



**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCE  
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION**

**GRADUATE THESIS**

**merGeo: Integration Platform for Linked Data Management  
Tools**

**Christos N Papaloukas**

**Supervisor: Manolis Koubarakis, Professor UoA**  
**Co-Supervisor: Georgios Stamoulis, Ph.D. Candidate UoA**

**ATHENS**

**OCTOBER 2017**



**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

**merGeo: Πλατφόρμα Ενοποίησης Εργαλείων Διαχείρισης  
Διασυνδεδεμένων Δεδομένων**

**Χρίστος Ν Παπαλουκάς**

**Επιβλέπων: Μανόλης Κουμπάρακης, Καθηγητής ΕΚΠΑ  
Συνεπιβλέπων: Γεώργιος Σταμούλης, Υποψήφιος Διδάκτωρ ΕΚΠΑ**

**ΑΘΗΝΑ**

**ΟΚΤΩΒΡΙΟΣ 2017**

## **GRADUATE THESIS**

merGeo: Integration Platform for Linked Data Management Tools

**Christos N Papaloukas**

**S.N.:** 1115201100086

**Supervisor:** **Manolis Koubarakis**, Professor UoA  
**Co-Supervisor:** **Georgios Stamoulis**, Ph.D. Candidate UoA

## **ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

merGeo: Πλατφόρμα ενοποίησης εργαλείων διαχείρισης Διασυνδεδεμένων Δεδομένων

**Χρίστος Ν Παπαλουκάς**

**A.M.: 1115201100086**

**Επιβλέπων:** Μανόλης Κουμπάρκης, Καθηγητής ΕΚΠΑ  
**Συνεπιβλέπων:** Γεώργιος Σταμούλης, Υποψήφιος Διδάκτωρ ΕΚΠΑ

## **ABSTRACT**

Nowadays, Semantic Web represents the next major evolution in connecting information. It enables data to be linked from one source to any other source on the web and to be understood by computers so that they can perform increasingly sophisticated tasks on our behalf. With this in mind, researchers and practitioners have engineered many tools that focus on manipulating this abundance of new web data. Although this managing process is quite demanding and complicated, with the use of appropriate tools that concentrate on transformation, exploration or visualization of these data, this whole handling process could become simpler. However, till now each tool focuses only on a single one of the aforementioned procedures and as a result, a combination of them could be used to take the linked data manipulation to the next level.

The objective of this thesis is to contribute to linked data management, exploration and visualization process by engineering merGeo, a web application which is focused on providing a user-friendly platform that combines the features of three already known linked data management tools: GeoTriples, Strabon and Sextant. MerGeo provides an easily operated platform that allows both to domain experts and daily users to take full advantage of semantic web tools and technologies and also convinces them to embrace these tools and technologies by demonstrating the benefits of using together different applications whose center of attention is linked data.

**SUBJECT AREA:** Semantic web, Linked data management, Software engineering

**KEYWORDS:** Data transformation, Data exploration, Data visualization, Linked data, Spring MVC

## ΠΕΡΙΛΗΨΗ

Στις μέρες μας, ο Σημασιολογικός Ιστός αντιπροσωπεύει την επόμενη σημαντική εξέλιξη στη σύνδεση πληροφοριών. Επιτρέπει τη σύνδεση δεδομένων από μια πηγή σε οποιαδήποτε άλλη πηγή στον ιστό, καθώς και την κατανόηση των δεδομένων αυτών από τους υπολογιστές, ώστε να μπορούν να εκτελούν ολοένα και πιο εξελιγμένες διεργασίες για λογαριασμό μας. Έχοντας αυτό κατά νου, οι ερευνητές και οι επαγγελματίες έχουν δημιουργήσει πολλά εργαλεία που επικεντρώνονται στη διαχείριση αυτής της αφθονίας των νέων δεδομένων ιστού. Αν και αυτή η διαδικασία διαχείρισης είναι αρκετά απαιτητική και περίπλοκη, με τη χρήση των κατάλληλων εργαλείων που επικεντρώνονται στη μετατροπή, την εξερεύνηση ή την οπτικοποίηση αυτών των δεδομένων μπορεί να γίνει πιο απλή. Ωστόσο, μέχρι στιγμής κάθε εργαλείο επικεντρώνεται σε μόνο μία από τις προαναφερθείσες διαδικασίες και ως αποτέλεσμα, ο συνδυασμός αυτών των εργαλείων θα μπορούσε να χρησιμοποιηθεί για να αναβαθμιστεί ο τρόπος διαχείρισης των διασυνδεδεμένων δεδομένων.

Σκοπός της παρούσας εργασίας είναι να συμβάλει στη διαδικασία διαχείρισης, διερεύνησης και οπτικοποίησης των διασυνδεδεμένων δεδομένων με την ανάπτυξη του merGeo, μιας εφαρμογής που επικεντρώνεται στην παροχή μιας εύχρηστης πλατφόρμας η οποία συνδυάζει τις δυνατότητες τριών ήδη γνωστών εργαλείων διαχείρισης διασυνδεδεμένων δεδομένων: του GeoTriples, του Strabon και του Sextant. Το merGeo προσφέρει μια φιλική προς το χρήστη εφαρμογή που επιτρέπει τόσο σε ειδήμονες του αντικείμενου όσο και σε μη ειδικούς να εκμεταλλευτούν εργαλεία και τεχνολογίες του σημασιολογικού ιστού και να τους πείσει να υιοθετήσουν αυτές τις τεχνολογίες παρουσιάζοντας τα οφέλη του να χρησιμοποιείς διαφορετικές εφαρμογές που βασίζονται σε διασυνδεδεμένα δεδομένα.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:** Σημασιολογικός ιστός, Διαχείριση διασυνδεδεμένων δεδομένων, Τεχνολογία λογισμικού

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:** Μετασχηματισμός δεδομένων, Εξερεύνηση δεδομένων, Οπτικοποίηση δεδομένων, Διασυνδεδεμένα δεδομένα, Spring MVC

## **AKNOWLEDGMENTS**

First of all I would like to acknowledge my supervisor Prof. Manolis Koubarakis. His course “Artificial Intelligence (I)” and several lectures from his postgraduate course “Knowledge Technologies” have directly contributed to discover my personal interest in AI and assisted me in selecting this specific topic. I would also like to acknowledge PhD candidate Georgios Stamoulis for his valuable assistance and guidance throughout the whole thesis implementation.

# CONTENTS

<b>1. INTRODUCTION</b> .....	<b>10</b>
1.1 Contribution of the thesis .....	11
1.2 Thesis outline .....	12
<b>2. BACKGROUND</b> .....	<b>13</b>
2.1 Resource description framework (RDF) .....	13
2.2 The RDF query languages SPARQL & GEOSPARQL .....	15
2.3 GeoTriples – A data transformation tool .....	16
2.4 Strabon – A spatiotemporal RDF store .....	18
2.5 Sextant - Visualizing linked geospatial data .....	19
<b>3. MERGEO: ENGINEERING THE PLATFORM OF LINKED DATA TOOLS</b> .....	<b>21</b>
3.1 MerGeo workflow in details .....	21
3.2 Architecture .....	23
3.2.1 Spring MVC Framework .....	23
3.2.2 Server analysis .....	25
3.2.3 Client analysis .....	26
<b>4. DEMONSTRATION OF MERGEO WEB PLATFORM THROUGH USE CASES</b> ..	<b>28</b>
4.1 Use case in a detailed description .....	28
4.2 Additional features .....	35
<b>5. CONCLUSIONS AND FUTURE WORK</b> .....	<b>37</b>
<b>ANNEX A INSTALLATION INSTRUCTIONS</b> .....	<b>38</b>
<b>REFERENCES</b> .....	<b>40</b>



## LIST OF FIGURES

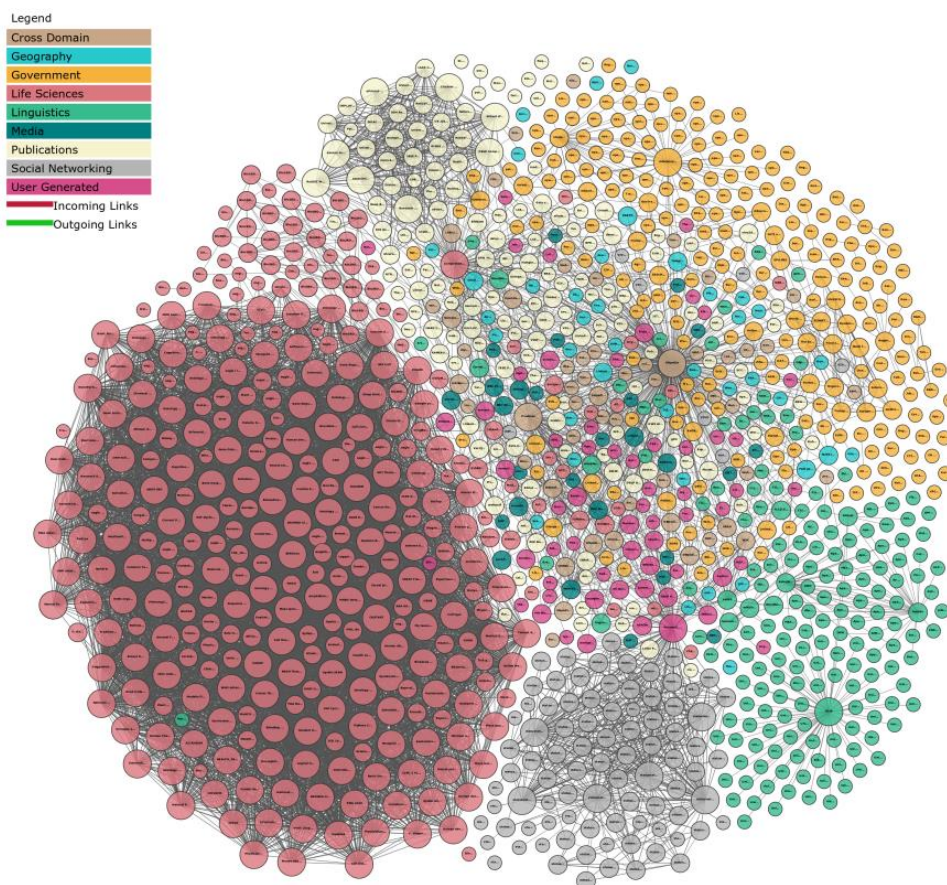
Figure 1.1 Recent state of the Linked Open Data cloud .....	10
Figure 2.1 Simple example of triples.....	14
Figure 2.2 Graph of the above example .....	14
Figure 2.3 GeoTriples architecture .....	17
Figure 2.4 Strabon Architecture.....	18
Figure 2.5 Sextant Architecture .....	20
Figure 3.1 Work-flow of the Spring Web MVC architecture .....	23
Figure 3.2 Spring MVC Framework architecture within merGeo.....	24
Figure 3.3 merGeo architecture.....	26
Figure 3.4 merGeo - Home page.....	27
Figure 3.5 merGeo - Management tools pages .....	27
Figure 4.1 merGeo - Strabon dropdown menu .....	28
Figure 4.2 merGeo - Strabon Endpoint creation form.....	29
Figure 4.3 merGeo - Endpoint main View.....	29
Figure 4.4 merGeo - Geotriples 1/3 step .....	30
Figure 4.5 merGeo - GeoTriples 2/3 step .....	30
Figure 4.6 merGeo - GeoTriples 3/3 step .....	31
Figure 4.7 merGeo - Strabon main after Store .....	32
Figure 4.8 merGeo - Strabon query results (HTML) .....	33
Figure 4.9 merGeo - Strabon query results (XML).....	33
Figure 4.10 merGeo - Strabon results displayed on map .....	34
Figure 4.11 merGeo - Strabon direct store .....	35
Figure 4.12 merGeo - Strabon store via file URI.....	35
Figure 4.13 merGeo - Sextant UI.....	36

## 1. INTRODUCTION

Nowadays, semantic web and especially linked data are receiving increasing attention as developers and researchers are using advanced semantic web technologies to structure and represent web data. The aspiration of this whole process is to allow people to share structured data on the web as easily as they already can do with documents and furthermore, to interlink them with other available data in order to increase their value for final users [1].

Consequently, this growing rate of new web data, especially geospatial, and their use across many social, environmental topics and more, has led to the need of more progressive applications and tools that focus on transforming, manipulating and exploring linked datasets. In this procedure, an important step is the initial transformation of the data from any form that they exist into a common format, the Resource Description Framework (RDF) in order to be easily integrated with other already transformed data.

All the available RDF data that are compatible with the Linked Data Principles<sup>1</sup> compose the Linked Open Data (LOD) cloud which is shown below (Figure 1.1).



**Figure 1.1 Recent state of the Linked Open Data cloud**

<sup>1</sup> <http://www.w3.org/DesignIssues/LinkedData.html>

The Resource Description Framework (RDF) is a directed, labeled graph data format for representing information (especially metadata) about resources in the Web. Resources can be anything, including documents, people, physical objects, and abstract concepts.

RDF is based on the idea of identifying resources using Web identifiers and describing resources in terms of simple properties and property values. To identify resources, RDF uses Uniform Resource Identifiers (URIs) and URI references (URIrefs). The resources being described have properties which have values, and that resources can be described by making statements, that specify those properties and values.

As we understand, in terms of the semantic web, having all the available data in a commonly used format is of high importance. GeoTriples<sup>2</sup> is a tool that fulfills this necessity as it is able to transform data into linked RDF data from many original formats and so, it enables us to start building the advantageous Semantic network.

Focusing specifically on spatial and temporal linked data, new extensions to RDF have been proposed and implemented, providing many new capabilities to these data subjects. GeoSPARQL [2] is a recent OGC standard which allows representing and querying geospatial data on the Semantic Web. Also, the data model stRDF accompanied by the query language stSPARQL [3] are extensions of the standard RDF and SPARQL for representing and querying geospatial data that change over time. Both of the above extensions are implemented in the open source spatiotemporal RDF store Strabon [4].

Furthermore, another notable application is Sextant [5, 7, 8]. Sextant was the first one to offer functionalities for supporting geospatial mapping. Also, it can be used to produce thematic maps by layering geospatial and temporal information which exists in a number of heterogeneous data sources ranging from standard SPARQL endpoints to SPARQL endpoints following the standard GeoSPARQL defined by the Open Geospatial Consortium (OGC), or the well-adopted geospatial file format KML [6]. Taking into consideration this whole abundance of data and the urge to have convenient tools to manipulate it, we decided to engineer an application that integrates some of the most critical procedures; the transformation (GeoTriples), the store and exploration (Strabon<sup>3</sup>) and finally the visualization (Sextant<sup>4</sup>) process.

## 1.1 Contribution of the thesis

This thesis constitutes an attempt to implement in practice the Semantic Web's approach about Linked Data manipulation which is to share and reuse Linked Data across different applications. Specifically, the main object is to provide a unified, user-friendly platform for everyone, from domain experts to non-experts at all, where one can transform many data formats to RDF, then store and explore these data by querying them on a SPARQL endpoint and finally, visualize and portray the stored data on a map (provided that these data include geometries).

At first, the main task was to create a web interface for GeoTriples whose whole process is divided into two steps; the generation of a mapping based on the input file

---

<sup>2</sup> <http://geotriples.di.uoa.gr/>

<sup>3</sup> <http://strabon.di.uoa.gr/>

<sup>4</sup> <http://sextant.di.uoa.gr/>

and the transformation of this mapping file into RDF data. We managed to provide user-friendly forms, where the user can set the options for the GeoTriples process and also we allowed the user to interact with the output files on every step and edit them instantly.

After the transformation is completed and the output RDF file is generated, the demand for storage comes in. The transition from GeoTriples to Strabon comes immediately after the final step of GeoTriples process, where the user can either store the RDF data on the default, already running SPARQL Endpoint, or create a new SPARQL Endpoint and store them into that.

So, proceeding to the next step, we had to embody Strabon interface and functionality in our application. We designed a new, handy interface where the user is able to query the Endpoint, view the results in the three most usable formats (HTML, XML, GeoJSON) and even store new data with direct input or via a URI.

The final step was to combine this whole transformation and storing process with Sextant. So, we provided the user the ability to visualize the stored RDF data that include geometries on a map, through some REST calls on the Sextant's API. In addition, Sextant's API can be used independently and full-featured through its exclusive section in our application.

In this manner we manage to expose semantic web technologies and tools to users from various domains, and even convince them to adopt these technologies by presenting the benefits of the linked open web through the use of merGeo.

## 1.2 Thesis outline

The structure of this thesis is organized as follows: In Chapter 2 the background knowledge on the RDF model and the query languages SPARQL and GEOSPARQL is presented. Moreover, we describe the functionality and the architecture of the three management tools that our application is based on; GeoTriples, Strabon and Sextant. In Chapter 3 we analyze the architecture of merGeo. Firstly, a step-by-step insight on the workflow of merGeo is given on how the selected framework (Spring MVC) accomplishes to make our platform more operative and then the components of the server and the client side of the application are described. Chapter 4 is about the demonstration of this platform and its purpose is to navigate the reader of this thesis through the available web services we are offer through this project and also present the site through the scope of use cases. Conclusively in Chapter 5 we summarize the work and discuss possible matters that can be dealt with in future upgrades of merGeo.

## 2. BACKGROUND

Recently, linked geospatial data have received considerable attention as practitioners and researchers have started to make full use of the wealth of geospatial information available online. Therefore, in the last few years semantic web has been enhanced with mature technologies and services that make researchers and practitioners capable of representing geospatial information in RDF and query it using SPARQL. Furthermore, the technological accomplishments of semantic web are offering plenty of tools and systems for querying, publishing, exploring and visualizing geospatial information represented in RDF. In this chapter, the main subject of discussion will be the presentation of concepts related to linked data manipulation along with the fundamental tools that constitute the basic compounds of merGeo functionality.

### 2.1 Resource description framework (RDF)

Resource Description Framework (RDF)<sup>1</sup> is commonly used in cases where information about web resources has to be processed by computers and applications, instead of being only displayed to users. It is also an optimal solution for expressing information that can be transferred between applications without any meaning loss. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created<sup>2</sup>.

The following examples illustrate the various different uses of RDF, aimed at different communities of practice.

- Adding machine-readable information to Web pages using, for example, the popular schema.org vocabulary, enabling them to be displayed in an enhanced format on search engines or to be processed automatically by third-party applications.
- Enriching a dataset by linking it to third-party datasets. For example, a dataset about paintings could be enriched by linking them to the corresponding artists in Wikidata, therefore giving access to a wide range of information about them and related resources.
- Interlinking API feeds, making sure that clients can easily discover how to access more information.
- Using the datasets currently published as Linked Data [LINKED-DATA], for example building aggregations of data around specific topics.
- Building distributed social networks by interlinking RDF descriptions of people across multiple Web sites.
- Providing a standards-compliant way for exchanging data between databases.
- Interlinking various datasets within an organization, enabling cross-dataset queries to be performed using SPARQL

---

<sup>1</sup> <https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>

<sup>2</sup> <https://www.w3.org/TR/rdf11-primer/>

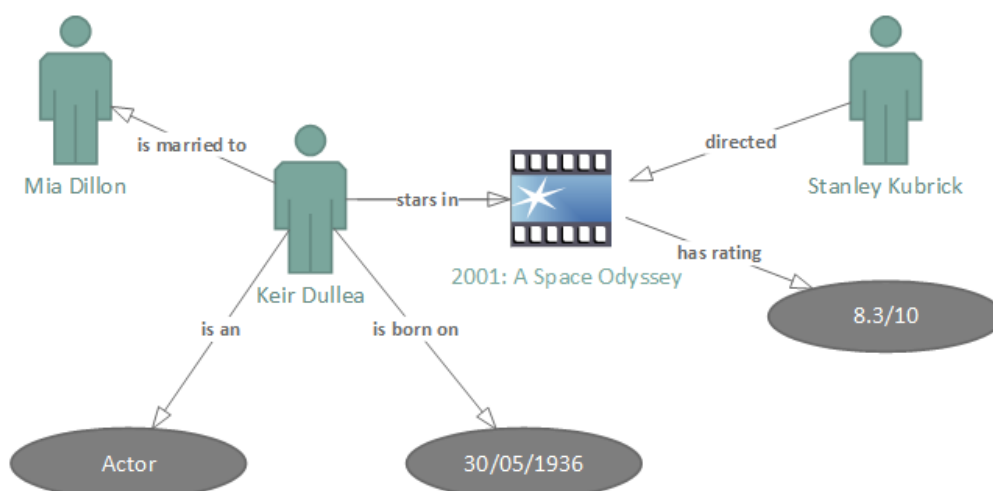
RDF also allows us to make statements about resources. The format of these statements is simple. An RDF statement expresses a relationship between two resources. The subject and the object represent the two resources being related; the predicate represents the nature of their relationship. The relationship is phrased in a directional way (from subject to object) and is called in RDF a property. Because RDF statements consist of three elements they are called triples. The following figures (Figures 2.1 & 2.2) show a simple example of triples and their relations.

```

1 <Keir Dullea> <is an> <actor>.
2 <Keir Dullea> <is married to> <Mia Dillon>.
3 <Keir Dullea> <is born on> <the 30th of May 1936>.
4 <Keir Dullea> <stars in> <2001: A Space Odyssey>.
5 <2001: A Space Odyssey> <has rating> <8.3 out of 10>.
6 <Stanley Kubrick> <directed> <2001: A Space Odyssey>.

```

**Figure 2.1 Simple example of triples**



**Figure 2.2 Graph of the above example**

The same resource is often referenced in multiple triples. In the example above, “Keir Dullea” is the subject of four triples, and “2001: A Space Odyssey” is the subject of one and the object of one triple. This ability to have the same resource be in the subject position of one triple and the object position of another makes it possible to find connections between triples, which is an important part of RDF’s efficiency.

So, once you have a graph like this you can use a RDF query language like SPARQL to query for e.g. actors starring in films directed by Stanley Kubrick.

## 2.2 The RDF query languages SPARQL & GEOSPARQL

As it is already mentioned, RDF is a labeled, directed graph data format used in information representation in the web. This specification defines the syntax and semantics of the SPARQL [9] query language for RDF. SPARQL<sup>3</sup> can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. Furthermore, SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports aggregation, subqueries, negation, creating values by expressions, extensible value testing, and constraining queries by source RDF graph. The results of SPARQL queries can be result sets or RDF graphs.

Most forms of SPARQL query contain a set of triple patterns called a basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph. The following example shows a SPARQL query to find the title of a movie from the given data graph. The applied query consists of two parts: (I) the *SELECT* clause that identifies the variables to appear in the query results and (II) the *WHERE* clause that provides the basic graph pattern to match against the data graph. The basic graph pattern in the following example consists of a single triple pattern with a single variable *?title* in the object position.

```

                                     DATA
-----:
<http://example.org/movie/movie1>
  <http://purl.org/dc/elements/1.1/title> "2001: A Space Odyssey" .

                                     QUERY
-----:
SELECT ?title
WHERE
{
<http://example.org/movie/movie1> <http://purl.org/dc/elements/1.1/title> ?title .
}

                                     RESULTS
-----:
Title
-----
"2001: A Space Odyssey"

```

The OGC GeoSPARQL standard supports representing and querying geospatial data on the Semantic Web. GeoSPARQL defines a vocabulary for representing geospatial data in RDF, and it defines an extension to the SPARQL query language for processing geospatial data. Moreover, GeoSPARQL is designed to accommodate systems based

<sup>3</sup> <http://www.w3.org/TR/sparql11-query/>

on qualitative spatial reasoning and systems based on quantitative spatial computations. The GeoSPARQL standard follows a modular design as it comprises several different components:

- A core component defines top-level RDFS/OWL<sup>4</sup> classes for spatial objects.
- A topology vocabulary component defines RDF properties for asserting and querying topological relations between spatial objects.
- A geometry component defines RDFS data types for serializing geometry data, geometry-related RDF properties, and non-topological spatial query functions for geometry objects.
- A geometry topology component defines topological query functions.
- An RDFS entailment component defines a mechanism for matching implicit RDF triples that are derived based on RDF and RDFS semantics.
- A query rewrite component defines rules for transforming a simple triple pattern that tests a topological relation between two features into an equivalent query involving concrete geometries and topological query functions.

Each of the components described above forms a requirements class for GeoSPARQL. Implementations can provide various levels of functionality by choosing which requirements classes to support. For example, a system based purely on qualitative spatial reasoning may support only the core and topological vocabulary components.

Moreover, GeoSPARQL is designed to accommodate systems based on qualitative spatial reasoning and systems based on quantitative spatial computations. Systems based on qualitative spatial reasoning, do not usually model explicit geometries, so queries in such systems will likely test for binary spatial relationships between features rather than between explicit geometries. To allow queries for spatial relations between features in quantitative systems, GeoSPARQL defines a series of query transformation rules that expand a feature-only query into a geometry-based query. With these transformation rules, queries about spatial relations between features will have the same specification in both qualitative systems and quantitative systems. The qualitative system will likely evaluate the query with an awkward-chaining spatial “reasoner”, and the quantitative system can transform the query into a geometry-based query that can be evaluated with computational geometry.

### 2.3 GeoTriples – A data transformation tool

GeoTriples [12] is a tool for transforming geospatial data from their original formats (e.g., shapefiles or spatially-enabled relational databases) into RDF. The following input formats are supported: spatially-enabled relational databases (PostGIS and MonetDB), ESRI shapefiles and XML, GML [10], KML, JSON, GeoJSON<sup>5</sup> and CSV documents.

Geotriples supports the mapping languages R2RML and RML and extends them for modeling the transformation of geospatial data into RDF graphs. RML is defined as a superset language of R2RML. The strong point of RML, is that is designed to allow the

---

<sup>4</sup> <http://www.w3.org/standards/techs/owl>

<sup>5</sup> <https://tools.ietf.org/html/rfc7946>



process of data that do not necessarily rely in tables and thus not having an explicit iteration pattern.

Focusing on its architecture, GeoTriples comprises two main components: the mapping generator and the R2RML/RML mapping processor. The mapping generator takes as input a geospatial data source (e.g., a shapefile) and creates automatically an R2RML or RML mapping that can transform the input into an RDF graph which uses the GeoSPARQL vocabulary. The user may edit the generated R2RML/RML mapping document to comply with her requirements (e.g., use a vocabulary different than the one of GeoSPARQL). Then, the mapping processor executes the R2RML/RML mappings to produce the output geospatial RDF graph. The mapping processor of GeoTriples comes in two forms: a single-node implementation and an implementation that uses Apache Hadoop for dealing with big geospatial data. The second implementation can be found here<sup>6</sup>.

It is often the case in applications that relevant geospatial data is stored in spatially-enabled relational databases (e.g., PostGIS<sup>7</sup>) or files (e.g., shapefiles), and its owners do not want to explicitly transform it into linked data. For example, this might be because these data sources get frequently updated and/or are very large. If this is the case, GeoTriples is still very useful. GeoTriples users can use the generated mappings in the system Ontop-spatial to view their data sources virtually as linked data. Ontop-spatial is a geospatial Ontology-Based Data Access system which performs on-the-fly GeoSPARQL-to-SQL translation over spatially-enabled relational databases using ontologies and mappings.

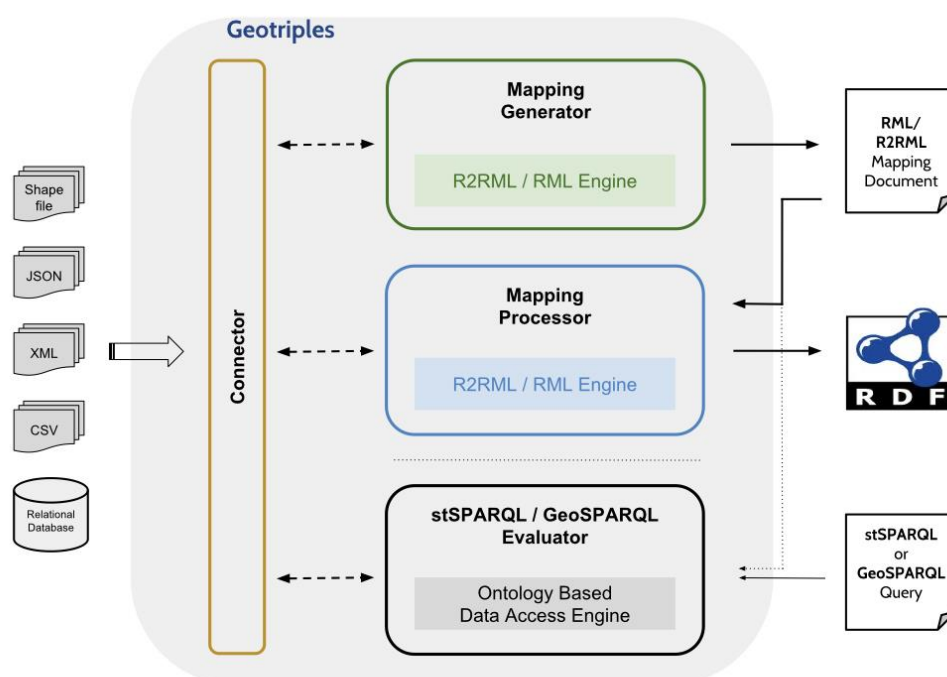


Figure 2.3 GeoTriples architecture

<sup>6</sup> <https://github.com/dimitrianos/GeoTriples-Hadoop>

<sup>7</sup> <http://postgis.net>

## 2.4 Strabon – A spatiotemporal RDF store

Strabon is a spatiotemporal RDF store. You can use it to store linked geospatial data that changes over time and pose queries using two popular extensions of SPARQL. Strabon supports spatial datatypes enabling the serialization of geometric objects in OGC standards WKT [11] and GML. It also offers spatial and temporal selections, spatial and temporal joins, a rich set of spatial functions similar to those offered by geospatial relational database systems and support for multiple Coordinate Reference Systems. Strabon can be used to model temporal domains and concepts such as events, facts that change over time etc. through its support for valid time of triples, and a rich set of temporal functions. Strabon is built by extending the well-known RDF store Sesame (now called RDF4J<sup>8</sup>) and extends RDF4J's components to manage thematic, spatial and temporal data that is stored in the backend RDBMS.

The first query language supported by Strabon is stSPARQL. stSPARQL can be used to query data represented in an extension of RDF called stRDF. stRDF and stSPARQL have been designed for representing and querying geospatial data that changes over time, e.g., the growth of a city over the years due to new developments can be represented and queried using the valid time dimension of stRDF and stSPARQL respectively. The expressive power of stSPARQL makes Strabon the only fully implemented RDF store with rich spatial and temporal functionalities available today.

Strabon has been shown experimentally to be the most efficient spatiotemporal RDF store available today. See the papers *Geographica: A Benchmark for Geospatial RDF Stores*, and *Representing and Querying the valid time of triples for Linked Geospatial Data* for more details of how Strabon outperforms all other geospatial or temporal RDF stores available today including commercial systems.

Strabon also supports the querying of static geospatial data expressed in RDF using a subset of the recent OGC standard GeoSPARQL which consists of the core, geometry extension and geometry topology extension. The implementation of the other components of GeoSPARQL is underway.

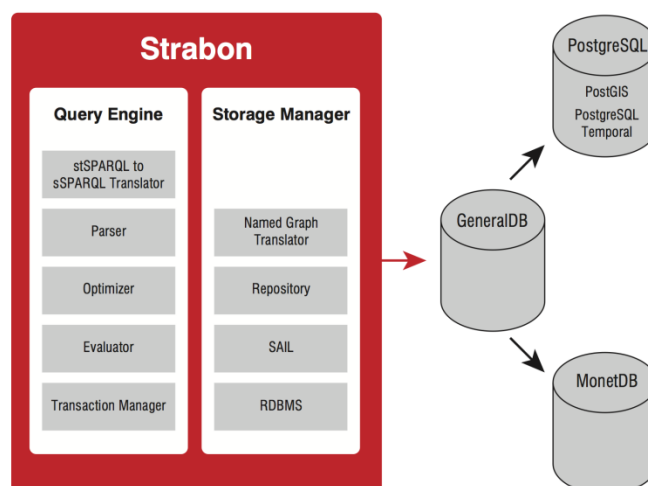


Figure 2.4 Strabon Architecture

<sup>8</sup> <http://rdf4j.org>

## 2.5 Sextant - Visualizing linked geospatial data

Sextant is a web-based and mobile ready application for exploring, interacting and visualizing time-evolving linked geospatial data. The platform is developed to be an application that is flexible, portable and interoperable with other GIS tools.

The core feature of Sextant is the ability to create thematic maps by combining geospatial and temporal information that exists in a number of heterogeneous data sources ranging from standard SPARQL endpoints, to SPARQL endpoints following the standard GeoSPARQL defined by the

Open Geospatial Consortium (OGC), or well-adopted geospatial file formats, like KML, GML and GeoTIFF<sup>9</sup>. These are the most promising file formats, and also provide tools for interaction with these layers, such as the colorization of geometry features according to specific values to create color maps for better understanding of the various aspects of layers. In that way, it overcomes the main disadvantage of the existing tools that allows the visualization of a single SPARQL endpoint, and provides functionality to domain experts from different fields in creating thematic maps, which emphasize spatial variation of one or a small number of geographic distributions. Moreover it goes beyond that and presents a map ontology that assists on modeling these maps in RDF and allows for easy sharing, editing and search mechanisms over existing maps.

Another important feature is the utilization of the temporal dimension. Implementation of the valid time component of stRDF and stSPARQL in system Strabon allows Sextant to query both the spatial and the temporal dimension. Enriching the results with temporal information allows the application to create layers with valid time. Using the SIMILE Timeline<sup>10</sup> widget it is able to make these layers appear and disappear from the map according to their valid time. This feature allows the creation of thematic maps that change over time and can assist experts in the fields of agriculture, biodiversity, climate, disasters, ecosystems, energy, water and weather in visualizing temporal maps that help them understand the evolution of data.

Apart from visualizing the spatial and temporal dimension, statistical charts play an important role in understanding the various measures of datasets. Statistical data is a foundation for policy prediction, planning and adjustments and underpins many of the mash-ups and visualizations we see on the web. There is strong interest in being able to publish statistical data in a web-friendly format to enable it to be linked and combined with related information. At the heart of a statistical dataset is a set of observed values organized along a group of dimensions, together with associated metadata. The Data Cube vocabulary<sup>11</sup> enables such information to be represented using the W3C RDF standard and published following the principles of linked data. Sextant also demonstrates how to utilize the Data Cube vocabulary to enhance existing datasets and allow the creation of charts through Sextant in an intuitive way that does not involve the use of SPARQL from the user point of view. The following figure shows the high-level architecture of Sextant web platform.

---

<sup>9</sup> <https://trac.osgeo.org/geotiff/>

<sup>10</sup> <http://www.simile-widgets.org/timeline/>

<sup>11</sup> <https://www.w3.org/TR/vocab-data-cube/>

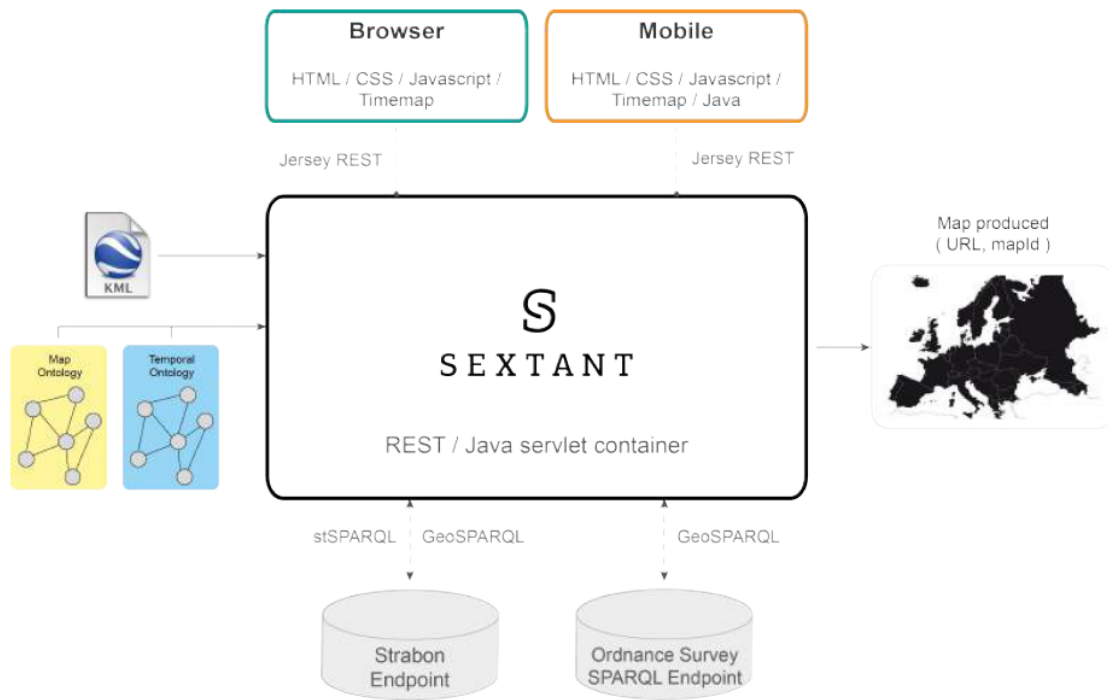


Figure 2.5 Sextant Architecture

### 3. MERGEO: ENGINEERING THE PLATFORM OF LINKED DATA TOOLS

In an information network that is constantly expanding, handling efficiently the available linked data and also trying to combine applications that process them, is very demanding. By putting into practice the Semantic Web's approach, merGeo is an attempt to share and reuse Linked Data across different applications. Although many tools and applications for exploiting this abundance of information have also started to emerge, most of them focus on a specific processing procedure. However, merGeo aims to combine three very essential tasks on Linked data; the transformation process from many data formats to RDF (GeoTriples), the storing and exploration of RDF datasets through SPARQL Endpoints (Strabon) and finally the visualization of these RDF data on a map (Sextant). All these features were merged together into one user-friendly, step-by-step guided and simple Web Application, offering the final user a unified platform for Linked (especially geospatial) Data manipulation. In this Chapter we are discussing the main modules constituting our API and also analyze the whole backend and frontend structure.

#### 3.1 MerGeo workflow in details

The core feature of merGeo is to provide the user, either he is an expert or not, a handy platform that implements three essential tasks on linked data; the transformation process of datasets into linked data, the storage and exploration of these data and finally the visualization of them on a map.

Until now, these task where accomplished by individual applications, which is justifiable if you are a domain expert or want to use each tool with specific and advanced options. However, if you want to perform the action triplet “transform-store-visualize” in a flash and utilize the key-features (and not only) of each tool, then the whole procedure of installing and configuring all the required applications is time-consuming and not so convenient. MerGeo is developed to satisfy this demand, through a handy application divided in three main sections and focused on both domain experts and daily users.

Firstly, the challenge was to provide a useful web interface for the command-line tool GeoTriples. Through user-friendly forms and helpful tips, the user is able to insert the options required for the transformation process. In that way, users who are not very familiar with command line tools are made capable of using a powerful transformation tool with great ease. At the beginning, user is given the choice of the input data format, which can be:

- a relational database
- a shape file<sup>1</sup>
- a XML file

Depending on user's choice, the appropriate form must be filled in order to generate the mapping file. After the mapping has been generated, the user is able to interact with it; the options to edit and download the file are been given immediately after the first

---

<sup>1</sup> <https://en.wikipedia.org/wiki/Shapefile>

transformation has been completed. If everything is correct and well-defined, he can proceed to the next step in order to generate the final RDF data. At this point, we have to mention that the final output format of the RDF data is also been given as an input option to the user, and can be:

- N-Triples
- Turtle<sup>2</sup>
- RDF/XML

Moreover, in some use-cases there is an additional option for RML processor usage instead of R2RML. Finally, and after the mapping has been generated into RDF data, the user can proceed to the next step which is to store these data into a SPARQL Endpoint.

Therefore, the need for Strabon functionality appears. So, we managed to embody Strabon services within a user-friendly interface, providing storage and data exploration through querying. The user is given the ability to work on a default, already running, SPARQL Endpoint or create a new one and store the desired data into that. In addition, the user can access two Endpoints at the same time, the default one which is created automatically after the application's initialization and also the last one created by the user himself. There is always an option to create a new, empty Endpoint but the previously created one will be removed for convenience.

So, the storing service is offered immediately after GeoTriples process has been finished but is also offered through a direct store interface, where the user can give the input either as a file URI or directly in a specific text input area. Immediately upon the data storage completion, the user is able to execute queries on the active SPARQL Endpoint and retrieve the results in the desired output format which can be:

- HTML
- SPARQL/XML
- GeoJSON

Last but not least is the significant process of data visualization. MerGeo combines the whole transformation and storing process with Sextant, giving the user the ability to visualize the stored RDF data that include WKT/GML geometries on a map. Through Strabon interface there is an option that enables the user to preview the already stored linked data on map, thanks to RESTful services based on Sextant. Additionally, Sextant's API can be used independently and full-featured through its exclusive section in MerGeo application.

The previously described procedure is visually represented through use cases and snapshots in Chapter 4.

---

<sup>2</sup> <https://www.w3.org/TR/turtle/>

## 3.2 Architecture

Focusing on merGeo architecture, one can see that we have chosen Spring MVC as a Web Framework and we make requests to GeoTriples, Strabon and Sextant as clients through RESTful APIs. So, every HTTP request passes from Spring, then depending on the request and if needed we interact with the proper API of one of the tools to retrieve the appropriate data and turn back to Spring to form the HTTP response for the user. Now, let's look the whole application's architecture in more detail.

### 3.2.1 Spring MVC Framework

Starting to develop merGeo, we had to decide whether the use of an application framework would be advantageous. After doing some research, we finally settled for Spring MVC Framework<sup>3</sup>. MVC stands for Model-View-Controller, which is one of the main features that make Spring prominent amongst web applications.

The Spring web MVC framework provides Model-View-Controller architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

The Model encapsulates the application data and generally they consist of POJO (Plain Old Java Objects). The View is responsible for rendering the model data and generating HTML output that the client's browser can interpret using files such as JSPs (in our case). Finally the Controller is responsible for processing user requests and building the appropriate model which passes to the view for rendering.

Spring is, like most of the Web MVC Frameworks, request-driven and is designed around a central Servlet that dispatches requests to controllers and offers other functionality that facilitates the development of web applications. However, Spring's `DispatcherServlet` offers more than that; it is responsible for handling all the HTTP requests and responses on the application. Below, on Figure 3.1, we can see the request processing work-flow on the Spring Web MVC `DispatcherServlet`.

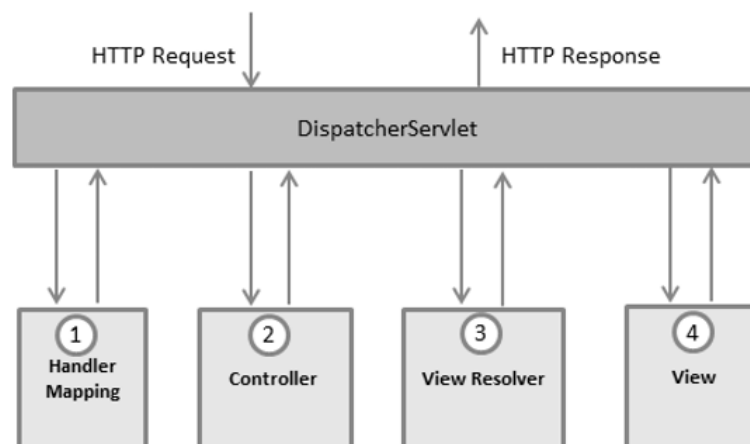


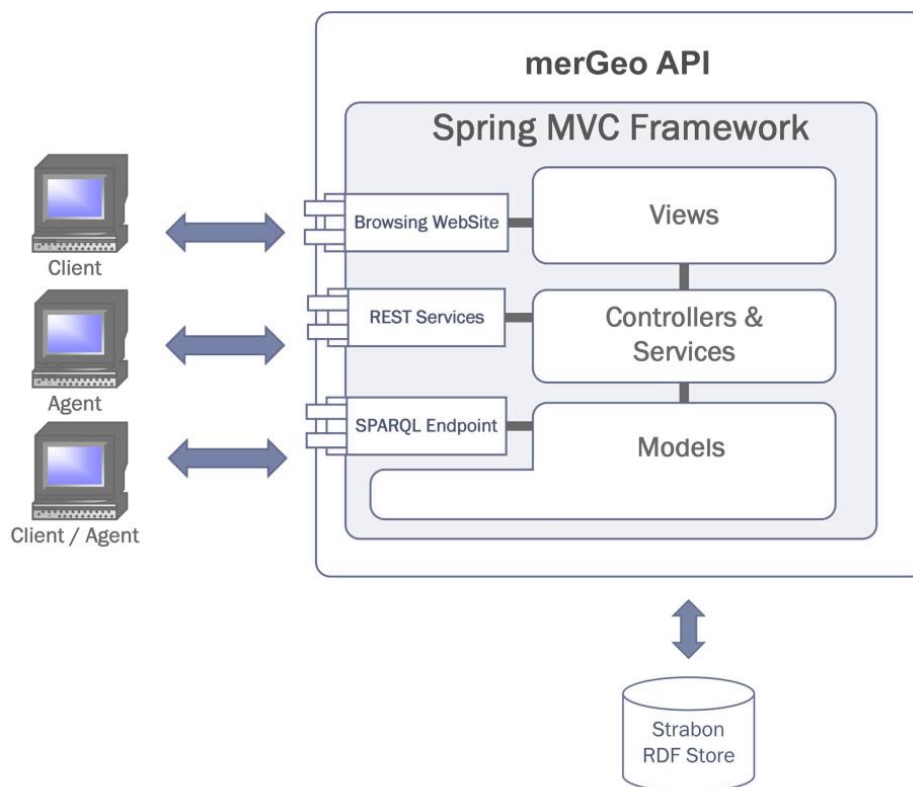
Figure 3.1 Work-flow of the Spring Web MVC architecture

<sup>3</sup> <https://spring.io>

Following is the sequence of events corresponding to an incoming HTTP request to *DispatcherServlet*:

1. After receiving an HTTP request, *DispatcherServlet* consults the *HandlerMapping* to call the appropriate Controller.
2. The Controller takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the *DispatcherServlet*.
3. The *DispatcherServlet* will take help from *ViewResolver* to pick up the defined view for the request.
4. Once view is finalized, The *DispatcherServlet* passes the model data to the view which is finally rendered on the browser.

Focusing on merGeo, separate controllers and models have been implemented to receive and serve many different requests, depending on the corresponding tool that invokes the request. At the same time Spring enables custom JSP tags<sup>4</sup> and it is meant for front-end frameworks like Bootstrap<sup>5</sup> which is the one we are using. However, to fully understand how Spring supports flexibility and clean roles we can take a look at the following figure.



**Figure 3.2 Spring MVC Framework architecture within merGeo**

<sup>4</sup> <https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/spring-form-tld.html>

<sup>5</sup> <http://getbootstrap.com/>



### 3.2.2 Server analysis

The server is built in Java 8<sup>6</sup> and follows the MVC design pattern. Mainly it is composed of Models handling the data from every input form, exclusive Controllers on every tool section handling GET/POST requests and finally Views that are responsible for the front-end content. Below we are going to analyze each one of these three sections in details.

Geotriples component is supported through an exclusive controller class (*GeotriplesController*), two model classes connected with the input forms (*MapInputModel*, *RdfInputModel*) and finally a service class (*GeotriplesService*) that parses the data and interacts with GeoTriples API.

*GeotriplesController* is responsible for handling all the requests made under the GeoTriples section. The controller class is divided into different methods; each of them handles the appropriate request and calls the proper service method, implemented in *GeotriplesService* class. The input data parsing is performed in services by analyzing the form data (*MapInputModel* for step 1/3 and *RdfInputModel* for step 2/3) constructing the final arguments-string to be given as input to Geotriples API. The communication between merGeo and GeoTriples is achieved inside the service methods. Output results from the transformation carried out by GeoTriples are returned back to the proper controller and finally presented to the end-user through the pertinent View.

Moving forward to Strabon, its full service is managed by one main controller (*EndpointController*) with similar functionality as GeoTriples controller. In order to initialize a new Endpoint, a creation form must be filled. This form is modeled through an exclusive Model class (*EndpointModel*) that shares the data between client and server components. However, a connection with an already running POSTGRESQL database must be established; action that is achieved through an individual controller (*PostgreSQLJDBC*). In order to validate the given form data with the background running database, a Validator class has been developed (*EndpointValidator*) that checks the availability of both the given username and database name.

After the Endpoint initialization, the whole communication between merGeo and Strabon is accomplished through a RESTful API (*GeneralSPARQLEndpoint*). Every query and store request is fulfilled through that API and the results are returned to the controller and finally to the end-user via the corresponding View.

Finally, the Sextant support comes through a RESTful API integrated with Strabon interface. Their integration becomes perceptible on the query results View of Strabon, where the user is able to portray the results on a map with just a click of a button. This service is supported through JavaScript functions in Sextant API that take as arguments the appropriate values about the running Endpoint, the requested query etc. Additionally, through its exclusive section on merGeo, the user is capable of fully using Sextant application in which he is able to visualize, explore and interact with the requested data. MerGeo is built using a client-server architecture model as depicted on the following figure (Figure 3.3).

---

<sup>6</sup> <https://docs.oracle.com/javase/8/docs/>

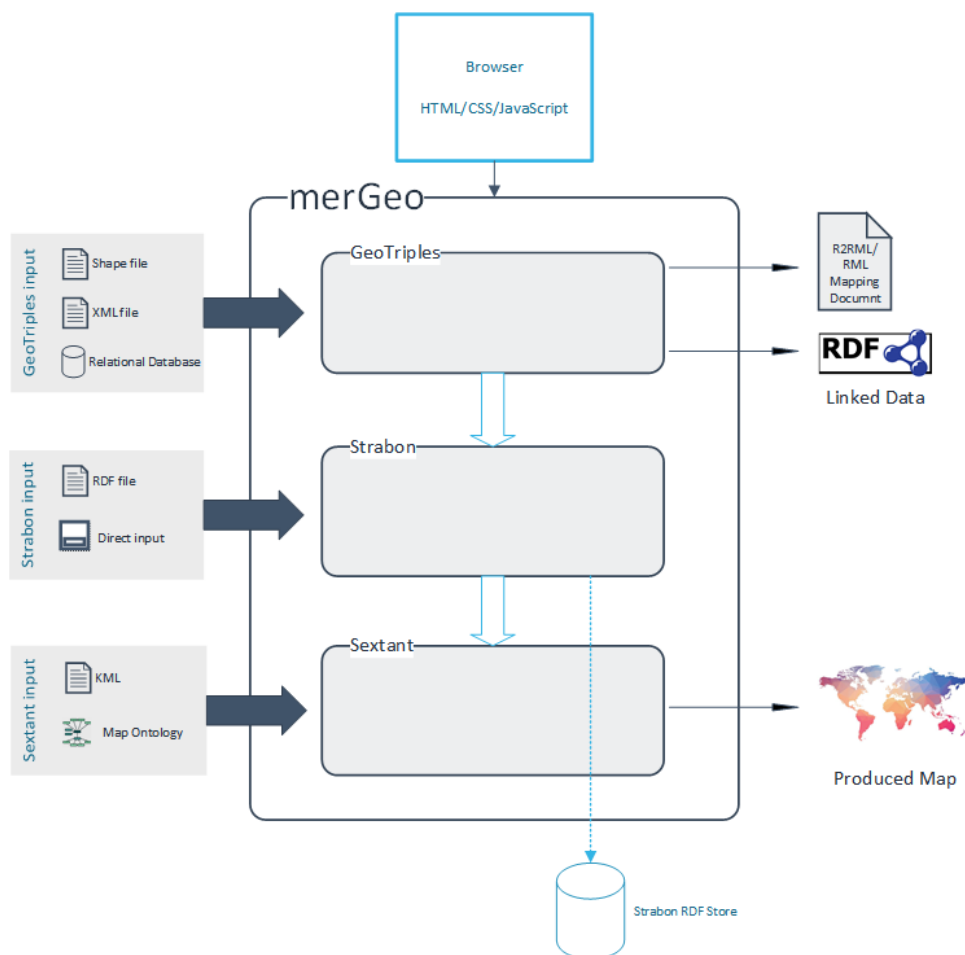


Figure 3.3 merGeo architecture

### 3.2.3 Client analysis

Taking into consideration the need for a convenient web application which encourages the user to perform the action triplet “transform-store-visualize” without frustration at all, we developed a responsive, full-guided though straightforward web interface.

We used Bootstrap framework to implement a responsive single code base user interface (UI), that is used across different platforms and screen displays. Besides, a lot of effort was put in designing user-friendly and flexible environment for the end-user, which is developed using HTML 5<sup>7</sup>, CSS<sup>8</sup> and JavaScript<sup>9</sup> technologies.

In Figure 3.4 we show the main page of merGeo, from which the user can either start with a chain process from transformation to visualization of linked data (from GeoTriples to Sextant) or he can use any of the available tools individually. The three centered navigation-bar tabs refer to each one of the modules.

<sup>7</sup> <http://www.w3.org/TR/html5/>

<sup>8</sup> <http://www.w3.org/Style/CSS/>

<sup>9</sup> <https://www.javascript.com/>

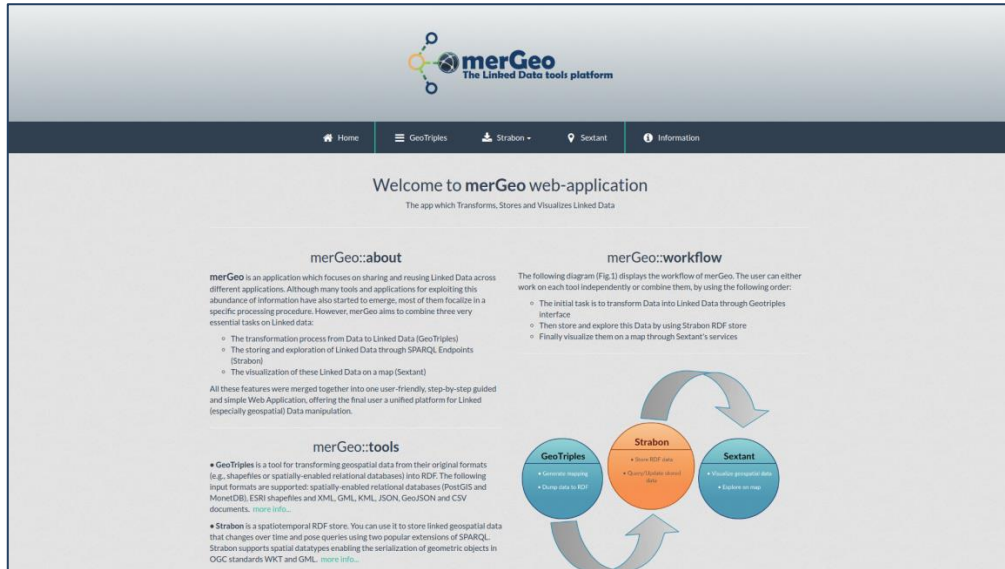


Figure 3.4 merGeo - Home page

From the first tab, users can begin with the transformation process (GeoTriples) by filling the displayed forms and generating the mapping/RDF files of the given dataset.

Following the second tab from the main page (Strabon), a dropdown menu appears, giving the user all the available options on Endpoints. He is able to create a new one; work on the latest created one or even work on the default endpoint.

On the third tab (Sextant) the user is given the ability to have full access on Sextant application and visualize his data. Below we present the initial View of each tool. Still, on the following Chapter we are going to demonstrate the whole client interface through use cases and screenshots.

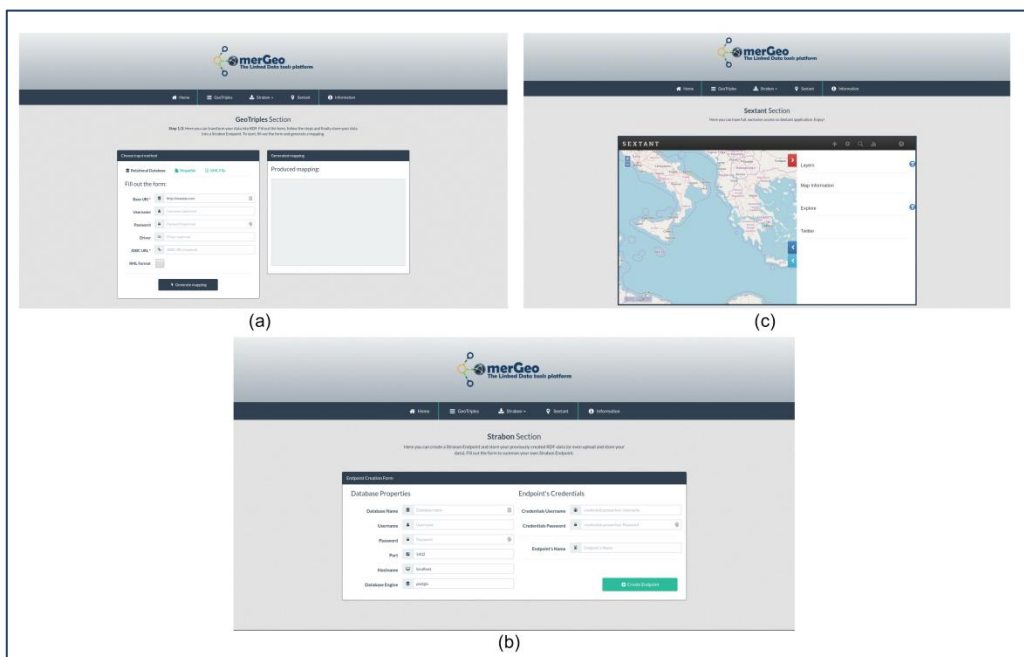


Figure 3.5 merGeo - Management tools pages

## 4. DEMONSTRATION OF MERGEO WEB PLATFORM THROUGH USE CASES

In the following chapter we will represent a detailed use case in order to give a more specific insight on how the application works and also to point out the convenience merGeo offers through its user-friendly interface. The selected use case will be described in detailed steps and will be followed by screenshots of the application throughout the whole procedure.

### 4.1 Use case in a detailed description

Suppose we have the following use case: the user wants to create a new Endpoint through Strabon, next needs to transform a dataset from a shape file to linked RDF dataset with GeoTriples tool and store it on the previously created Endpoint and finally wants to query and visualize the results. So, the following steps should be performed:

1. Create a new Endpoint.
2. Generate mapping from input shape file.
3. Generate RDF data.
4. Store the RDF data on the previously created Endpoint.
5. Query the previously created Endpoint and explore the results.
6. Visualize the results.

So, to begin with, the user has to click on Strabon section and from the displayed dropdown menu has to select the “Create new Endpoint” option, as displayed on Figure 4.1 below:

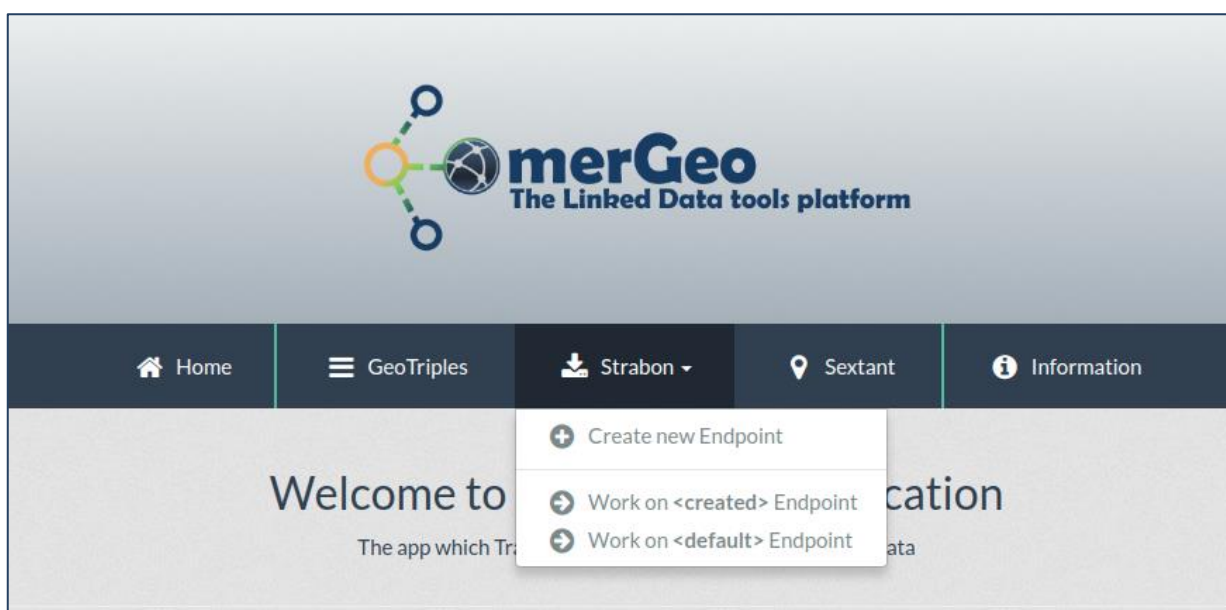


Figure 4.1 merGeo - Strabon dropdown menu

Next, the displayed form must be filled (Figure 4.2).

**Figure 4.2 merGeo - Strabon Endpoint creation form**

There are placeholders to all the text inputs, giving tips about the requested input. At this point we should mention that an input validator runs in the background of the form, checking for the availability of the following input fields: Database Name, Username, Endpoint's Name. If any of the given input already exists then the requested "Create Endpoint" action will not be completed, and the form will be reloaded with a warning notification and with error marks on every failed input field.

If the form is admissible, the following View is loaded:

**Figure 4.3 merGeo - Endpoint main View**

This is the main View of an active Endpoint. On the left side there are several functional buttons. The first one opens the “Store” interface for the current Endpoint (see Chapter 4.2), the second one clears the query text area on the right and the following buttons execute the displayed queries on the active Endpoint.

Let us now leave from Strabon section and move to GeoTriples section by clicking on the corresponding button on the navigation bar. When we do so, the initial GeoTriples View is displayed, with an input form on the left and a text area output on the right. On the left form, we choose as input method the Shape file and give the file input by clicking on the “Choose File” button, selecting the appropriate file from our computer.

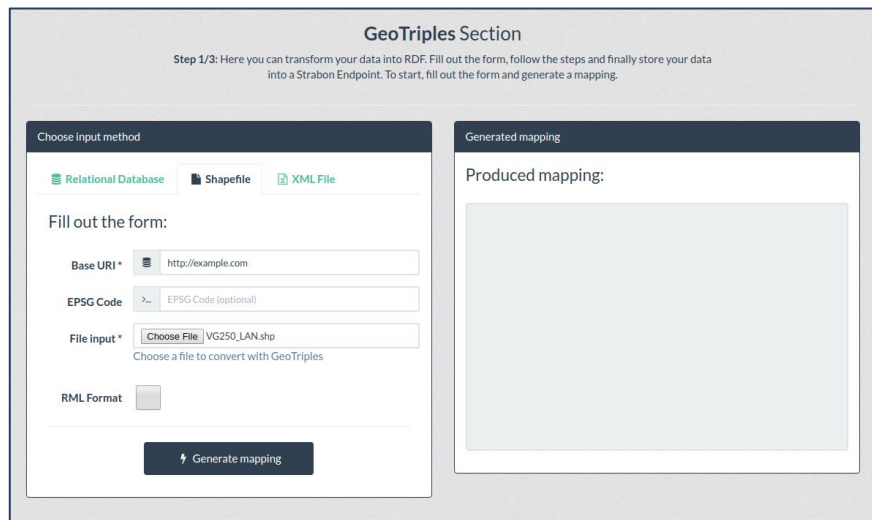


Figure 4.4 merGeo - Geotriples 1/3 step

When we are ready, we click on the primary “Generate mapping” button and a new View is displayed, with a new form on the left and a filled text area on the right that contains the mapping output.

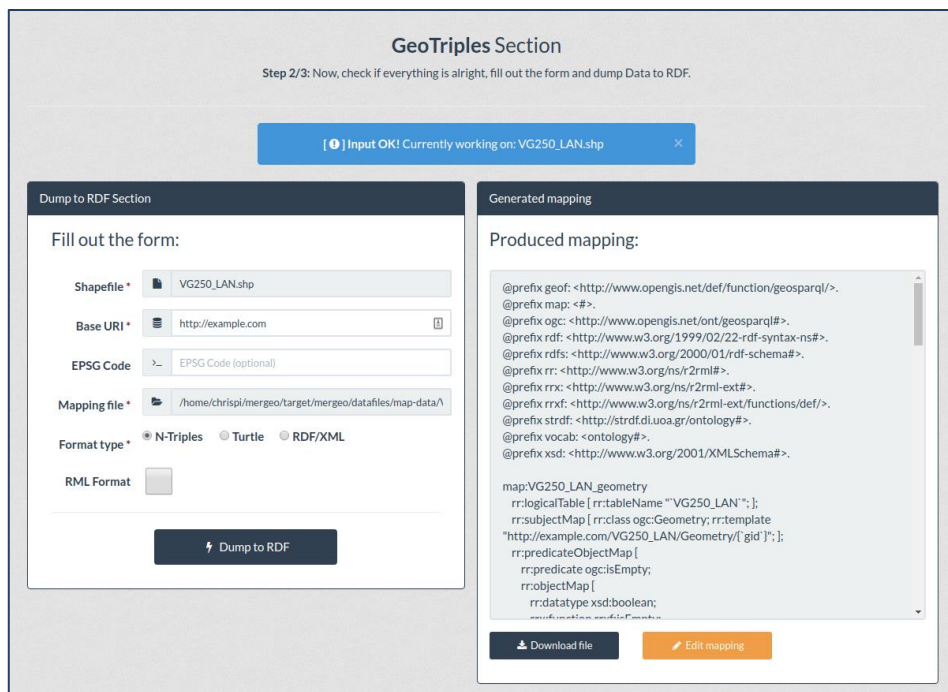


Figure 4.5 merGeo - GeoTriples 2/3 step

On the right, the user is able to preview the produced mapping. He is also been given the ability to download the mapping file or even edit and save it by clicking on the corresponding buttons. Let us assume that everything seems correct and the user wants to move on to generate the final linked RDF data. On the left side, he can select the options he desires and then click on the primary “Dump to RDF” button. That’s it! The final linked dataset is ready and the following View opens to the browser:

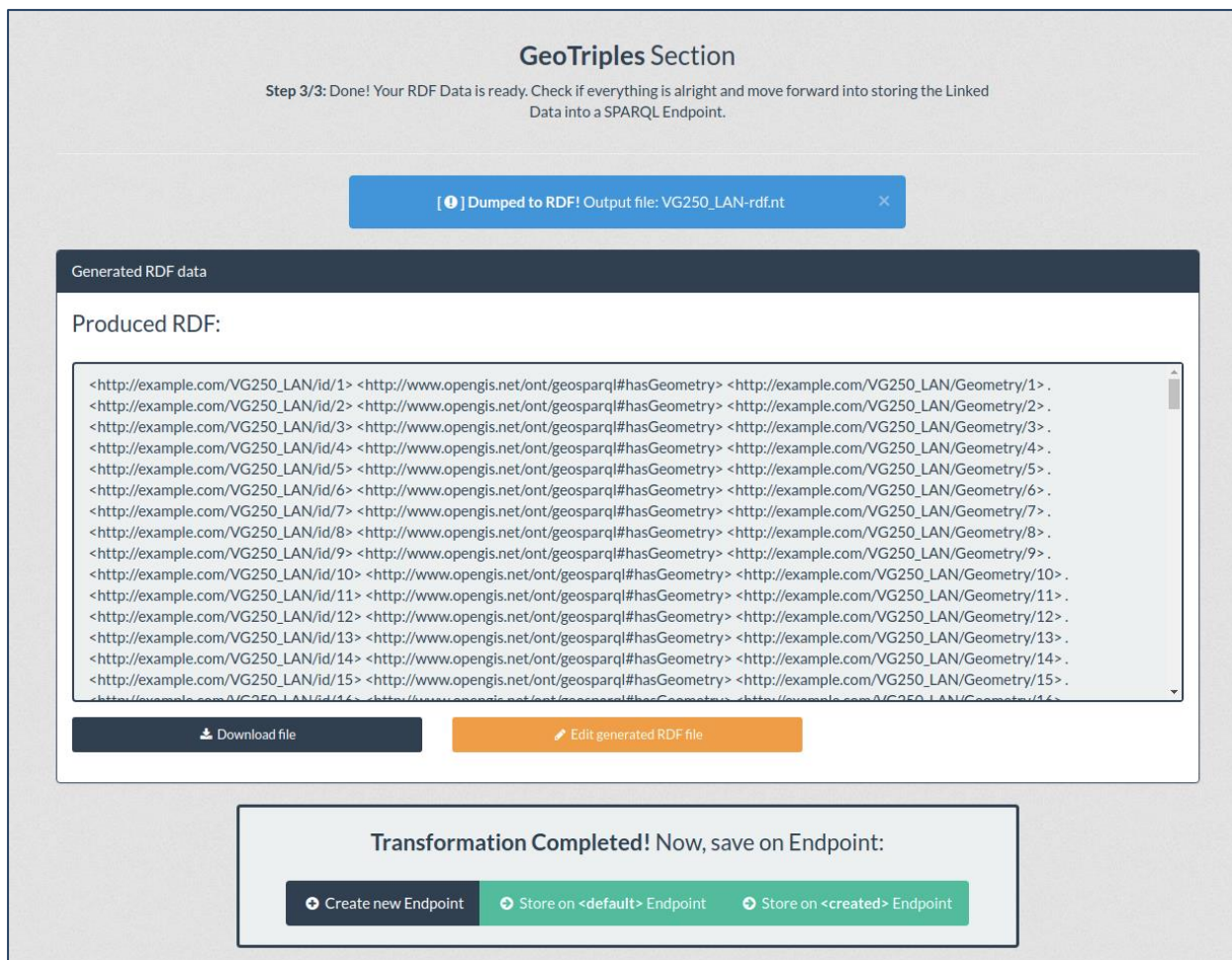
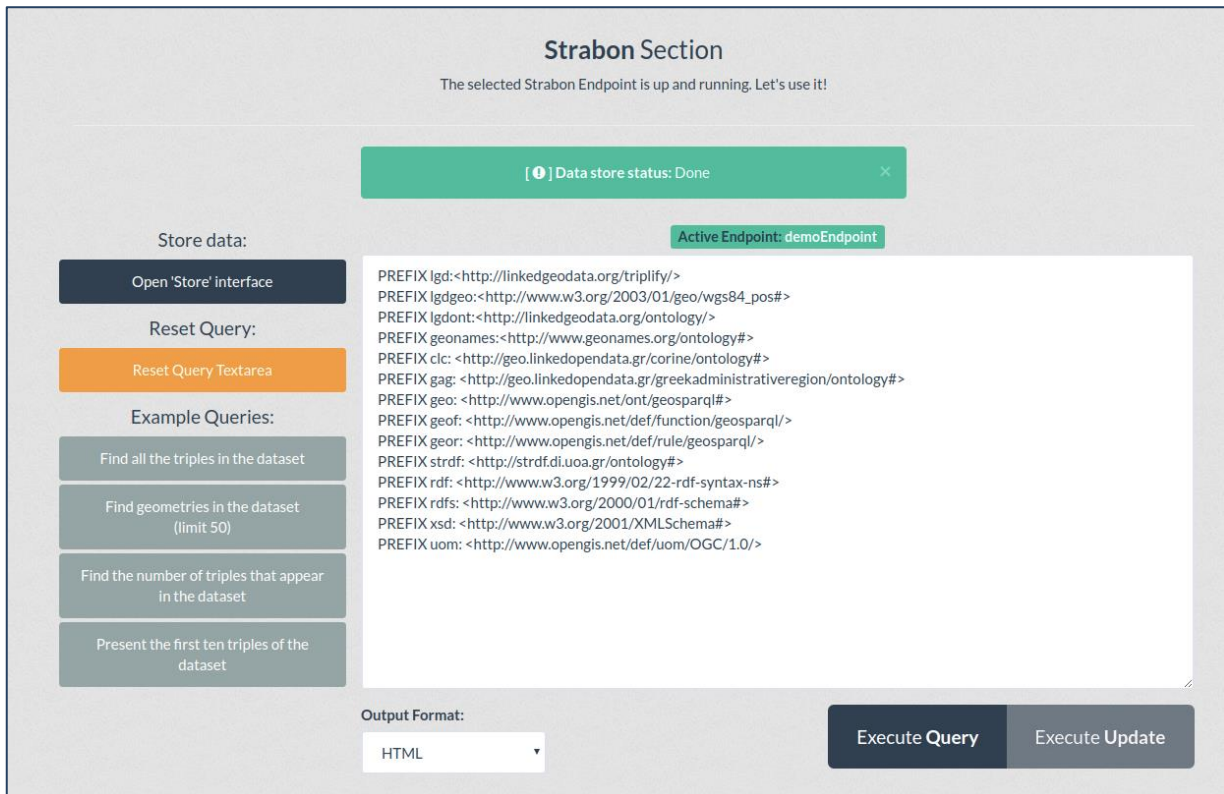


Figure 4.6 merGeo - GeoTriples 3/3 step

Again, options for download and edit are available. But the important feature lies beyond the RDF output. The user is given the ability to directly store the generated data on an Endpoint. As one can see, the given options are the following:

1. Create a new Endpoint and save the data into that
2. Store the data into the default Endpoint
3. Store the data into the created Endpoint, which in our situation is the Endpoint that the user created at the beginning of this use case

The user wants to save them on the “demoEndpoint” he created previously, so the third option is selected and the following View opens:



**Figure 4.7 merGeo - Strabon main after Store**

A notification appears in the middle, informing the user about the store status (in our case the given data were successfully stored). Now, the user can query the active Endpoint through the query text area we showed before or use any of the pre-defined query examples on the left. The query language used is SPARQL and as the above figure shows, we have already injected the query text area with some commonly used prefixes.

Furthermore, below the text area input there is an option for the user to choose the output format of the query results. For the time being, only three output formats are supported:

- HTML
- SPARQL/XML
- GeoJSON

Also, the predefined example queries are displayed by default in HTML format. In case the user wants to view their results in other formats, the query must be executed again by clicking the “Execute Query” button on the right, after the desired format has been selected.

Suppose the user clicks the first example query button named “Find all the triples in the dataset”. The query is executed and the results appear as follows, divided in triplets per row and in “subject” “predicate” “object” per column:




Endpoint Results:			
s	p	o	
<a href="http://example.com/VG250_LAN/id/1">http://example.com/VG250_LAN/id/1</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/1">http://example.com/VG250_LAN/Geometry/1</a>	
<a href="http://example.com/VG250_LAN/id/2">http://example.com/VG250_LAN/id/2</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/2">http://example.com/VG250_LAN/Geometry/2</a>	
<a href="http://example.com/VG250_LAN/id/3">http://example.com/VG250_LAN/id/3</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/3">http://example.com/VG250_LAN/Geometry/3</a>	
<a href="http://example.com/VG250_LAN/id/4">http://example.com/VG250_LAN/id/4</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/4">http://example.com/VG250_LAN/Geometry/4</a>	
<a href="http://example.com/VG250_LAN/id/5">http://example.com/VG250_LAN/id/5</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/5">http://example.com/VG250_LAN/Geometry/5</a>	
<a href="http://example.com/VG250_LAN/id/6">http://example.com/VG250_LAN/id/6</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/6">http://example.com/VG250_LAN/Geometry/6</a>	
<a href="http://example.com/VG250_LAN/id/7">http://example.com/VG250_LAN/id/7</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/7">http://example.com/VG250_LAN/Geometry/7</a>	
<a href="http://example.com/VG250_LAN/id/8">http://example.com/VG250_LAN/id/8</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/8">http://example.com/VG250_LAN/Geometry/8</a>	
<a href="http://example.com/VG250_LAN/id/9">http://example.com/VG250_LAN/id/9</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/9">http://example.com/VG250_LAN/Geometry/9</a>	
<a href="http://example.com/VG250_LAN/id/10">http://example.com/VG250_LAN/id/10</a>	<a href="http://www.opengis.net/ont/geosparql#hasGeometry">http://www.opengis.net/ont/geosparql#hasGeometry</a>	<a href="http://example.com/VG250_LAN/Geometry/10">http://example.com/VG250_LAN/Geometry/10</a>	

Figure 4.8 merGeo - Strabon query results (HTML)

In HTML format, every one of the results is clickable (a tags). So, in case the user wants to execute a query with one specific subject, predicate or object, the only thing he has to do is to click on the desired item and the query results page refreshes with the new triplets. Also, notice that the executed query is always displayed on the text area above.

Another supported output format is XML. Assuming that the user has selected XML as the output format, the query results area will display the following content:

Endpoint Results:		
<pre>&lt;?xml version='1.0' encoding='UTF-8'?&gt; &lt;sparql xmlns='http://www.w3.org/2005/sparql-results#'&gt;   &lt;head&gt;     &lt;variable name='s'/&gt;     &lt;variable name='p'/&gt;     &lt;variable name='o'/&gt;   &lt;/head&gt;   &lt;results&gt;     &lt;result&gt;       &lt;binding name='s'&gt;         &lt;uri&gt;http://example.com/VG250_LAN/id/1&lt;/uri&gt;       &lt;/binding&gt;       &lt;binding name='p'&gt;         &lt;uri&gt;http://www.opengis.net/ont/geosparql#hasGeometry&lt;/uri&gt;       &lt;/binding&gt;       &lt;binding name='o'&gt;         &lt;uri&gt;http://example.com/VG250_LAN/Geometry/1&lt;/uri&gt;       &lt;/binding&gt;     &lt;/result&gt;   &lt;/results&gt; &lt;/sparql&gt;</pre>		

Figure 4.9 merGeo - Strabon query results (XML)

Now, assuming that the result set include WKT/GML geometries, the user is able to click the button named “Show results on map”. However, in case the result set does not include any WKT/GML geometries, the user will still be able to click the button but the Sextant instance that shows up will show no layers.

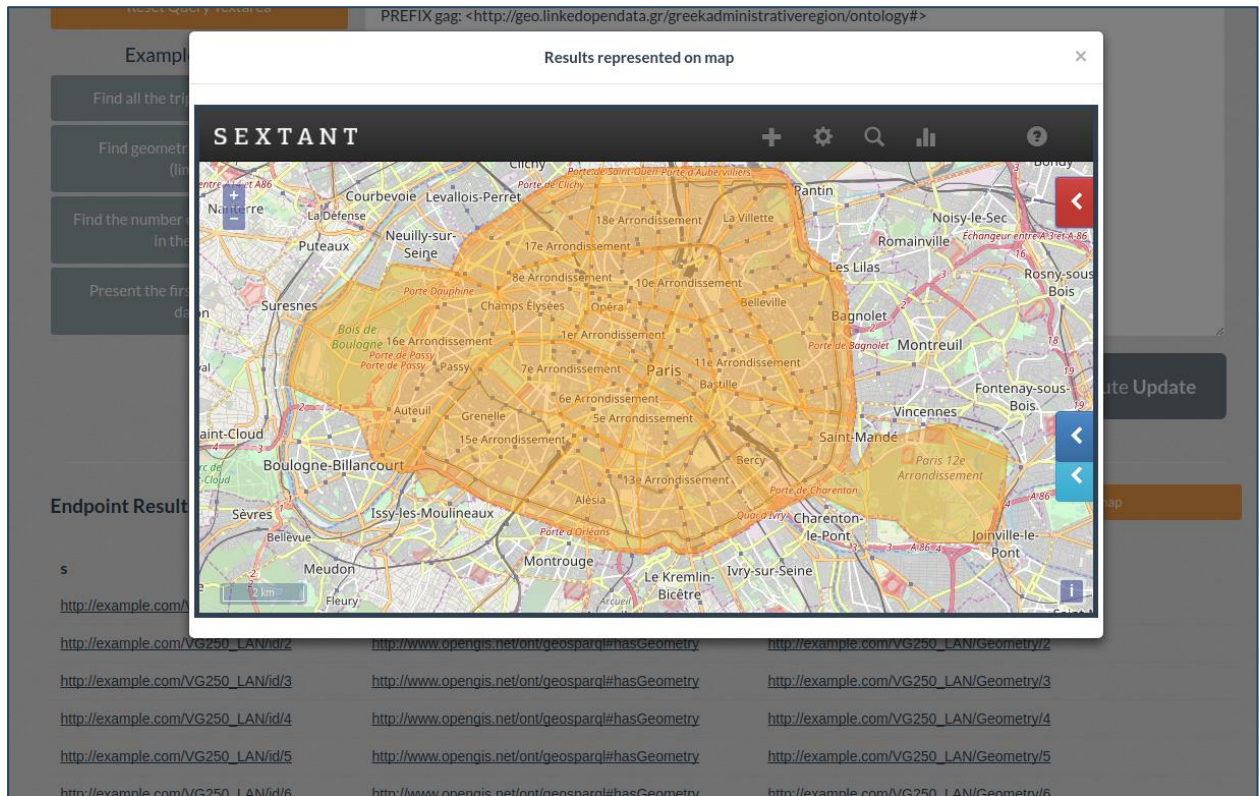


Figure 4.10 merGeo - Strabon results displayed on map

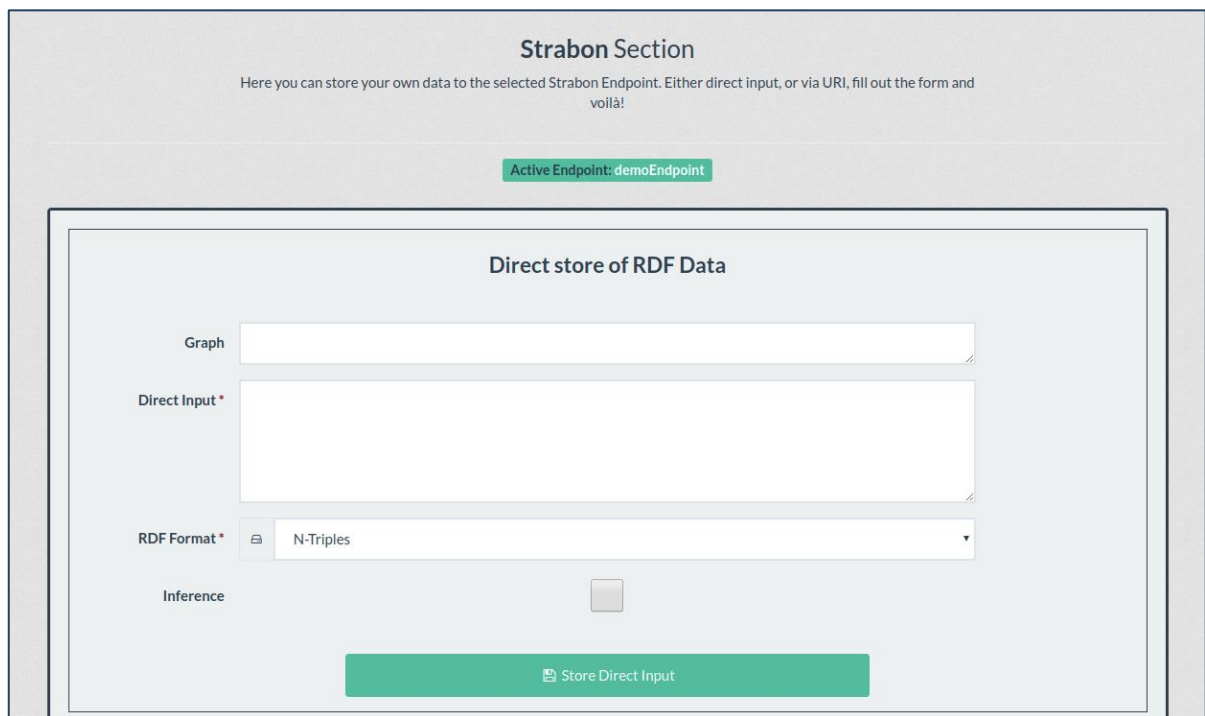
The modal's content is a Sextant instance with a created layer depending on the results of the query. The user is able to explore and interact with the visualized data through the above interface.

To summarize, this was one extended use case of merGeo web platform, where most of its features were demonstrated briefly and giving an idea of how it works. For some more available features, see the following chapter.

## 4.2 Additional features

MerGeo web platform offers some additional features about linked data manipulation that were not mentioned in the previous use case. These features accomplish to make each one of the management tools to be full independent so the user would be able to use them individually.

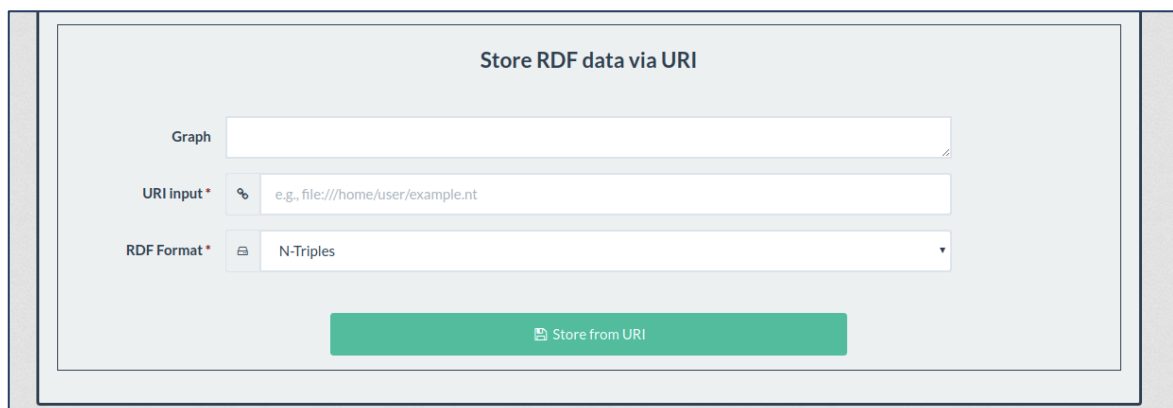
The first one is the option to directly store data into a Strabon Endpoint, without the need for GeoTriples to mediate. When the main View of the requested Endpoint shows up, as we already mentioned, it includes a “Store” button on the top left. By clicking on that button, a new View is displayed, giving the user the capability to store data on the active Endpoint directly or via a file URI. These two options are featured as follows:



The screenshot shows a web interface titled "Strabon Section". Below the title is a subtitle: "Here you can store your own data to the selected Strabon Endpoint. Either direct input, or via URI, fill out the form and voilà!". Below this is a green button labeled "Active Endpoint: demoEndpoint". The main content area is titled "Direct store of RDF Data" and contains the following fields:

- "Graph": A text input field.
- "Direct Input \*": A large text area for direct input.
- "RDF Format \*": A dropdown menu currently showing "N-Triples".
- "Inference": A checkbox.
- A green button at the bottom labeled "Store Direct Input".

Figure 4.11 merGeo - Strabon direct store



The screenshot shows a web interface titled "Store RDF data via URI". It contains the following fields:

- "Graph": A text input field.
- "URI input \*": A text input field with a URL icon and the example text "e.g., file:///home/user/example.nt".
- "RDF Format \*": A dropdown menu currently showing "N-Triples".
- A green button at the bottom labeled "Store from URI".

Figure 4.12 merGeo - Strabon store via file URI

The second one is the ability to use Sextant tool directly, independently and full-featured through merGeo. Our platform achieves to offer this functionality through the “Sextant” tab from the navigation bar. After clicked, a new View is displayed and the user is able to use Sextant in a flash and of course, fully functional.

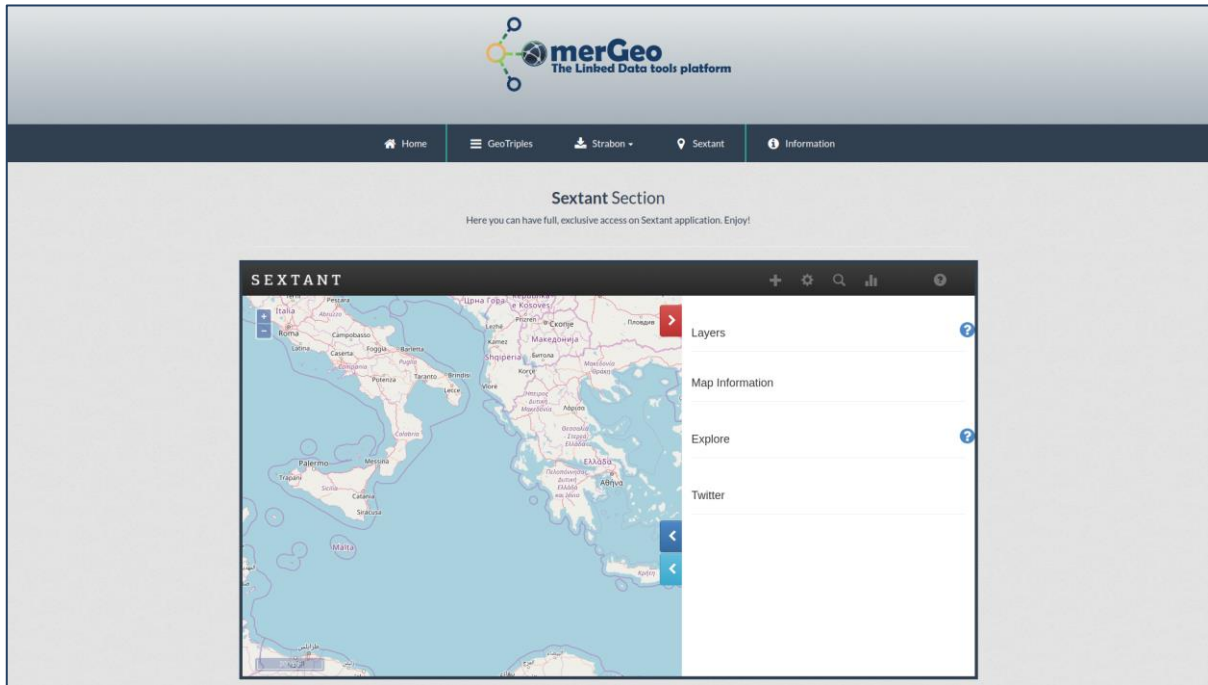


Figure 4.13 merGeo - Sextant UI

## 5. CONCLUSIONS AND FUTURE WORK

In this thesis we presented merGeo, an application that integrates the functionality of three very useful tools to manipulate linked data. Through its user-friendly interface that accomplishes to assist non-expert users to take advantage of semantic web technologies, we managed to provide the action triplet “transform-store-visualize” in a convenient and expeditious way.

We developed three components, one for each tool; GeoTriples, Strabon and Sextant. In the first one, the user is able to experience GeoTriples services without having to use a command line, trying to find out the way it executes and the arguments it takes. Through a handy, full-guided interface the user is capable of transforming data to linked data in RDF format within a few clicks and some input fields.

Moving on to Strabon, most of its functions are supported through a new simple and practical web interface. The transaction from GeoTriples to Strabon comes immediately after the data transformation is completed, giving the user the option to directly save the generated linked data on an Endpoint. Moreover, the user is able to store additional data through Strabon store interface. Of course, user is given the ability to work with Strabon to query and explore the stored data and finally visualize them on a map (provided that they include WKT/GML geometries) through Sextant API.

Finally, as one can see from merGeo application, Sextant services are fully supported. The user can either visualize the already stored data through the proper option on Strabon interface or he can experience the complete Sextant API straight from its exclusive section on merGeo.

To summarize, merGeo achieves to provide an easily operated application that allows both to domain experts and non-experts at all to make full use of semantic web tools and technologies and also convinces them to adopt these technologies by presenting the benefits of using together different applications that are based on linked data management.

In the future we would like to make merGeo more robust and versatile by improving existing aspects and adding more functionality.

Our next objective is to bring temporal support into merGeo and especially into both Strabon and Sextant components as they already support it through their native applications. By supporting spatiotemporal linked data, merGeo application fields are enhanced and the whole web platform becomes more functional. Also, we plan to add extra formats to be supported as results of the queries.

Furthermore, we would like to support full management over running Endpoints, where the user would be able to run multiple Endpoints at the same time and also would be able to delete them anytime. Additionally, as all the three tools are still supported and constantly updated, a mechanism to fully support the latest versions should be necessarily added to merGeo in order to remain an actually productive tool.

All these new features would turn merGeo into an even more powerful tool for leveraging the linked spatiotemporal data cloud.

## ANNEX A INSTALLATION INSTRUCTIONS

MerGeo platform is compressed in *mergeo-1.0.0-BUILD-SNAPSHOT.war* file. The *.war* file in folder */webapps* needs to be deployed in a Web Application Server (Apache Tomcat) and also collaborates/communicates with external web applications like Sextant and Strabon, who are both server sided but are auto-deployed through merGeo. We will give instructions of how-to complete installation. After completing installation, you can browse merGeo platform at <http://localhost:8080>.

Due to some backend services, merGeo can only be supported on Linux systems. Follow the next steps and install the required applications/modules in order to successfully deploy merGeo on Tomcat.

### 1. Apache Tomcat (<http://tomcat.apache.org/>)

Apache Tomcat can be downloaded here: (<https://tomcat.apache.org/download-90.cgi>), depends on the operating system and its distribution. You can find installation instructions here (<http://tomcat.apache.org/tomcat-9.0-doc/setup.html>). After installation, you can point your browser at this location (<http://localhost:8080/>) to verify that the deployment succeeded. You can find information of how-to deploy an application here: (<http://tomcat.apache.org/tomcat-9.0-doc/deployer-howto.html>)

Important! In order to create folders and edit files inside Tomcat location, we have to change the permissions to Tomcat folder and to all its sub-folders. This can be done through the graphical interface of our operating system, on “tomcat” folder: Right-Click->Permissions and change everything to “Create and delete files”. Otherwise, through command line you can run:

```
$> sudo chmod -R 777 /tomcat
```

(We know it’s not quite safe to change the permissions in 777, this is going to be fixed in the next updates)

### 2. Installing PostgreSQL and PostGIS

1. Install PostgreSQL 9.0 or higher. More information can be found at <http://www.postgresql.org/download/>

```
$> sudo apt-get install postgresql-9.5
```

2. Install PostGIS 2.2 or higher. More information can be found at <http://postgis.refractory.net/download/>

```
$> sudo apt-get install postgresql-9.5-postgis-2.2
```

3. Provide a password for default user (postgres)

```
$> sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD 'postgres';"
```

## Creating a spatially enabled database

Spatially-enabled databases permit the use of spatial function calls. MonetDB creates spatially-enabled databases by default if you have enabled the geom module. More information on how to create a spatially-enabled database in PostGIS can be found at <http://postgis.refractory.net/docs/>.

1. Set postgis-2.2 path.

```
$> POSTGIS_SQL_PATH=`pg_config --sharedir`/contrib/postgis-2.2
```

2. Create the spatial database that will be used as a template.

```
$> createdb -E UTF8 -T template0 template_postgis -U postgres
```

3. Add PLPGSQL language support.

```
$> createlang -d template_postgis plpgsql -U postgres
```

## Load the PostGIS SQL routines

```
$> psql -d template_postgis -f $POSTGIS_SQL_PATH/postgis.sql
```

```
$> psql -d template_postgis -f $POSTGIS_SQL_PATH/spatial_ref_sys.sql
```

## Allow users to alter spatial tables

```
$> psql -d template_postgis -c "GRANT ALL ON geometry_columns TO PUBLIC;"
```

```
$> psql -d template_postgis -c "GRANT ALL ON geography_columns TO PUBLIC;"
```

```
$> psql -d template_postgis -c "GRANT ALL ON spatial_ref_sys TO PUBLIC;"
```

## Perform garbage collection

```
$> psql -d template_postgis -c "VACUUM FULL;"
```

```
$> psql -d template_postgis -c "VACUUM FREEZE;"
```

## Allows non-superusers the ability to create from this template

```
$> psql -d postgres -c "UPDATE pg_database SET datistemplate='true' WHERE datname='template_postgis';"
```

```
$> psql -d postgres -c "UPDATE pg_database SET datallowconn='false' WHERE datname='template_postgis';"
```

## Create a spatially-enabled database named endpoint

```
$> createdb endpoint -T template_postgis
```

<!> Notice: In some psql commands you may need to add: -U postgres <!>

## REFERENCES

- [1] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1--22, 2009.
- [2] OGC. GeoSPARQL - A geographic query language for RDF data, November 2010.
- [3] Manolis Koubarakis and Kostis Kyzirakos. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. In *ESWC*, 2010.
- [4] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: a semantic geospatial dbms. In *The Semantic Web--ISWC 2012*, pages 295--311. Springer, 2012.
- [5] Konstantina Bereta, Charalampos Nikolaou, Manos Karpathiotakis, Kostis Kyzirakos, and Manolis Koubarakis. SexTant: Visualizing Time-Evolving Linked Geospatial Data. In *International Semantic Web Conference (Posters & Demos)*, 2013.
- [6] Open Geospatial Consortium. OGC KML. OGC® Implementation Standard, 2008. Available from: [http://portal.opengeospatial.org/files/?artifact\\_id=27810](http://portal.opengeospatial.org/files/?artifact_id=27810).
- [7] Charalampos Nikolaou, Kallirroï Dogani, Kostis Kyzirakos, and Manolis Koubarakis. Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data. In *ESWC (Satellite Events)*, pages 209--213, 2013.
- [8] Manolis Koubarakis. Linked Open Earth Observation Data: The LEO Project, 2014.
- [9] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: a semantic geospatial dbms. In *The Semantic Web--ISWC 2012*, pages 295--311. Springer, 2012.
- [10] Open Geospatial Consortium. Geography Markup Language (GML) Encoding Standard. OpenGIS Implementation Standard, 2007. Available from: [http://portal.opengeospatial.org/files/?artifact\\_id=20509](http://portal.opengeospatial.org/files/?artifact_id=20509).
- [11] Open Geospatial Consortium. OpenGIS Implementation Standard for Geographic information-Simple feature access - Part 1: Common Architecture. OpenGIS Implementation Standard, 2010. Available from: [http://portal.opengeospatial.org/files/?artifact\\_id=25355](http://portal.opengeospatial.org/files/?artifact_id=25355).
- [12] Kostis Kyzirakos, Ioannis Vlachopoulos, Dimitrianos Savva, Stefan Manegold, and Manolis Koubarakis. GeoTriples: a Tool for Publishing Geospatial Data as RDF Graphs Using R2RML Mappings. Technical report, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, 2014. <http://linkedeodata.eu/publications/iswc-2014-terracognita.pdf>.