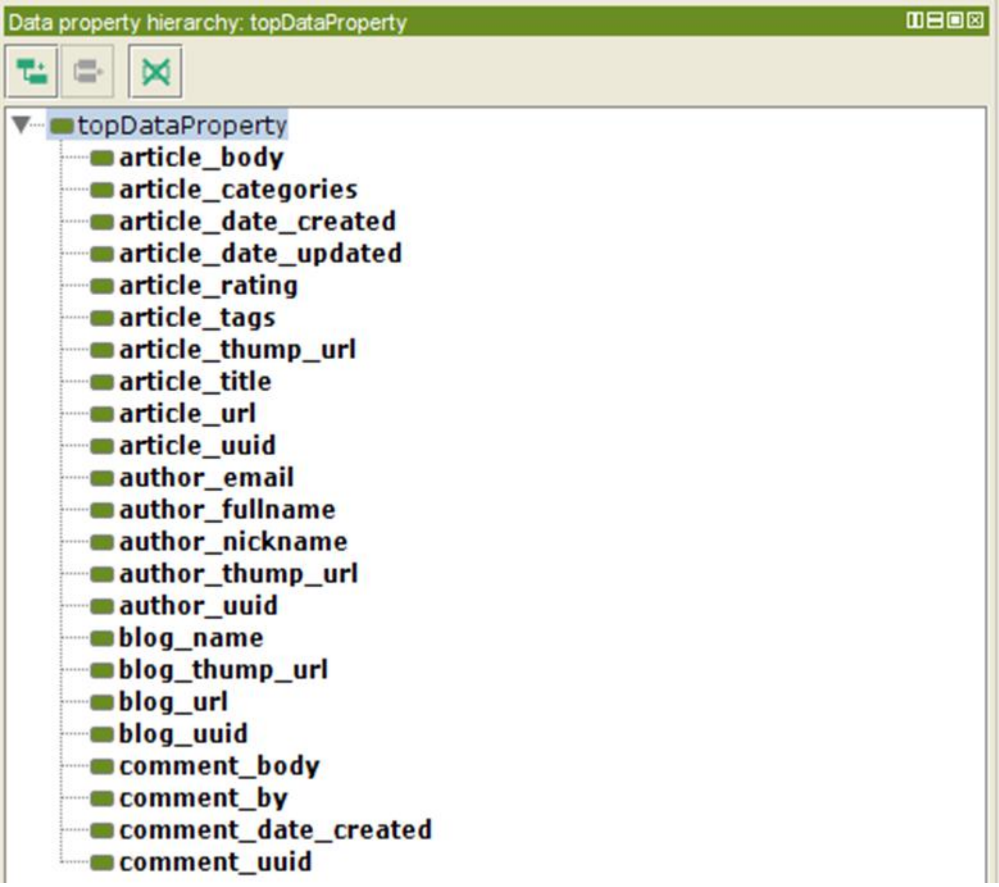
The datatype of the date the comment created was declared as dateTime.

comment_uuid

The datatype of the comment unique ID was declared as string.

The data properties can be seen in Figure 10 as shown in Protégé.

Figure 10: Data Properties



```
▼ topDataProperty
    article_body
    article_categories
    article_date_created
    article_date_updated
    article_rating
    article_tags
    article_thump_url
    article_title
    article_url
    article_uuid
    author_email
    author_fullname
    author_nickname
    author_thump_url
    author_uuid
    blog_name
    blog_thump_url
    blog_url
    blog_uuid
    comment_body
    comment_by
    comment_date_created
    comment_uuid
```

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

4.2 OpenLink Virtuoso

Virtuoso Universal Server is a middleware and database engine hybrid that combines the functionality of a traditional RDBMS, ORDBMS, virtual database, RDF, XML, free-text, web application server and file server functionality in a single system. Rather than have dedicated servers for each of the aforementioned functionality realms, Virtuoso is a "universal server"; it enables a single multithreaded server process that implements multiple protocols. The open source edition of Virtuoso Universal Server is also known as OpenLink Virtuoso. The software has been developed by OpenLink Software with Kingsley Uyi Idehen and Orri Erling as the chief software architects [40]

4.3 Web Services

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. Two major classes of Web services can be identified:

- **R.E.S.T.-compliant Web services**, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and
- **Arbitrary Web services**, in which the service may expose an arbitrary set of operations.

Both classes of Web services use URIs to identify resources and use Web protocols (such as HTTP and SOAP 1.2) and XML data formats for messaging [41].

4.4 REST Architecture

A Web API is a development of Web services emphasizing into simpler representational state transfer (R.E.S.T.) based communications. R.E.S.T.ful APIs do not require XML-based Web service protocols, like SOAP and WSDL, to support their interfaces. Thus, they are more suitable for the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, making them more efficient for the Web. In the REST architectural style data and functionality are considered resources are accessed using Uniform Resource Identifiers (URIs, typically links on the Web). The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol [42]. The following principles encourage RESTful applications to be simple, lightweight, and fast:

Resource identification through URI: A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.

Uniform interface: Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.

Self-descriptive messages: Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.

Stateful interactions through hyperlinks: Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

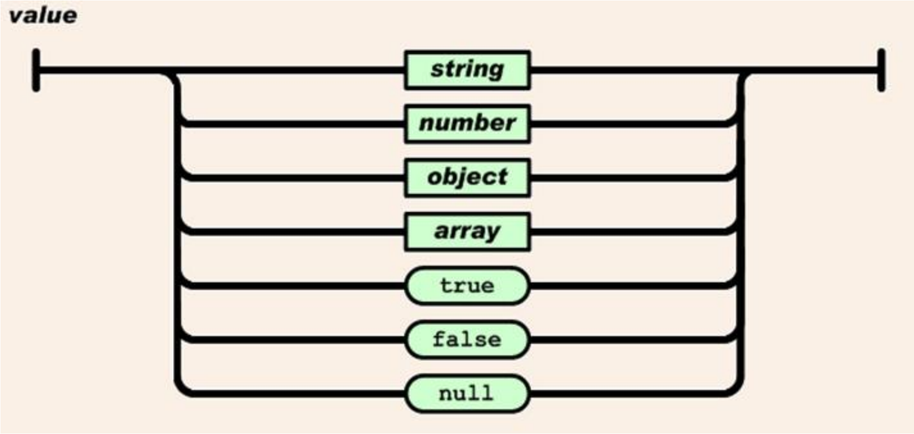
4.5 JSON

The JS.O.N. format (JavaScript Object Notation) is a lightweight open-standard data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures [42].

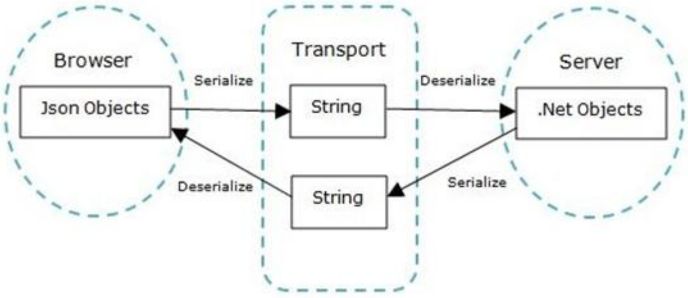
Figure 11: Structures of a Value at JSON Format



In JSON, they take on these forms:

An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma). An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma). A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested. A string is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string. A number is very much like a C or Java number, except that the octal and hexadecimal formats are not used [42].

Figure 12: JSON Serialization between Server and Browser



4.6 Virtuoso Database

4.6.1 Core database engine

Virtuoso provides an extended object-relational model, which combines the flexibility of relational access with inheritance, run time data typing, late binding, and identity based access. Virtuoso Universal Server database includes physical file and in memory storage and operating system processes that interact with the storage. There is one main process, which has listeners on a specified port for HTTP, SOAP, and other protocols.

4.6.2 Architecture

Virtuoso is designed to take advantage of operating system threading support and multiple CPUs. It consists of a single process with an adjustable pool of threads shared between clients. Multiple threads may work on a single index tree with minimal interference with each other. One cache of database pages is shared among all threads and old dirty pages are written back to disk as a background process.

The database has at all times a clean checkpoint state and a delta of committed or uncommitted changes to this checkpointed state. This makes it possible to do a clean backup of the checkpoint state while transactions proceed on the commit state.

A transaction log file records all transactions since the last checkpoint. Transaction log files may be preserved and archived for an indefinite time, providing a full, recoverable history of the database.

A single set of files is used for storing all tables. A separate set of files is used for all temporary data. The maximum size of a file set is 32 terabytes, for 4G × 8K pages.

4.6.3 Locking

Virtuoso provides dynamic locking, starting with row level locks and escalating to page level locks when a cursor holds a large percentage of a page's rows or when it has a history of locking entire pages. Lock escalation only happens when no other transactions hold locks on the same page, hence it never deadlocks. Virtuoso SQL provides means for exclusive read and for setting transaction isolation.

Transactions

All four levels of isolation are supported: Dirty read, read committed, repeatable read and serializable. The level of isolation may be specified operation by operation within a single transaction. Virtuoso can also act as a resource manager and/or transaction coordinator under Microsoft's Distributed Transaction Coordinator (MS DTC) or the XA standard.

4.6.4 Data integrity

Virtuoso ORDBMS database supports entity integrity and referential integrity. Virtuoso ensures that relationships between records in related tables are valid by enforcing referential integrity. Integrity constraints include:

NOT NULL - Within the definition of a table, Virtuoso allows data to contain a NULL value. This NULL value is not really a value at all and is considered an absence of value. The constraint of NOT NULL required a value to be assigned to the field.

Unique Key - Uniqueness for a column or set of columns means that the values in that column or set of columns must be different from all other columns or set of columns in that table. A unique key may contain NULL values since they are by definition a unique non-valued value.

Primary Key - Primary key are much like unique keys except that they are designed to uniquely identify a row in a table. They can consist of a single column or multiple columns. The primary key cannot contain a NULL value.

CHECK Constraint - Virtuoso provides on a column an integrity constraint that requires certain conditions to be met before the data is inserted or modified. If the checks are not satisfied then the transaction cannot be completed.

4.6.5 Data dictionary

Virtuoso stores all its information about all user objects in the database in the system catalog tables designated by db.dba*.

4.7 SPARQL

SPARQL is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. It was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web. SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns. Implementations for multiple programming languages exist. There exist tools that allow one to connect and semi-automatically construct a SPARQL query for a SPARQL endpoint, for example ViziQuer. In addition, there exist tools that translate SPARQL queries to other query languages, for example to SQL and to XQuery.

4.7.1 SPARQL Advantages

SPARQL allows users to write queries against data that can loosely be called "key-value" data or, more specifically, data that follows the RDF specification of the W3C. The entire database is thus a set of "subject-predicate-object" triples. This is analogous to some NoSQL databases' usage of the term "document-key-value", such as MongoDB.

RDF data can also be considered in SQL relational database terms as a table with three columns - the subject column, the predicate column and the object column. Unlike relational databases, the object column is heterogeneous, the per-cell data type is usually implied (or specified in the ontology) by the predicate value. Alternately, again comparing to SQL relational, all of the triples for a given subject could be represented as a row, with the subject being the primary key and each possible predicate being a column and the object is the value in the cell. However, SPARQL/RDF becomes easier and more powerful for columns that could contain multiple values (like "children"), and where the column itself could be a joinable variable in the query, rather than directly specified.

SPARQL thus provides a full set of analytic query operations such as JOIN, SORT, AGGREGATE for data whose schema is intrinsically part of the data rather than requiring a separate schema definition. Schema information (the ontology) is often provided externally, though, to allow different datasets to be joined in an unambiguous manner. In addition, SPARQL provides specific graph traversal syntax for data that can be thought of as a graph.

4.7.2 SPARQL Query Forms

In the case of queries that read data from the database, the SPARQL language specifies four different query variations for different purposes.

SELECT query: Used to extract raw values from a SPARQL endpoint, the results are returned in a table format.

CONSTRUCT query: Used to extract information from the SPARQL endpoint and transform the results into valid RDF.

ASK query: Use to provide a simple True/False result for a query on a SPARQL endpoint.

DESCRIBE query: Used to extract an RDF graph from the SPARQL endpoint, the contents of which is left to the endpoint to decide based on what the maintainer deems as useful information.

Each of these query forms takes a WHERE block to restrict the query although in the case of the DESCRIBE query the WHERE is optional. SPARQL 1.1 specifies a language for updating the database with several new query forms [43].

4.8 Virtuoso Setup

Another important key for our infrastructure is the host computer. In order for full 24/7 availability, the implementation must be hosted in a server always-on and connected. Thus, we chose the IaaS (Infrastracture as a Service) Cloud Service solution, requesting a Virtual Machine from Okeanos, a Cloud Service Provider designed and developed by the Greek Research and Technology Network (GRNET). A Virtual Machine from Cyclades, Okeanos' Compute and Network Service is used as the server for our infrastructure.

We use OpenLink Virtuoso open source v.6.1 edition. Before we start the installation we set the appropriate environment variables. We determine the root location for the Virtuoso installation. We put the zip folder into the C:/Program Files (x86) directory. We start the System control panel (right-click My Computer and select Properties, or drill down through the Start menu -> Control Panels -> Administrative Tools -> System).

Then we followed the steps below:

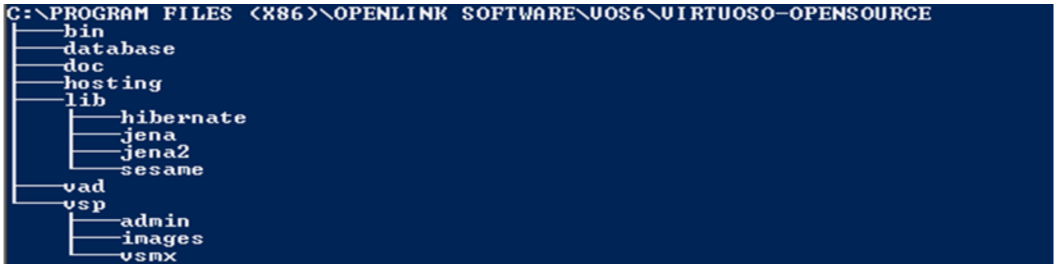
1. Click Advanced -> Environment Variables, create a new system environment variable called VIRTUOSO_HOME, with this path C:/Program Files/OpenLink Software/VOS6/virtuoso-opensource/) for its value.
2. Locate the PATH system environment variable, click to EDIT it.
3. Add the string below to the end of the existing PATH value.

We have to be careful not overwrite the existing PATH value! Doing so will disrupt all use of our Windows environment.

;%VIRTUOSO_HOME%/bin;%VIRTUOSO_HOME%/lib

After that, click OK or Exit buttons until we have fully exited the System control panel. Now we are ready to unzip the file to our chosen location. This will create a directory virtuoso-open-source, containing 6 subfolders:

Figure 13: The six subfolders of Virtuoso



```
C:\PROGRAM FILES (X86)\OPENLINK SOFTWARE\VOS6\VIRTUOSO-OPENSOURCE
├───bin
├───database
├───doc
├───hosting
├───lib
│   ├───hibernate
│   ├───jena
│   ├───jena2
│   └───sesame
├───vad
└───vsp
    ├───admin
    ├───images
    └───vsmx
```

The current Windows binary package is missing a php.ini file, required for PHP runtime hosting support. So, we download a copy of this file and manually place it in the database directory.

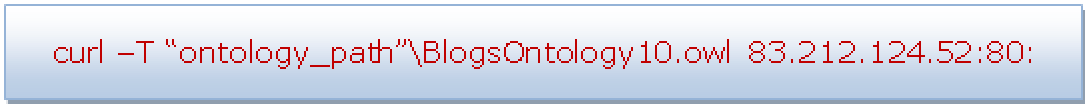
The default administrator username and password are both dba. By default, the Virtuoso server will listen for HTTP connections at TCP port 8890, and for SQL data access (via iSQL, ODBC, JDBC, OLE DB, ADO.NET, etc.) at TCP port 1111. These ports may be changed by editing the virtuoso.ini file.

4.8.1 Ontology Import

After having virtuoso running on a public server, the next step is to upload the ontology, we have created in Protégé, in order to be able to insert, update or query for data via Jena. In our workaround, the usage of a text based application named ‘curl’ is necessary, so we simply had to download and install it, as it is not a windows default. Now in windows command prompt we uploaded the ontology by simple typing:

83.212.124.52:80

Figure 14: Ontology's import main command



```
curl –T "ontology_path"\BlogsOntology10.owl 83.212.124.52:80:
```

4.9 Web Service Implementation

The project was developed completely using NetBeans IDE. The IDE supports rapid development of RESTful web services using JSR 311 - Java API for RESTful Web Services (JAX-RS) and Jersey, the reference implementation for JAX-RS.

In addition to building RESTful web services, the IDE also supports testing, building client applications that access RESTful web services, and generating code for invoking web services (both RESTful and SOAP-based.)

Here is the list of RESTful features provided by the IDE:

a. Rapid creation of RESTful web services from JPA entity classes and patterns.
b. Rapid code generation for invoking web services such as Google Map, Yahoo News Search, and StrikeIron web services by drag-and-dropping components from the Web Services manager in the Services window.
c. Generation of RESTful Java Clients for services registered in the Web Services manager.
d. Test client generation for testing RESTful web services.
e. Logical view for easy navigation of RESTful web service implementation classes in the project.

4.9.1 Structure

The most important step in any RESTful design is the identification of the resources that should be made. Our Web Service is a Resource-Oriented RESTful service revolving four resources: Blogs, Authors, Articles, Comments. In order to be available for HTTP interactions, each resource is identified and assigned to a URI, such as http://.../SemanticWS-1.4/rest/blogs, http://.../SemanticWS-1.4/rest/authors etc.

In addition, we introduced specific methods for insert and retrieve interactions between the web service and the resource. In both cases, resources are accessed using actual HTTP GET and PUT requests. Some of these methods can be used from different resources, enabling interoperate capabilities. Recourses and methods introduced analytically in API Reference section.

Figure 15: Structure of the Web Service

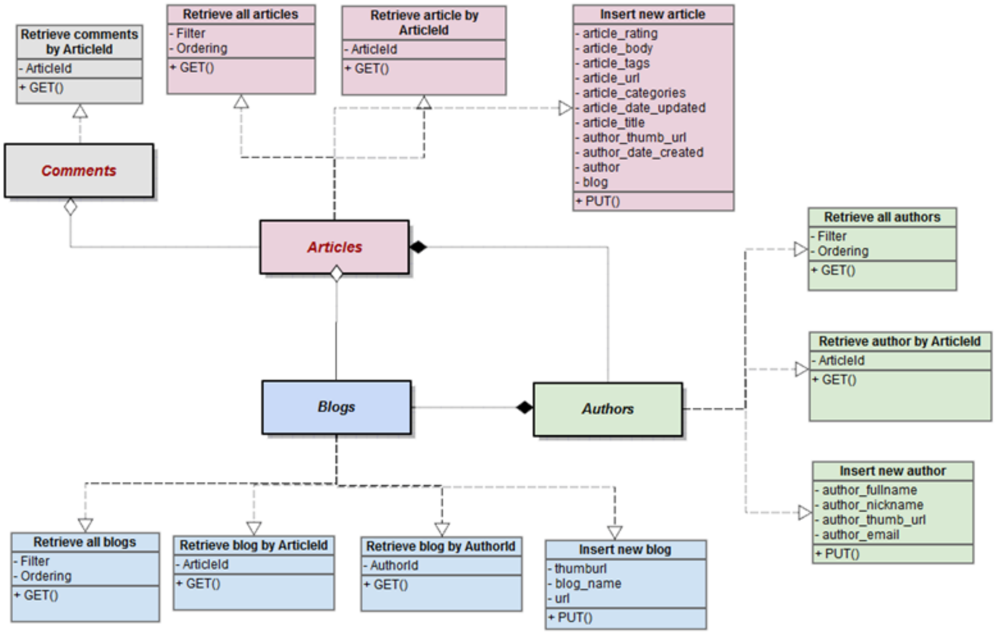
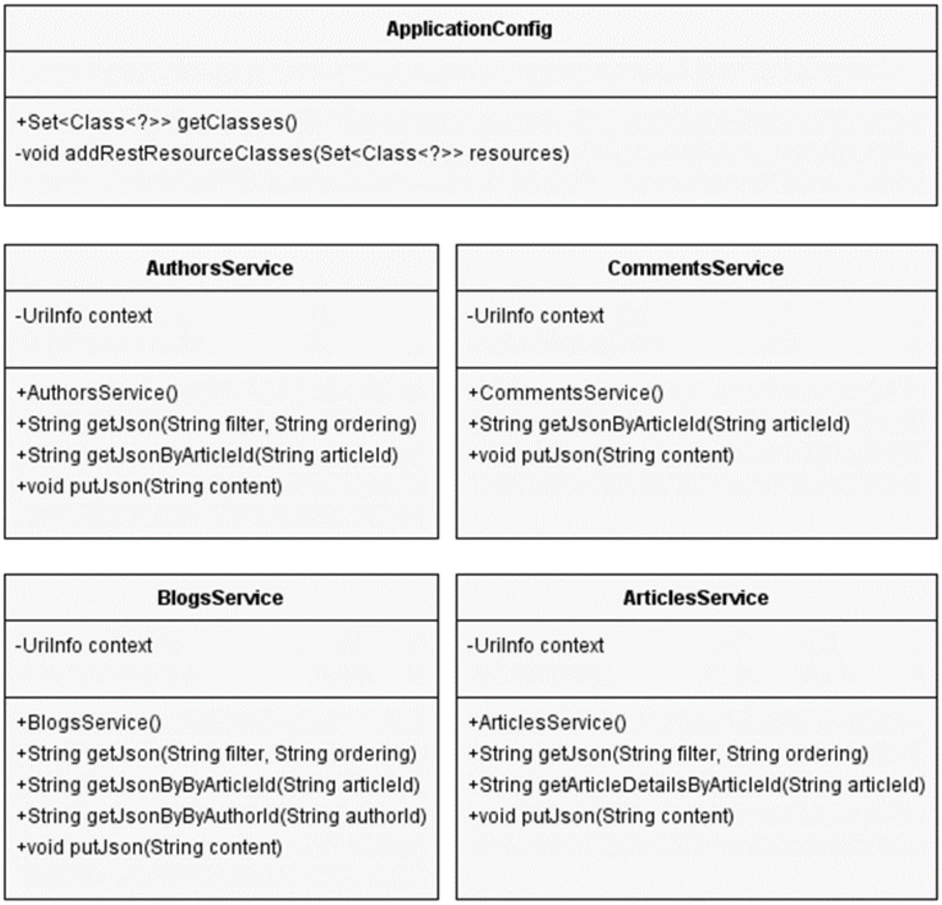


Figure 16: Class Diagram



```
ApplicationConfig

+Set<Class<?>> getClasses()
-void addRestResourceClasses(Set<Class<?>> resources)
```

```
AuthorsService
-UriInfo context
+AuthorsService()
+String getJson(String filter, String ordering)
+String getJsonByArticleId(String articleId)
+void putJson(String content)
```

```
CommentsService
-UriInfo context
+CommentsService()
+String getJsonByArticleId(String articleId)
+void putJson(String content)
```

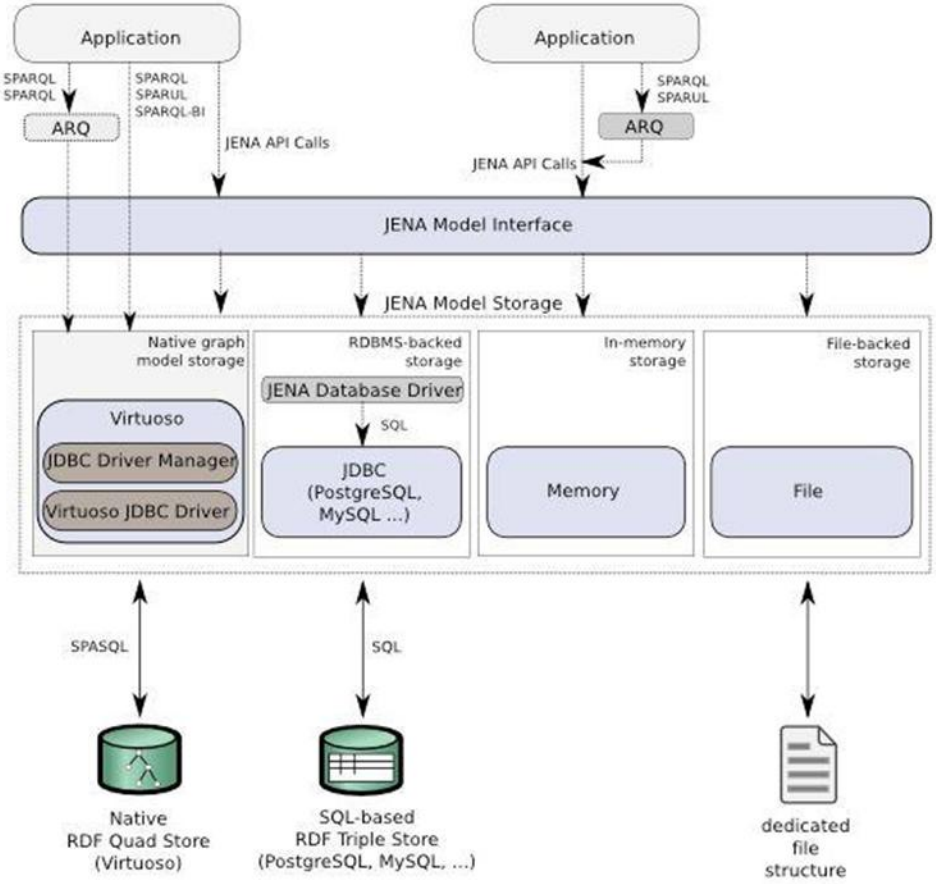
```
BlogsService
-UriInfo context
+BlogsService()
+String getJson(String filter, String ordering)
+String getJsonByByArticleId(String articleId)
+String getJsonByByAuthorId(String authorId)
+void putJson(String content)
```

```
ArticlesService
-UriInfo context
+ArticlesService()
+String getJson(String filter, String ordering)
+String getArticleDetailsByArticleId(String articleId)
+void putJson(String content)
```

4.10 Jena API

For the implementation we used the Apache Jena RDF API and the connection was with the Virtuoso Jena Provider. Jena is an open source Semantic Web framework for Java which can be used to create and manipulate RDF graphs (including data extraction and writing to RDF graphs)The graphs are represented as an abstract model which denotes an RDF graph, so called because it contains a collection of RDF nodes, attached to each other by labeled relations. A model can be sourced with data from files, databases, URIs or a combination of these. Jena has methods for reading and writing RDF as XML. These can be used to save an RDF model to a file and later read it back in again. For linking Jena with Virtuoso we used the Virtuoso Jena Provider. The Virtuoso Jena RDF Data Provider is a fully operational Native Graph Model Storage Provider for the Jena Framework, which enables Semantic Web applications written using the Jena RDF Frameworks to directly query the Virtuoso RDF Quad Store. The way of how this tool works is shown in the next figure.

Figure 17: The Way Jena Works



Each arc in an RDF Model is called a statement. Each statement asserts a fact about a resource. A statement has three parts (the subject, the predicate and the object). A statement is sometimes called a triple, because of its three parts. An RDF Model is represented as a set of statements. The Jena model interface defines a listStatements() method which returns an StmtIterator, a subtype of Java's Iterator over all the statements in a Model. StmtIterator has a method nextStatement() which returns the next statement from the iterator (the same one that next() would deliver, already cast to Statement). The Statement interface provides accessor methods to the subject, predicate and object of a statement. Since the object of a statement can be either a resource or a literal, the getObject() method returns an object typed as RDFNode, which is a common superclass of both Resource and Literal. The underlying object is of the appropriate type, so the code uses instance of to determine which and processes it accordingly.

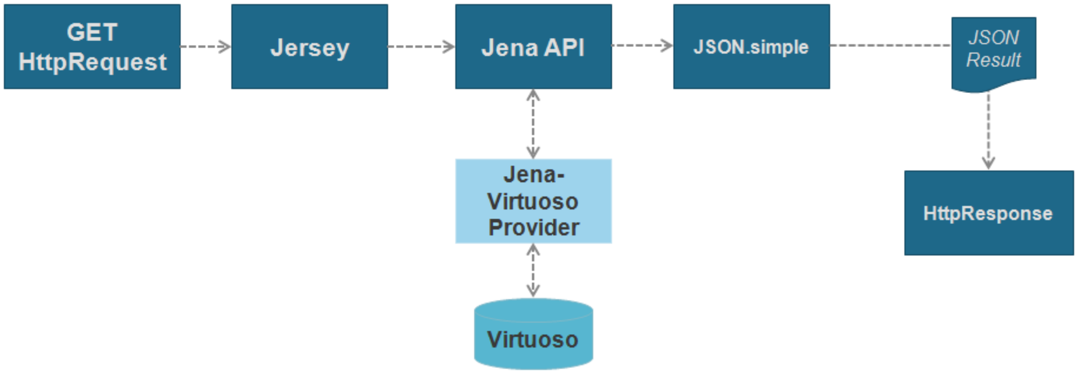
To use Virtuoso-specific SPARQL extensions, queries must bypass the Jena/ARQ parser and go straight to the Virtuoso server. This is done by using the VirtuosoQueryExecutionFactory.create() method, which always invokes the Jena/ARQ parser, which in turn rejects any Virtuoso-specific extensions. Some of the important methods used in our project mainly for the database communication were: get(gets metadata by ID), insert(inserts a new file), patch(updates file metadata), update(updates content as well as file metadata),copy(copies files),delete(deletes files). A big amount of these methods could be used by case not only for files and metadata but for parent-children relationships, permissions, revision, apps, comments and replies, real-life factors and properties for a bunch of other situation as well, along with other important methods.

Some examples of the executed code are presented below:

***GET** Handling*

```
VirtGraph set = new VirtGraph(....);
String query = "[Select query]";
Query sparql = QueryFactory.create(query);

VirtuosoQueryExecution vqe = VirtuosoQueryExecutionFactory.create(sparql, set);
ResultSet results = vqe.execSelect();

while (results.hasNext()) {

    QuerySolution result = results.nextSolution();
    RDFNode graph = result.get("graph");
    RDFNode comment_uuid = result.get("comment_uuid");
    RDFNode comment_by = result.get("comment_by");
    RDFNode comment_date_created = result.get("comment_date_created");
    RDFNode comment_body = result.get("comment_body");

    ...
    ** Json Handling **
    ....

}
```

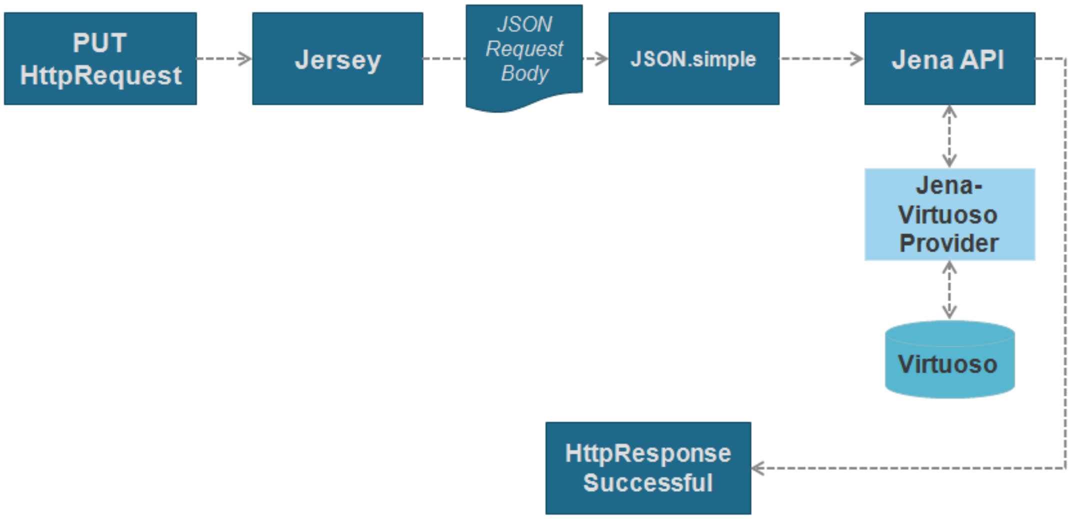
Figure 18: GET Request Lifecycle



***PUT** Handling*

```
VirtGraph set = new VirtGraph(...);

String query = "[Insert query]";
System.out.println(query);
VirtuosoUpdateRequest req = VirtuosoUpdateFactory.create(query, set);
req.exec();
```

Figure 19: PUT Request Lifecycle



4.11 JSON Parsing

For our project we used the JSON.simple toolkit for Java. JSON.simple allows us to encode and decode JSON text and it is fully compatible with JSON specifications. It is flexible, easy to use, provides encode, decode/parse and escape JSON text functionalities, supports streaming output of JSON text and it is not dependent on external libraries. The mapping between JSON and Java entities is described in the following table:

Table 1: Mapping between JSON and Java Entities

JSON	JAVA
String	java.lang.String
Number	java.lang.Number
true|false	java.lang.Boolean

Null	null
Array	java.util.List
Object	java.util.Map

Chapter 5 - Evaluation

Evaluating and validating software product, play important role to both its possession and development. The amount of importance of the several characteristics of software quality is strictly tied to the intended usage and the objectives of the system. In order to appraisal the quality issues during all the life cycle of the software, we followed standardized software development practices to reduce the likelihood of defects and the cost for both users and developers. For the assurance of quality, norms defined from the International Organization for Standardization (ISO) such as the Software Product Quality Requirements and evaluation [44] (SQUARE) has been used as a reference model. To this direction, the applicable functional and non-functional requirements according to ISO/IEC 25023 [45] had been defined in the early stages of the project and monitored during the whole software life-cycle.

For the evaluation, the quality attributes from the item quality model of the ISO/IEC 25000 arrangement alongside the System Usability Scale (SUS) for worldwide evaluation of frameworks convenience were utilized. At the assessment stage 30 understudies from the postgraduate project of the Department of Informatics Engineering, Technological Educational Institute of Crete were utilized. Having such a gathering of evaluators, the assessment questions must be basic, precise, straightforward, non-tedious and without loss of usefulness/quality. Thus the significant sub-attributes of programming quality measures from ISO/IEC 25000 arrangement have been deciphered into straightforward inquiries in regular dialect. The assessment type of the web crawler was a rundown of such inquiries where the evaluator needed to reply with a level of fulfillment with Likert scale [46]. The structure can be found in the Appendix. We likewise utilized the System Usability Scale (SUS) [47] for worldwide appraisal of frameworks ease of use.

Table 2: The results for the various evaluation categories

Functionality	Suitability	3,57 / 5	3,62
	Accurateness	3,43 / 5	
	Compliance	3,86 / 5	
Efficiency	Time Behavior	4,36 / 5	4
	Resource utilization	3,64 / 5	

Compatibility	Co-existence	3,57 / 5	
	Interoperability	3,21 / 5	3,39
Usability	Understandability	3,71 / 5	3,98
	Learnability	4,14 / 5	
	Operability	3,86 / 5	
	Attractiveness	4,21 / 5	
Reliability	Maturity	3,36 / 5	3,26
	Fault tolerance	3,07 / 5	
	Recoverability	3,36 / 5	
Maintainability	Analyzability	4 / 5	4,05
	Changeability	4,14 / 5	
	Stability	4 / 5	
	Testability	4,07 / 5	
Portability	Adaptability	3,86 / 5	3,8
	Installability	3,71 / 5	
	Conformance	3,93 / 5	
	Replaceability	3,71 / 5	
Quality of use	Effectiveness	3,43 / 5	3,55
	Efficiency	3,5 / 5	
	Satisfaction	3,57 / 5	
	Health and safety risk	3,71 / 5	
SUS	71,5 / 100		

The results are shown in Table 2. Values more than three represent to abnormal state of the particular software characteristics while values somewhere around 2.5 and 3 are in generally safe. Values beneath 2.5 are viewed as high risk. For our situation, all outcomes were evaluated with a normal above 3 demonstrating the high quality of the product. The most reduced normal score was for the unwavering quality classification (3,26/5) since now and again, blunders show up without an appropriate clarification. Also, the viability class has the most astounding normal score (4,05) since it is entirely insignificant to redesign an API call, a SPARQL inquiry or a call from the interface.

The SUS usability score was 71 on a scale of 0 to 100. In the literature, The average SUS score been measured by Sauro et al. [48] as the 62.1 but as a standard that can be characterized as “gold” is often used the 68. A SUS score above 68 would be considered above average and anything below 68 is below average. This indicating that the Blogs Search engine reaches an acceptable level, but that there is still lots of space for improvement in the usability.

At last, the evaluation brought about important feedback on the present state and in clear headings for the future development and deployment. For instance, some minor issues were distinguished in the route on the indexed lists and on the requesting of them which were rectified.

Chapter 6 - Conclusion

Concluding, in this work, it is presented a novel platform allowing registering and searching blog articles as Linked Open Data. Besides the SPARQL endpoint, the platform also provides APIs which expose the information in public and allow external applications to perform CRUD operations on it. All information is represented using an RDF/S ontology for blogs.

As future work, the creation of a crawler, which is able to automatically insert blog sites and articles is being considered. Additionally, within our future plans is to annotate the blog text using terms from a refined version of the ontology; resulting in identifying the content of the article based on the whole published text available. Blogs will become increasingly popular in the following years and feature interesting challenges remained to be investigated in the near future.

References

[1] "Wikipedia Article about Blog," [Online]. Available: http://en.wikipedia.org/wiki/Blog. [Accessed June 2016].

[2] N. Bansal and N. Koudas, "BlogScope: a system for online analysis of high volume text streams," in *VLDB '07 Proceedings of the 33rd international conference on Very large data bases*, Vienna, Austria, 2007.

[3] R. Blood, "How blogging software reshapes the online community," *Communications of the ACM - The Blogosphere*, vol. 12, no. 47, pp. 53-55, December 2004.

[4] R. Blood, "Weblogs: A history and perspective.," Rebecca's Pocket, 7 September 2000. [Online]. Available: http://www.rebeccablood.net/essays/weblog_history.html. [Accessed July 2016].

[5] J. Rodzvilla and R. Blood, We've Got Blog: How Weblogs Are Changing Our Culture, Basic Books, 2002.

[6] B. A. Nardi, D. J. Schiano and M. Gumbrecht, "Blogging as social activity, or, would you let 900 million people read your diary?," in *CSCW '04 Proceedings of the 2004 ACM conference on Computer supported cooperative work*, New York, USA, 2004.

[7] T. Stauffer, Blog On: Building Online Communities with Web Logs, New York: McGraw-Hill, 2002.

[8] P. Bausch, M. Haughey and M. Hourihan, We Blog: Publishing Online with Weblogs, New York: Wiley, 2002.

[9] S. C. Herring, I. Kouper, J. C. Paolillo, L. A. Scheidt, M. Tyworth, P. Welsch, E. Wright and N. Yu, "Conversations in the Blogosphere: An Analysis "From the Bottom Up"," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005.

[10] M. Brandy, "Blogging: personal participation in public knowledge-building on the web," in *Participating in the knowledge society*, Palgrave Macmillan UK, Springer, 2005, pp. 212-228.

[11] C. Barrett, "Anatomy of a Weblog," camworld, 26 January 1999. [Online]. Available:

http://camworld.org/1999/01/26/anatomy-of-a-weblog-2/. [Accessed June 2016].

[12] C. Marlow, "Audience, structure and authority in the weblog community.," in *International Communication Association Conference*, 2007.

[13] D. Drezner and H. Farrell, "The power and politics of blogs," in *100th Annual Meeting of the American Political Science Association*, Chicago, Illinois, USA, 2004.

[14] M. Hauskrecht, "CS 2740 Knowledge representation: Semantic Web," [Online]. Available: https://people.cs.pitt.edu/~milos/courses/cs2740/Lectures/class16.pdf. [Accessed June 2016].

[15] L. Ding, T. Finin, A. Joshi, R. Pan, S. R. Cost, Y. Peng, P. Reddivari, V. Doshi and J. Sanchs, "Swoogle: a search and metadata engine for the semantic web," in *CIKM '04 Proceedings of the thirteenth ACM international conference on Information and knowledge management*, New York, USA, 2004.

[16] G. Ramanathan, R. McCool and E. Miller, "Semantic Search," in *Proceedings of the 12th international conference on World Wide Web*, Budapest, 2003.

[17] F. Arooj, C. Luca and G. Wilson, "User experience and efficiency for semantic search engine," in *2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, Bran, 2014.

[18] "Regator - Curated Blog Search and Discovery," [Online]. Available: http://regator.com/.

[19] "Meltwater IceRocket," [Online]. Available: http://www.icerocket.com/.

[20] T. Berners-Lee, J. Hendler and O. Lassila, "The semantic web," *Scientific american*, vol. 5, no. 284, pp. 28-37, 2001.

[21] P. Wilton, J. Tarling and J. McGinnis, "Storyline Ontology," BBC, 1 May 2013. [Online]. Available: http://www.bbc.co.uk/ontologies/storyline. [Accessed June 2016].

[22] J. G. Breslin, S. Decker, A. Harth and U. Bojars, "SIOC: an approach to connect web-based communities," *International Journal of Web Based Communities*, vol. 2, no. 2, pp. 133-142, 2006.

[23] "MerchantCircle," [Online]. Available: http://www.bloglines.com. [Accessed June 2016].

[24] "Technorati," [Online]. Available: http://technorati.com/. [Accessed June 2015].

[25] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn and G. Tummarello,

"Sindice.com: a document-oriented lookup index for open linked data," *International Journal of Metadata, Semantics and Ontologies,* vol. 1, no. 3, pp. 37-52, 2008.

[26] K. R. Page, D. R. C. David and K. Martinez, "REST and Linked Data: a match made for domain driven development?," in *WS-REST '11 Proceedings of the Second International Workshop on RESTful Design*, New York, NY, USA, 2011.

[27] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Regator. [Accessed 25 6 2016].

[28] "Twingly Blog," [Online]. Available: https://blog.twingly.com/. [Accessed 25 6 2016].

[29] I. Fundulaki, E. Daskalaki, G. Flouris, V. Papakonstantinou and N. Minadakis, "D4.4.1 Use Case Analysis and Classification of Choke Points," in *LDBC consortium*, 2013.

[30] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000.

[31] X. Feng , J. Shen and Y. Fan, "REST: An alternative to RPC for Web services architecture," in *Future Information Networks, 2009. ICFIN 2009. First International Conference on*, Beijing, 2009.

[32] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," 2001. [Online]. [Accessed June 2016].

[33] "JSON for Linked Data," [Online]. Available: http://json-ld.org/. [Accessed June 2016].

[34] D. Jurafsky and J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Pearson Education, Inc., 2000.

[35] S. Aït-Mokhtar, B. De Bruijn, C. Hagège and P. Rupi, "EURECA Enabling information re-Use by linking clinical Research and CAre, Deliverable D3.2: Initial prototype for relation identification between concepts," 8 July 2013. [Online]. Available: http://eurecaproject.eu/files/6514/3516/8469/D3.2_-_Initial_prototype_for_relation_identification_between_concepts.pdf. [Accessed June 2016].

[36] Y. Lei, V. Uren and E. Motta, "SemSearch: a search engine for the semantic web," in *EKAW'06 Proceedings of the 15th international conference on Managing Knowledge in a*

World of Networks, Berlin, 2006.

[37] "MerchantCircle Bloglines," [Online]. Available: http://www.bloglines.com. [Accessed December 2015].

[38] L. Zhang, Q. Liu, J. Zhang, H. Wang, Y. Pan and Y. Yu, "Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data," *The Semantic Web,* no. 4825, pp. 652-665, 2007.

[39] M. Melucci, "Vector-Space Model," in *Encyclopedia of Database Systems*, Springer US, 2009, pp. 3259-3263.

[40] H. Kondylakis, L. Koumakis, M. Psaraki, G. Troullinou, M. Chatzimina, E. Kazantzaki, K. Marias and M. Tsiknakis, "Semantically-enabled Personal Medical Information Recommender," in *International Semantic Web Conference (ISWC)*, Bethlehem, Pennsylvania, 2015.

[41] H. Kondylakis, L. Koumakis, S. Rüping, E. Kazantzaki, K. Marias and M. Tsiknakis, "PMIR: A Personal Medical Information Recommender," 2014.

[42] "Wikipedia article about Virtuoso Universal Server," [Online]. Available: http://en.wikipedia.org/wiki/Virtuoso_Universal_Server. . [Accessed June 2016].

[43] "Web Services Architecture - W3C," [Online]. Available: https://www.w3.org/TR/ws-arch/. [Accessed June 2016].

[44] "JSON," [Online]. Available: http://www.json.org/. [Accessed June 2016].

[45] "Wikipedia Article about SPARQL," [Online]. Available: https://en.wikipedia.org/wiki/SPARQL. [Accessed June 2016].

[46] "1471-2000-IEEE Recommended Practice for Architectural Description for Software-Intensive Systems.," IEEE Standards Association, 2000.

[47] *ISO/IEC 25023:2016 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Measurement of system and software product quality.*

[48] R. Likert, "A technique for the measurement of attitudes," in *Archives of psychology,* 1932.

[49] J. Brooke, "SUS-A quick and dirty usability scale," *Usability evaluation in industry,* vol. 194, no. 189, pp. 4-7, 1996.

[50] J. Sauro and J. R. Lewis, "Correlations among prototypical usability metrics: evidence for the construct of usability," in *CHI '09 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, USA, 2009.

[51] J. L. J. Sauro, "Correlations among prototypical usability metrics: evidence for the construct of usability, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems," *ACM*, pp. 1609-1618, 2009.

Appendix A - API Reference

Blogs

Method Name	Retrieve all blogs
Request	GET http://83.212.124.52:8080/SemanticWS-1.5/rest/blogs/retrieve/all
Head Parameters	Filter, Ordering
Query Parameters	-

Method Name	Retrieve blog by ArticleId
Request	GET http://83.212.124.52/SemanticWS-1.5/rest/blogs/retrieve?ArticleId=[article Id]
Head Parameters	-
Query Parameters	ArticleId

Method Name	Retrieve blog by AuthorId
Request	GET http://83.212.124.52/SemanticWS-1.5/rest/blogs/retrieveByAuthor?authorId=[author id]
Head Parameters	-
Query Parameters	AuthorId

Method Name	Insert new blog
Request	PUT http://83.212.124.52/SemanticWS-1.5/rest/blogs/insert
Head Parameters	-
Query Parameters	-
Request Body (JSON)	[{ "thumburl": "value", "blog_name": "value", "url": "value"}]

Authors

Method Name	Retrieve all authors
Request	GET http://83.212.124.52:8080/SemanticWS-1.5/rest/authors/retrieve/all
Head Parameters	Filter, Ordering
Query Parameters	-
Method Name	Retrieve author by ArticleId
Request	GET http://83.212.124.52:8080/SemanticWS-1.5/rest/authors/retrieve?ArticleId=[article Id]
Head Parameters	-
Query Parameters	ArticleId
Method Name	Insert new author
Request	PUT http://83.212.124.52:8080/SemanticWS-1.5/rest/authors/insert
Head Parameters	-
Query Parameters	-
Request Body (JSON)	[{ "author_fullname": "value", "author_nickname": "value", "author_thumb_url": "value", "author_email": "value" }]

Articles

Method Name	Retrieve all articles
Request	GET http://83.212.124.52:8080/SemanticWS-1.5/rest/articles/retrieve/all
Head Parameters	Filter, Ordering
Query Parameters	-
Method Name	Retrieve article by ArticleId
Request	GET http://83.212.124.52:8080/SemanticWS-1.5/est/articles/retrieve

	?ArticleId=[article Id]
Head Parameters	-
Query Parameters	ArticleId

Method Name	Insert new article
Request	PUT http://83.212.124.52:8080/SemanticWS-1.5/rest/articles/insert
Head Parameters	-
Query Parameters	-
Request Body (JSON)	[{ "article_rating": "value", "article_body": "value", "article_tags": "value", "article_url": "value", "article_categories": "value", "article_date_updated": "value", "article_title": "value", "article_thump_url": "value", "article_date_created": "value", "author" : "Author UUID value", "blog" : "Blog UUID value" }]

Comments

Method Name	Retrieve comments by ArticleId
Request	GET http://83.212.124.52:8080/SemanticWS-1.5/rest/comments/retrieve ?ArticleId=[article Id]
Head Parameters	-
Query Parameters	ArticleId

Appendix B - Evaluation schema

The selected sub-characteristics for the evaluation form and their translation into simple questions

Functionality	Suitability	Can software perform the tasks required?
	Accurateness	Is the result as expected?
	Compliance	Is the system compliant with standards?
Efficiency	Time Behavior	How quickly does the system respond?
	Resource utilization	Does the system utilize resources efficiently?
Compatibility	Co-existence	Can the system share resources without loss of its functionality?
	Interoperability	Can the system share information/data with other components?
Usability	Understandability	Does the user comprehend how to use the system easily?
	Learnability	Can the user learn to use the system easily?
	Operability	Can the user use the system without much effort?
	Attractiveness	Does the interface look good?
Reliability	Maturity	Have most of the faults in the software been eliminated over time?
	Fault tolerance	Is the software capable of handling errors?
	Recoverability	Can the software resume working & restore lost data after failure?
Security	Authenticity	Does the system provide identification access wherever is needed?
	Confidentiality	Are data accessible only to authorized users?
	Accountability	Can the system trace actions uniquely?
	Integrity	Does the system prevent unauthorized access?
Maintainability	Analyzability	Can faults be easily diagnosed?

	Changeability	Can the software be easily modified?
	Stability	Can the software continue functioning if changes are made?
	Testability	Can the software be tested easily?
Portability	Adaptability	Can the software be moved to other environments?
	Installability	Can the software be installed easily?
	Conformance	Does the software comply with portability standards?
	Replaceability	Can the software easily replace other software?
Quality of use	Effectiveness	How accurate and complete is the software for the intended use?
	Efficiency	How accurate and complete is the software for the intended use?
	Satisfaction	Does the software satisfy the perceived achievements of pragmatic goals?
	Health and safety risk	Can the software harm people in the intended contexts of use?