Mazil

Anirudh Nain (B.Tech. LNMIIT)
Hemant Jain (B.Tech. LNMIIT

Mentor:
Dr. Debajyoti Bera, IIIT-Delhi

***Abstract*—This document describes Mazil, its features and internal working in detail, and how it makes searching for e-mails using complex queries efficient and simple.**

I. Introduction

Mazil is a web based smart mail search that provides an user extensive mail search platform to perform complex queries.

It uses Faceted Search with SPARQL [6] to give faster search results when searching for e-mails.

It uses **IMAP** (Internet Message Access Protocol) [2] protocol for e-mail retrieval and then store e-mails in the form of RDF (Resource Description Framework) [3] triples using the Apache Jena [4] TDB component for high performance disk storage on PC and provides a jsp browser-based search UI which uses SPARQL [6] for searching in the database.

II. Related Applications

Right now there are various softwares, provided by different vendors, on internet which lets you store and search e-mails . Some of these that are free as well as paid software and applications are:

- X1 Search 8
- CloudMagic
- Microsoft Outlook
- FindIt

III. Internals of Mazil

This section describes the internal details of how Mazil has been implemented.
Mazil can be obtained from Mazil project on github.

A. Architecture

Mazil is a client-server application, hence it requires a server side application(e.g. Apache Tomcat[1]) and a client like application/browser(e.g. Chrome, Firefox, IE) to work. The flowchat below describes the basic layout of Mazil.

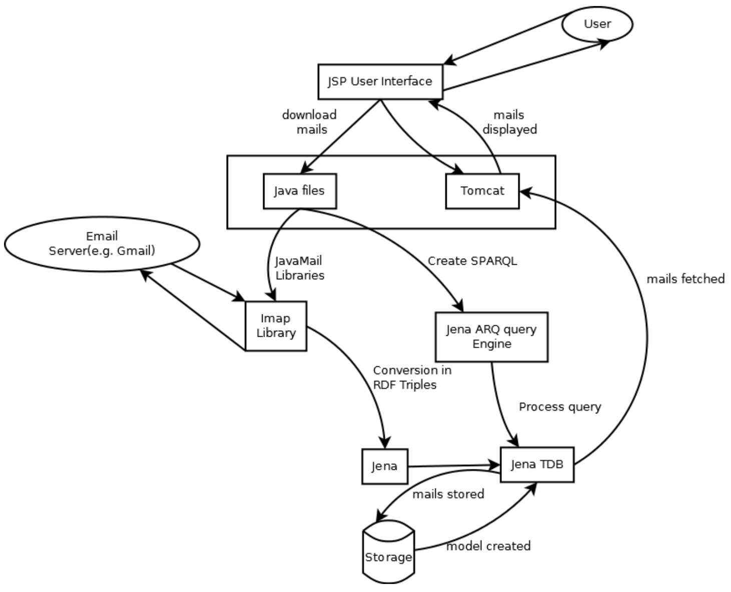


Fig. 1. Basic Layout Of Mazil

The first page of Mazil is an Authentication page that requests the user to provide his/her gmail username and password. Once the authentication is completed, the user is redirected to the querying page where he can filter his e-mails depending upon various options provided to him; while at the backend Mazil download all the e-mails from the user's Gmail account. Mazil synchronises well with the user's gmail account, which means that when the user logins using his gmail username and password Mazil starts downloading only those e-mails that were send to him after he logged out of the application. This functionality is achieved by storing the message ID of the last send mail as well as the UIDvalidity value of the Folder it was send to and later reading the value at the time of e-mail download.

The e-mails stored and queried in form of RDF triples [3] using Apache Jena TDB component and SPARQL [6] in order to boost the search functionality of Mazil, as in Apache Jena all the information provided by a collection of RDF triples is contained in a data structure called a Model which denotes a RDF graph, as it contains a collection of RDF nodes attached to each other by labelled relations. Later, this stored database is queried upon using SPARQL [6] and the result is displayed to the user.

The user can filter the e-mails displayed initally by using number of options provided to him in the query page: some of which include filtering by Content, Subject, Sender's Name Email-ID, Receiver's Name Email-ID, Attachment name, etc. A number of other options like remembering the current query result and again using it later when searching for e-mails, removing the current filters, etc. are also provided to help the user find e-mails quickly and mostly in ways no other e-mail searching application currently provides.

To achieve this type of searching efficiency SPARQL [6] is used to take full advantage of the RDF graph storage [3] as complex queries can be easily written and implemented using SPARQL [6]. As SPARQL queries on a graph type database, the amount of time it takes for the output to be displayed based on the query executed is minimal and the returned result set is very well organized.

B. External Libraries

As Mazil is build using java, various open-source external libraries were used to achieve wide range of tasks needed for the application development and its functioning. All the external libraries that were used are :

- **Xerces** – It is the next generation of high performance, fully compliant XML parsers in the Apache Xerces family. This new version of Xerces introduces the Xerces Native Interface (XNI), a complete framework for building parser components and configurations that is extremely modular and easy to program.part of jena package.
- **xml-commons** provides an Apache-hosted set of DOM, SAX, and JAXP interfaces for use in other xml-based projects. The External Components portion of xml-commons contains interfaces that are defined by external standards organizations. For DOM, that's the W3C; for SAX it's David Megginson and sax.sourceforge.net; for JAXP it's Sun.part of jena package.
- **StAX API** – It is the standard java XML processing API defined by JSR-173.part of tdb.
- **The Simple Logging Facade for Java** (SLF4J) serves as a simple facade or abstraction for various logging frameworks (e.g. java.util.logging, logback, log4j) allowing the end user to plug in the desired logging framework at deployment time.part of jena.
- **Mail-API**
  - **The JavaMail API** provides a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API is available as an optional package for use with the Java SE platform and is also included in the Java EE platform.
- With **log4j** it is possible to enable logging at runtime without modifying the application binary. The log4j package is designed so that these statements can remain in shipped code without incurring a heavy performance cost. Logging behavior can be controlled by editing a configuration file, without touching the application binary.Logging equips the developer with detailed context for application failures. On the other hand, testing provides quality assurance and confidence in the application. Logging and testing should not be confused. They are complementary. When logging is wisely used, it can prove to be an essential tool.
- **Jena-TDB** – TDB is a component of Jena for RDF storage and query. It support the full range of Jena APIs. TDB can be used as a high performance RDF store on a single machine. This documentation describes the latest version, unless otherwise noted.A TDB store can be accessed and managed with the provided command line scripts and via the Jena API. When accessed using transactions a TDB dataset is protected against corruption, unexpected process terminations and system crashes.
- **The Jena IRI Library** is an implementation of RFC 3987 (IRI) and RFC 3986 (URI), and a partial implementation of other related

standards.The IRI module provides an implementation of the IRI and URI specifications (RFC 3987 and 3986) which are used across Jena in order to comply with relevant W3C specifications for RDF and SPARQL which require conformance to these specifications.

- **ARQ** - ARQ is a query engine for Jena that supports the SPARQL RDF Query language. SPARQL is the query language developed by the W3C RDF Data Access Working Group.
  ARQ Features :
  1) Standard SPARQL
  2) Free text search via Lucene
  3) SPARQL/Update
  4) Access and extension of the SPARQL algebra
  5) Support for custom filter functions
  6) Property functions for custom processing of semantic relationships
  7) Aggregation, GROUP BY and assignment as SPARQL extensions
  8) Support for federated query
  9) Support for extension to other storage systems
  10) Client-support for remote access to any SPARQL endpoint
- **Jena**
  - **Apache Jena** (or Jena in short) is a free and open source Java framework for building semantic web and Linked Data applications. The framework is composed of different APIs interacting together to process RDF data. If you are new here, you might want to get started by following one of the tutorials. You can also browse the documentation if you are interested in a particular topic.

These are the parts of jena package: commons-

codec-1.6.jar jcl-over-slf4j-1.6.4.jar jena-iri-1.0.1.jar log4j-1.2.16.jar xercesImpl-2.11.0.jar httpclient-4.2.3.jar jena-arq-2.11.1.jar jena-sdb-1.4.1.jar slf4j-api-1.6.4.jar xml-apis-1.4.01.jar httpcore-4.2.2.jar jena-core-2.11.1.jar jena-tdb-1.0.1.jar slf4j-log4j12-1.6.4.jar

C. User Interface

Mazil is designed to give the user a clean and responsive website, so that the user can get used to Mazil very quickly and can get his/her way around Mazil and start performing complex queries. In order to achieve this Twitter bootstrap [5] was used for designing purpose so that website can be clean as well as responsive without missing out on any options that are required to give the user a clean simple interface.

Though using Mazil is straightforward, the following user interface manual is provided so that user gets to know all the features and the functionality of Mazil and start using it to its full potential.

1) Home Page: The home page is a basic login page where the user has to provide his/her gmail username and password.This is followed by username and password verification and if all details entered by the user are correct the user is redirected to the query page or if the entered username or password is wrong the user is redirected to the home page with authentication error printed.

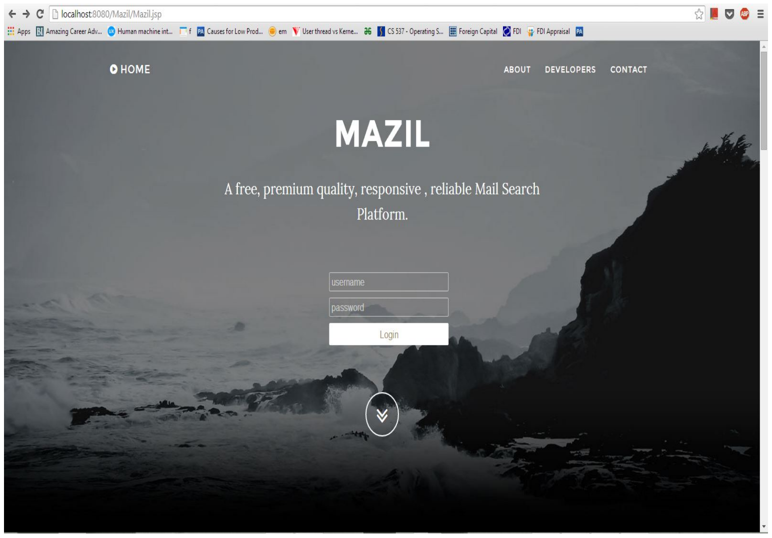
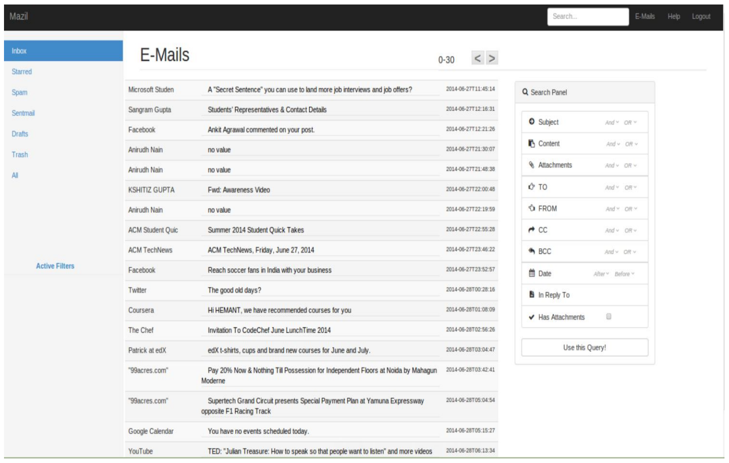

localhost:8080/Mazil/Mazil.jsp
HOME
ABOUT
DEVELOPERS
CONTACT
MAZIL
A free, premium quality, responsive , reliable Mail Search Platform.
username
password
Login

Fig. 2. Home Page

2) Query Page: The query page is where the user view the e-mails along with options provided form him/her to filter through his e-mails. Various options are provided here so that all possible queries that a user can think of can be covered.

a) Mail folders: In this page the e-mails that are present inside the users Inbox Folder are displayed. Options are given on the left panel for the user to choose which Mail folder he wants to view. An image of the options provided is given below.

b) Search Panel: The searching panel is provided on the right side of the webpage. The panel provides the user with options such as filter by subject name along with “and/or” options so that e-mails can be



Microsoft Studen	A "Secret Sentence" you can use to land more job interviews and job offers?	2014-06-27T11:45:14
Sangram Gupta	Students' Representatives & Contact Details	2014-06-27T12:16:31
Facebook	Ankit Agrawal commented on your post.	2014-06-27T12:21:26
Anirudh Nain	no value	2014-06-27T21:30:07
Anirudh Nain	no value	2014-06-27T21:48:38
KSHITIZ GUPTA	Fwd: Awareness Video	2014-06-27T22:00:48
Anirudh Nain	no value	2014-06-27T22:19:59
ACM Student Quic	Summer 2014 Student Quick Takes	2014-06-27T22:55:28
ACM TechNews	ACM TechNews, Friday, June 27, 2014	2014-06-27T23:46:22
Facebook	Reach soccer fans in India with your business	2014-06-27T23:52:57
Twitter	The good old days?	2014-06-28T00:28:16
Coursera	Hi HEMANT, we have recommended courses for you	2014-06-28T01:08:09
The Chef	Invitation To CodeChef June LunchTime 2014	2014-06-28T02:56:26
Patrick at edX	edX t-shirts, cups and brand new courses for June and July.	2014-06-28T03:04:47
"99acres.com"	Pay 20% Now & Nothing Till Possession for Independent Floors at Noida by Mahagun Moderne	2014-06-28T03:42:41
"99acres.com"	Supertech Grand Circuit presents Special Payment Plan at Yamuna Expressway opposite F1 Racing Track	2014-06-28T05:04:54
Google Calendar	You have no events scheduled today.	2014-06-28T05:15:27
YouTube	TED: "Julian Treasure: How to speak so that people want to listen" and more videos	2014-06-28T06:13:34

Fig. 3. Query Page

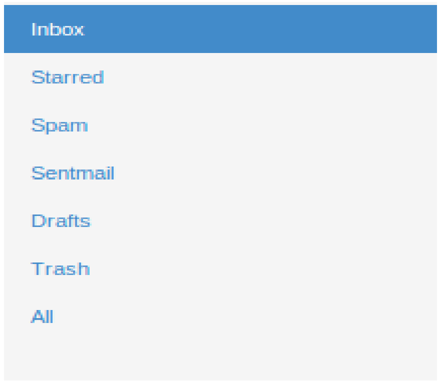


Fig. 4. Mail Folders

filtered by multiple subject names e.g. user can query his e-mails by “Subject:Phd” and after choosing “and option” the filtering changes to “Subject:Phd and Admission” so all those e-mails that contain the Subject with words “Phd” **as well as”** “Admission” are displayed. These filters are provided uniformly on all the querying options.

The searching panel contains other options like filtering through date, where a calendar is displayed for easy input as well as the user can filter through the attachment name with and/or options provided or can search e-mails that have attachments by liking on the checkbox given at the end of the searching panel.

At the very end of the searching panel the user is provide with a option of “Search among current results!”. When this option is used Mazil remembers the output of the current query and the user can later filter on the result set by applying the remembered query.

Right above the searching panel there is a search option which is provided so that the user can filter his e-mails not by a particular part of the mail(e.g. Subject, Content) but through the whole mail.

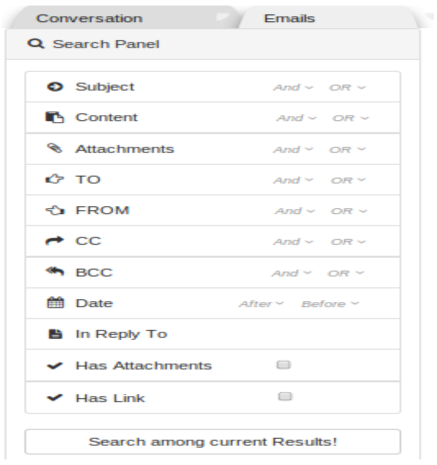


Fig. 5. Search Panel

c) Conversation: This is one of the trademark feature of Mazil which makes it even more convenient and handy .With this unique feature you can check your conversation on just a click .It provides searching options tailor made for searching conversations where you can enter details of any part of conversation ,any specific details of any mail that took part in the conversation and Mazil will search that conversation instantly.This trait of Mazil provides it more functionality and edge over other mail clients and applications.

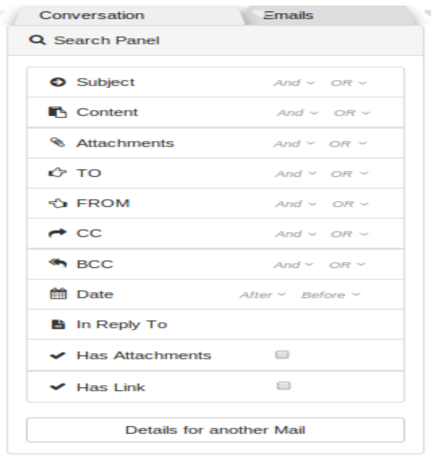


Fig. 6. Search Panel

d) Active Filters: When the user chooses an option to filter his e-mails, all the active filters are added in the panel on the lower-left side of the screen , where the user can view the filters applied by him. The user is provided the option of removing the current filters by clicking on the cross button next to the filter name. After clicking on this the e-mails are refreshed and the applied filter is removed from the result set.

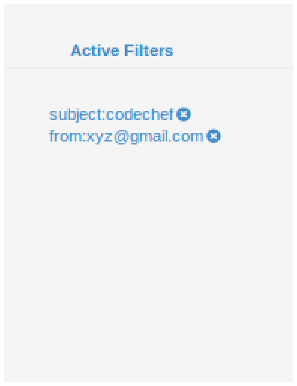


Fig. 7. Active Filters

IV. Usage

To start using Mazil you need to have a Software that can do the role of a server, a browser and internet. To make the life of user simpler a step-by-step installation guide is oulined below.

1) Download Apache Tomcat. (http://tomcat.apache.org/download-80.cgi)
2) Install Apache Tomcat.
3) Deploy the war file inside the webapps directory of Tomcat.(e.g. C:FilesSoftware Foundation8.0)
4) Start the server(e.g. Tomcat)
5) Start your Web Browser (e.g. Chrome, Firefox)
6) In the Address Bar type the url of Mazil home page.(e.g. http://localhost:8080/Mazil/Index.jsp)
7) Start using Mazil.

V. Evaluation

The difference between Mazil and Google Mail Search is that it can perform complex queries that cannot be done using Gmail and thus would result in savage of a whole lot of time. Suppose, if we want to search for emails which **contains attachment and which were sent to me by people who at some point sent me emails with subject “xyz**. Normally one will use gmail and query for mails which either contains attachment or which have subject “xyz . This would result in plethora of mails in which you have to go through every mail to get your desired emails . This is the area where Mazil comes into play. It will show you in result the exact mails you wanted and thus saving you lot of time and trouble. This was a general query and there are many other searches we come across in day to day work which almost consume 10-20 mins of our Busy life per query. With Mazil life would be much more simpler with emails searching more extensive and powerful .

Another example of query can be : **searching for a conversation of emails in which one email contains word “xyz and another email contains word “abc**. Conventionally on Gmail one will either search for emails which contains word “xyz or emails with word “abc .Then again it would also result in large number of emails which would be useless.

Mazil takes about 10-20 MB of your Disk Space for storing 100 emails i.e. about a tenth of a MB for a single email which is not much as compared to size of emails.

The time taken by programs to query from the database largely depends on the complexity of the queries.While simple queries requiresless than 50ms while complex queries may end up taking about 200-400ms. The time percieved by the user which includes the time taking the inputs, processing them, bulding up queries, getting results from database and displaying them range drom 700-800ms.

The section below shows some sample queries along with the time of execution to give a rough idea about the efficiency of Mazil.

1) SELECT ?x ?ax ?bx ?cx WHERE {
?a $\langle SUB:\rangle ?x.$
$?a\langle DATE:\rangle ?ax.$
$?a\langle SENDERNAME:\rangle ?bx.$
$?a\langle MESSAGEID:\rangle ?cx.$
$?a\langle FOLDERNAME:\rangle `inbox'$
} LIMIT 30 OFFSET 0

Objective – To Select the Subject, Date, Sender’s Name and Message Id of the email contained in the Inbox folder.

Time taken : 39ms

2) SELECT DISTINCT ?x ?ax ?bx ?cx WHERE {

?a $\langle SUB:\rangle ?x.$
$?a\langle DATE:\rangle ?ax.$
$?a\langle SENDERNAME:\rangle ?bx.$
$?a\langle MESSAGEID:\rangle ?cx.$
$?a\langle REFERENCES:\rangle ?z.$
$FILTERregex(?z,'<','i')$
} LIMIT 30 OFFSET 0

Objective – To Select the Subject, Date, Sender's Name and Message Id of the converstions.

Time taken: 82ms

3) SELECT ?x ?ax ?bx ?cx WHERE {
?a $\langle SUB:\rangle ?x.$
$?a\langle DATE:\rangle ?ax.$
$?a\langle SENDERNAME:\rangle ?bx.$
$?a\langle MESSAGEID:\rangle ?cx.$
$?a\langle FROM:\rangle ?u.$
{
$SELECT(?xas?u)WHERE$
{
$?a\langle FROM:\rangle ?x.$
$?a\langle SUB:\rangle ?q.$
$FILTERregex(?q,'abc','i')\}LIMIT30\}.$
$?a\langle CONTENT:\rangle ?q.$
$FILTERregex(?q,'xyz','i')$
} LIMIT 30 OFFSET 0

Objective – To Select the Subject, Date, Sender's Name and Message Id of the mails which have subject abc and were sent to me by people who at some point sent me mails that contained word 'xyz'.

Time taken:92ms

4) SELECT DISTINCT ?x ?ax ?bx ?cx WHERE {
?a $\langle SUB:\rangle ?x.$
$?a\langle DATE:\rangle ?ax.$
$?a\langle SENDERNAME:\rangle ?bx.$
$?a\langle MESSAGEID:\rangle ?cx.$
$\{?a\langle SUB:\rangle ?z.$
$FILTERregex(?z,\text{“abc”},'i')\}$
$UNION$
$\{?a\langle CONTENT:\rangle ?ab.$
$FILTERregex(?ab,\text{“abc”},'i')\}$
$UNION$
$\{?a\langle FROMFULL:\rangle ?b.$
$FILTERregex(?b,\text{“abc”},'i')\}$
} LIMIT 30 OFFSET 0

Objective – To Select the Subject, Date, Sender's Name and Message Id of the emails which either have sendername,subject,content have word 'abc'.

Time taken:243ms

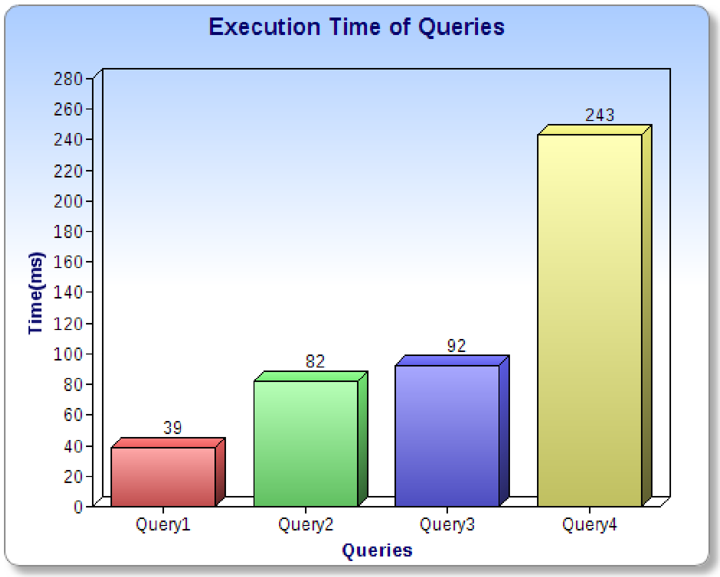


Fig. 8. Time taken for queries to execute

VI. Conclusion

Mazil is powerful, extensive and efficient mail searching platform desgined to help the user find e-mails with least amount of information available with him/her without wasting his time in filtering them manually as on other mail clients. To achieve this task e-mails are stored in form of RDF triples [3] using the Jena TDB component and then the stored database is queried upon using SPARQL [6]. The UI of Mazil was build using JSP along with Twitter bootstrap to build a dynamic and responsive website.

Right now Mazil is just a searching application that provides you with powerful and extensive searching options. In Future Mazil can further be improved on UI, queries, Disk Usage and could even contain an option for sending e-mails so that it can act as fully operational Email Client with immense searching options .

References

[1] Apache tomcat. http://tomcat.apache.org/.

[2] Internet message access protocol. http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol.

[3] Rdf concepts and abstract syntax. http://www.w3.org/TR/rdf11-concepts/.

[4] Apache Jena. Apache jena, 2013.

[5] Mark Otto and Jacob Thornton. Bootstrap. *Twitter Bootstrap*, 2010.

[6] Eric PrudHommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.