NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

POSTGRADUATE STUDIES

MASTER THESIS

A web-based GML to stRDF / GeoSPARQL conversion tool

Marios S. Bekatoros

Supervisor: **Manolis Koubarakis**, Professor UoA

ATHENS

February 2015

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΕΣ ΣΠΟΥΔΕΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

A web-based GML to stRDF / GeoSPARQL conversion tool

Μάριος Μπεκατώρος

Επιβλέπων : **Μανόλης Κουμπαράκης**, Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΦΕΒΡΟΥΑΡΙΟΣ 2015

MASTER THESIS

A web-based GML to stRDF / GeoSPARQL conversion tool

MARIOS S. BEKATOROS

A.M.: M1187

SUPERVISOR : Μανόλης Κουμπαράκης, Professor

EXAMINATION COMMITEE : Izambo Karali, Assistant Professor

February 2015

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

A web-based GML to stRDF / GeoSPARQL conversion tool

Μάριος Σ. Μπεκατώρος

Α.Μ.: Μ1187

ΕΠΙΒΛΕΠΩΝ : Μανόλης Κουμπαράκης, Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ : Ιζαμπώ Καράλη, Επίκουρη Καθηγήτρια

Φεβρουάριος 2015

To Evi for her support.

Acknowledgements

I want to thanks Professor Manolis Koubarakis for being the supervisor of this Master's thesis and the Strabon team for their help and support, especially Mrs. Stella Giannakopoulou for the guidance she provided.

Abstract

The main goal of this master's thesis is the development of a user-friendly web based tool for GML to stRDF / GeoSPARQL transformations. In this thesis we will talk about open and linked open data, the tools that are available to data scientists for the exploitation of the available linked data and especially geospatial linked data.

Furthermore we will present the motivation for the development of a new tool that is able to transform GML data into RDF.

Tools performing similar transformation already exist but in most cases either they do not follow the standards or offer no integration - extension capabilities. The design decisions will be explained based on the requirements and limitations of the project. The system architecture of the tool will be presented and also the way it was implemented using the latest technologies.

Finally the benefits from its use will be demonstrated by its evaluation in real-world applications.

Subject area: Linked data

Keywords : GML, RDF, Linked data, open data , GeoSPARQL

Περίληψη

Ο κύριος στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη ενός εύχρηστου διαδικτυακού εργαλείου για μετατροπές GML αρχείων σε stRDF / GeoSPARQL. Στη εργασία θα αναφερθούμε στα ανοικτά και διασυνδεδεμένα ανοικτά δεδομένα, τα εργαλεία τα οποία είναι διαθέσιμα στους επιστήμονες δεδομένων για την εκμετάλλευση διαθέσιμων διασυνδεδεμένων δεδομένων και ειδικότερα γεοχωρικών διασυνδεδεμένων δεδομένων.

Επιπλέον θα παρουσιαστούν οι λόγοι για τους οποίους είναι απαραίτητη η δημιουργία ενός νέου εργαλείου που θα μετατρέπει GML δεδομένα σε RDF.

Εργαλεία που πραγματοποιούν αντίστοιχες μετατροπές υπάρχουν ήδη, αλλά στις περισσότερες περιπτώσεις είτε δεν ακολουθούν τα πρότυπα ή δε προσφέρουν δυνατότητες επέκτασης τους. Θα εξηγηθούν οι σχεδιαστικές επιλογές βάσει των απαιτήσεων και των περιορισμών που υπάρχουν. Επίσης θα παρουσιαστεί η αρχιτεκτονική του εργαλείου και ο τρόπος που υλοποιήθηκε χρησιμοποιώντας τις πιο σύγχρονες τεχνολογίες.

Στο τέλος θα παρουσιαστούν τα οφέλη από τη χρήση του εργαλείου μέσα από τη χρήση του σε εφαρμογές του πραγματικού κόσμου.

Θεματική περιοχή: Διασυνδεμένα δεδομένα

Λέξεις-κλειδιά : GML, RDF, Διασυνδεμένα δεδομένα, Ανοικτά δεδομένα, GeoSPARQL

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main aim of this thesis is the creation of a web-based transformation tool for geospatial data. The geospatial data saved in the GML format will be transform to a linked data format like RDF. The geospatial information residing in these files will be transformed following the stRDF or GeoSPARQL standards.

Linked data refers to a set of best practices for publishing and connecting structured data on the Web. RDF is a framework for representing information about Web resources and is an implementation of Linked data.

Due to the abundance of geospatial data in GML format, a domain specific tool is required to create linked data according to stRDF or GeoSPARQL standards. Both standards use the widely adopted OGC standards Well Known Text (WKT) and Geography Markup Language (GML) to represent geospatial data.

By creating this web-based tool even everyday users will be able to perform transformation in an easy use and user-friendly environment. Also this tool will enable data scientist to dedicate more time to creating knowledge from geospatial data rather than trying to transform them to a usable for them format. Data scientist can combine data from multiple sources to create new knowledge from them, adding the geospatial perspective can effect to even better understanding of the data.

Furthermore the datasets that will be transform will be loaded and tested with the Strabon system perfoming complex queries. The development of Strabon started in the context of European FP7 project SemsorGrid4Env[1] and is currently extended with new functional-

[1] http://www.semsorgrid4env.eu

ities in the FP7 project TELEIOS[2] by the University of Athens (UOA) Strabon team.

[2] http://www.earthobservatory.eu/

Chapter 2

Background

2.1 Data

Data [1] is a set of values of qualitative or quantitative variables; restated, pieces of data are individual pieces of information. Data in computing (or data processing) is represented in a structure that is often tabular (represented by rows and columns), a tree (a set of nodes with parent-children relationship), or a graph (a set of connected nodes). Data is typically the result of measurements and can be visualized using graphs or images. Data as an abstract concept can be viewed as the lowest level of abstraction, from which information and then knowledge are derived.

2.1.1 Open Data

Open data [2] is the idea that certain data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control. The goals of the open data movement are similar to those of other ”Open” movements such as open source, open hardware, open content, and open access. The philosophy behind open data has been long established , but the term ”open data” itself is recent, gaining popularity with the rise of the Internet and World Wide Web.

[1] http://en.wikipedia.org/wiki/Data

[2] http://en.wikipedia.org/wiki/Open_data

2.1.2 Linked Data

The term Linked Data [3] refers to a set of best practices for publishing and connecting structured data on the Web. These best practices have been adopted by an increasing number of data providers over the last years, leading to the creation of a global data space containing billions of assertions - the Web of Data. Linked Data is simply about using the Web to create typed links between data from different sources. These may be as diverse as databases maintained by two organisations in different geographical locations, or simply heterogeneous systems within one organisation that, historically, have not easily interoperated at the data level. Technically, Linked Data refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets.

Linked Data does not in general have to be open – there is a lot of important use of linked data internally, and for personal and group-wide data.

2.1.3 Geospatial Data

Geospatial data is information about a physical object that can be represented by numerical values in a geographic coordinate system. Generally speaking, spatial data represents the location, size and shape of an object on planet Earth such as a building, lake, mountain or township. Spatial data may also include attributes that provide more information about the entity that is being represented.

2.1.4 Linked Open Data

Linked Open Data (LOD) is Linked Data which is released under an open licence, which does not impede its reuse for free. Tim Berners-Lee, the inventor of the Web and Linked Data initiator, suggested a 5 star deployment scheme for Open Data. [3]

Under this scheme data are divided in 6 groups.[4]

0-star No star means the data is not available under an open licence, even if it is available on-line.

[3] http://www.w3.org/DesignIssues/LinkedData.html

[4] http://www.epsiplatform.eu

1-star One star means a good start: the data is accessible on the Web. It is readable by the human eye, but not by a software agent, because it is in a 'closed' document format, and therefore cannot be easily re-used.

2-stars Two stars mean that the data is accessible on the Web in a structured, machine-readable format. Thus, the re-user can process, export and publish the data easily, still depending however on proprietary software like Word or Excel.

3-stars Three stars mean that re-users will no longer need to rely on proprietary software (like CSV instead of Excel). Accordingly, re-users can manipulate the data in any way, without being confined to a particular software producer.

4-stars Four stars mean that the data is now in the Web as opposed to on the Web through the use of a URI, a Uniform Resource Identifier . As a URI is completely unique, it gives a fine-granular control over the data, allowing for things like bookmarking and linking.

5-stars Five stars mean that the data is not only in the Web but is also linked to other data, fully exploiting its network effects. Through this interlinking, data gets interconnected whereby the value increases exponentially, since it becomes discoverable from other sources and is given a context (e.g., through links to Wikipedia)

2.1.5 Linked Open Geospatial Data

Populating linked open data with, the domain specific, geospatial data is increasing its value for users. Examples of open linked geospatial data are the Ordnance Survey[5], OpenStreetMap[6] and GeoLinked data [7]. These efforts demonstrate that the development of useful Web applications utilizing geospatial data might be just a few SPARQL queries away. [12]

[5] http://www.ordnancesurvey.co.uk/oswebsite/products/os-opendata.html
[6] http://linkedgeodata.org/
[7] http://geo.linkeddata.es/

2.2 OGC Standards

In this section we present some well-known standards developed by the Open Geospatial Consortium (OGC)[8]. OGC is an international consortium of companies, government agencies and universities participating in a consensus process to develop publicly available interoperability standards for geospatial data.

2.2.1 Well-Known Text

Well-Known Text (WKT) is a widely used OGC standard for representing geometries. WKT can be used for representing geometries, coordinate reference systems, and transformations between coordinate reference systems.

The syntax of the WKT representation of a geometry is presented in detail in [1]. The interpretation of the coordinates of a geometry depends on the coordinate reference system that is associated with the geometry. Note that according to the WKT standard, the coordinate reference system that is associated to a geometry is never embedded in the object's representation, but is given separately using appropriate notation.

2.2.2 Geography Markup Language

The Geography Markup Language (GML) [7] is the most common XML-based encoding standard for the representation of geospatial data. GML was developed by the OGC and it is based on the OGC Abstract Specification. GML provides XML schemas for defining a variety of concepts that are of use in Geography: geographic features, geometry, coordinate reference systems, topology, time and units of measurement. Initially, the GML abstract model was based on RDF and RDFS, but later the consortium decided to use XML and XML Schema

[8] http://www.opengeospatial.org

2.3 Representing and Querying information on the Semantic Web

In this section we introduce the Resource Description Framework (RDF) together with its accompanying vocabulary RDFS. In addition, we present the corresponding standardized query language, called SPARQL (SPARQL Protocol And RDF Query Language), for querying RDF and RDFS data.

2.3.1 RDF

The Resource Description Framework (RDF)[17] is a framework for representing information about Web resources. It consists of W3C recommendations that enable the encoding, exchange and reuse of structured data, providing means for publishing both human-readable and machine-processable vocabularies. Nowadays the current W3C recommendations for RDF are used in a variety of application areas. The Linked Data initiative, which aims connecting data sources on the Web, has already become very popular and has exposed many datasets using RDF and RDFS. DBpedia , BBC music information , government datasets are only a few examples of the constantly increasing Linked Data cloud . The RDF data model offers the following basic concepts:

- Resources: In RDF, a resource is anything that we want to describe in the World Wide Web. A resource may be a Web page, a book, an author, a paper or a computer file. Every resource is uniquely identified by a Universal Resource Identifier (URI) .
- Properties: A property is a characteristic of a resource. For example, "hasName" may be used for describing the name of an author. Properties are also identified by URIs.
- Literals: Literals are constant values of any property.
- Statements: Statements are the constructs offered by RDF for representing information about a domain. A statement has three parts: the resource the statement is about, the property of the resource the statement refers to, and the value of that

property. The three parts of a statement are named, respectively, subject, predicate and object. The object of a statement can be another resource or a literal.

Although the conceptual model of RDF is a graph, RDF provides an XML syntax, called RDF/XML, for writing and exchanging RDF graphs. Apart from the RDF/XML serialization, there are other formats commonly used such as N3 , N-Triples and Turtle.

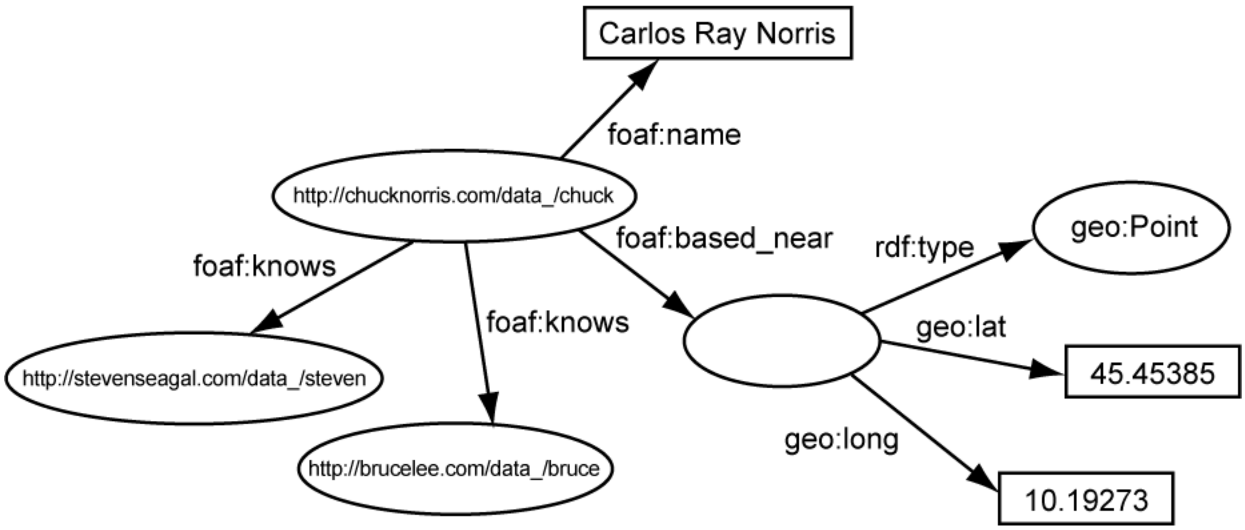

Figure 2.1: Example of a RDF graph

2.3.2 RDF Schema (RDFs)

The RDF data model offers a simple way for describing interrelationships among resources in terms of named properties and values, but does not provide mechanisms for declaring these properties, nor does it provide any ways for defining the relationships between these properties and other resources. This is the role of RDF Schema (RDFS) . RDFS[4] is like a vocabulary of RDF; it provides the means to a user to define terms that will be used in RDF statements and give specific meaning to them. RDFS defines not only the properties of the resource (e.g., title, author, subject etc.) but also the kinds of resources being described (people, paper, Web pages, books etc.). Resources having similar characteristics can be divided into groups called RDFS classes. When we want to refer to both RDF and RDFS information we will use the term RDF(S).

2.3.3 SPARQL

SPARQL[22] is a query language for RDF graphs and has the ability to extract information about both the data and the schema. SPARQL is based on the concept of matching graph patterns. The simplest graph patterns are triple patterns, which are like an RDF triple but with the possibility of a variable in any of the subject, predicate or object positions. A query that contains a conjunction of triple patterns is called basic graph pattern. A basic graph pattern matches a subgraph of the RDF data when RDF terms from the subgraph may be substituted with the variables of the graph pattern. [11]

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?friend
WHERE {
    ?person foaf:name "Carlos Ray Norris" .
    ?person foaf:knows ?friend .
}
```

Table 2.1: Example of a SPARQL query

?friend
http://stevenseagal.com/data_/steven
http://brucelee.com/data_/bruce

Table 2.2: Results of the above SPARQL query based on the dataset shown before

2.3.4 stRDF / stSPARQL - GeoSPARQL

The data model stRDF and the query language stSPARQL are extensions of the standards RDF and SPARQL 1.1 respectively for representing and querying geospatial data that changes over time.

stRDF

stRDF and stSPARQL use the widely adopted OGC standards Well Known Text (WKT) and Geography Markup Language (GML) to represent geospatial data. This is achieved by introducing the datatypes strdf:WKT and strdf:GML for the representation of geometries encoded in WKT and GML respectively. The datatype strdf:geometry is defined as the union of the datatypes strdf:WKT and strdf:GML. For the temporal dimension stRDF and stSPARQL assumes a discrete time line and uses the value space of the datatype xsd:dateTime of XML-Schema to model time. Two kinds of time primitives are supported: time instants and time periods. Time instants are represented using literals of the xsd:dateTime datatype. Time periods are represented by literals of the new datatype strdf:period that we introduce in stRDF. Values of the datatype strdf:period can be used as objects of a triple to represent user-defined time. In addition, they can be used to represent valid times of temporal triples. A temporal triple (quad) is an expression of the form `s p o t.` where `s p o.` is an RDF triple and t is a time instant or a time period called the valid time of a triple. An stRDF graph is a set of triples and temporal triples. The existence of temporal constants NOW and UC inspired from the literature of temporal databases is also assumed. NOW represents the current time and can appear in the beginning or the ending point of a period. UC means “Until Changed” and is used for introducing valid times of a triple that persist until they are explicitly terminated by an update.[14],[13],[2]

stSPARQL

The query language stSPARQL[14] extends SPARQL 1.1 with functions that take as arguments spatial or temporal terms and can be used in the SELECT, FILTER, and HAVING clause of a SPARQL 1.1 query. Functions from the “OpenGIS Simple Feature Access - Part 2: SQL Option” standard (OGC-SFA) are used for querying stRDF data. This standard defines relational schemata that support the storage, retrieval, query and update of sets of simple features using SQL. stSPARQL extends SPARQL 1.1 with the machinery of the OGC-SFA standard. This is achieved by defining a URI for each of the SQL functions defined in the standard and use them in SPARQL queries. Similarly, a Boolean SPARQL extension function is defined for each topological relation defined in OGC-SFA (topological relations for simple features), Egenhofer relations[8] and RCC-8[6] relations. In this way stSPARQL supports multiple families of topological relations the users might be familiar with. Using these functions stSPARQL can express spatial selections and spatial joins. The stSPARQL extension functions can also be used in the SELECT clause of a SPARQL query. As a result, new spatial literals can be generated on the fly during query time based on pre-existing spatial literals. Update operations are also supported in stSPARQL conforming to the W3C standard SPARQL Update 1.1.

The query language stSPARQL is also enabled with valid time support as follows. First, temporal triple patterns are introduced as the most basic way of querying temporal triples. A temporal triple pattern is an expression of the form `s p o t.`, where `s p o.` is a triple pattern and t is a time period or a variable. Second, Temporal extension functions are defined in order to express temporal relations between expressions that evaluate values of the datatypes xsd:dateTime and strdf:period. The first set of such temporal functions are 13 Boolean functions that correspond to the 13 binary relations of Allen's Interval Algebra. stSPARQL offers nine functions that are "syntactic sugar", i.e., they encode frequently-used disjunctions of these relations. stSPARQL also defines functions that allow relating an instant with a period and functions that construct new (closed open) periods from existing ones, as well as temporal aggregates.[13],[2]

GeoSPARQL

GeoSPARQL [20] is a standard for representation and querying of geospatial linked data for the Semantic Web from the Open Geospatial Consortium (OGC).[9] The definition of a small ontology based on well-understood OGC standards is intended to provide a standardized exchange basis for geospatial RDF data which can support both qualitative and quantitative spatial reasoning and querying with the SPARQL database query language. 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.

[9] http://en.wikipedia.org/wiki/GeoSPARQL

2.4 Strabon

Strabon[15] is a semantic spatiotemporal RDF store. You can use it to store linked geospatial data that changes over time and pose stSPARQL and GeoSPARQL queries. Strabon supports spatial datatypes enabling the serialization of geometric objects in the OGC standards WKT 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[5][10] and extends Sesame's components to manage thematic, spatial and temporal data that is stored in the backend RDBMS. The following figure shows the architecture of Strabon.

Sesame was chosen because of its open-source nature, layered architecture, wide range of functionalities and the ability to have, a "spatially enabled" or/and "temporally-enabled" DBMS, as a backend to exploit its variety of spatial functions and operators. Strabon is implemented by creating a layer that is included in Sesame's software stack in a transparent way so that it does not affect its range of functionalities, while benefitting from new versions of Sesame. Strabon uses Sesame and comprises three modules: the storage manager, the query engine and the DBMS. Currently, in Strabon both PostGIS[11] and MonetDB[12] can be used as back-end databases. In order to exploit the temporal features of Strabon, the back-end should also be "temporally-enabled" as well, so PostgreSQL temporal is used in addition to PostGIS.

2.5 Conclusion

All of the above technologies and standards are used by a vast amount of people and especially scientists in order to help them with their research or by using them in innovative new applications that will help even more people. The open data movement especially has

[10]http://rdf4j.org/

[11]http://postgis.net/

[12]https://www.monetdb.org/

seen recently great rise and support even from government organizations.

Chapter 3

Related Work

In this chapter we present related work to our GML to RDF transformation tool. These are tools that perform transformation of data or geodata from a database or a file to some form of RDF. And also languages that were created to enable the transformation between them.

3.1 From GML to Linked Data

In [23] a transformation for translating any correctly structured GML to RDFS/OWL automatically is described, using XSLT[1]. Because GML's object-property structure translates very well to triples, the transformation is straightforward. Well-known GML content elements such as names and descriptions are mapped to their RDF equivalent. The XSLT stylesheet that performs the transformation is publicly available and can be used. This stylesheet does not follow neither the stRDF nor the GeoSPARQL standard and only produces proper RDF graphs .

3.2 R2RML

R2RML[18] is a generic language to describe a set of mappings that translate data from a relational database into RDF. It is the result of preliminary work held by the W3C. Starting from an initial proposal in 2007 , the W3C RDB2RDF Incubator Group ran a comprehensive survey of existing approaches and described high level characteristics that a language

[1] https://en.wikipedia.org/wiki/XSLT

should cover to map relational databases to RDF . Finally, the RDB2RDF Working Group standardized the R2RML mapping language in 2012. As a standard, R2RML has a particular importance as it steers the development of many RDB-to-RDF tools. Compliant tools may therefore adopt completely different strategies.

3.3 D2R Server and the D2RQ language

The D2R Server[2][18] is an open source academic project. It provides an integrated environment with multiple options to access relational data using different methods such as the SPARQL endpoint, Linked Data (content negotiation, HTTP 303 dereferencing), RDF dump. D2RQ supports both direct and domain semantics-driven mappings. The DR2Q declarative mapping language is formally defined by an RDFS schema. It is the successor to the XML-based D2R MAP language.

3.4 Geotriples

GeoTriples[16] is an open source tool, that takes as input geospatial data that are stored in a spatially enabled database or data that reside in raw files or the results that derive from processing of the aforementioned data. Geotriples was developed under the European Project LEO[3]

GeoTriples allows the transformation of the input data into the GeoSPARQL standard. It is implemented as an extension to the D2RQ platform and goes beyond the state of the art by extending the R2RML mapping language to deal with the specifities of geospatial data.

At a lower level, GeoTriples uses a connector for each type of input data that transparently accesses and processes the input data. It also consists of two main components: the mapping generator and the R2RML processor. The mapping generator creates automatically an R2RML mapping document from the input data source. The mapping is also enriched with subject and predicate object maps so that the RDF graph that will be

[2] http://d2rq.org/
[3] http://www.linkedeodata.eu/

produced follows the GeoSPARQL vocabulary. Geospatial information is modeled using a variety of data models (e.g., relational, hierarchical) and is made available using a variety of formats . The R2RML processor is responsible for producing the desired RDF output by taking into account the mapping document generated, which is also optionally edited by the user.

3.5 Sparqlify

Sparqlify [4] is a SPARQL-SQL rewriter that enables one to define RDF views on relational databases and query them with SPARQL. It powers the Linked-Data Interface of the LinkedGeoData Server, where access to billions of virtual triples of OpenStreetMaps database is provided .Sparqlify is a SPARQL-SQL query rewriter that simplifies the definition of RDF views thanks to the Sparqlification Mapping Language Sparqlify mappings are expressed as view definitions based on the SPARQL grammar that has been extended with a few custom production rules. As such, users that know SPARQL are already familiar with most of Sparqlify's syntactic elements.

3.6 morph-RDB

morph-RDB[21] is an RDB2RDF engine developed by the Ontology Engineering Group, which follows the R2RML [5] specification. morph-RDB supports two operational modes:

- Data upgrade, which consists in generating RDF data from a relational database according to the R2RML mapping descriptions.
- Query translation, which allows evaluating SPARQL queries over a virtual RDF dataset, by rewriting those queries into SQL according to the R2RML mapping descriptions.

morph-RDB outperforms similar state-of-the-art tools (D2R) and query translation algorithms by employing various types of optimisations during the query rewriting process, so as to generate more efficient SQL queries.

[4] http://sparqlify.org/
[5] http://www.w3.org/TR/r2rml/

At the moment, morph-RDB works with relational database management systems like MySQL, PostgreSQL and MonetDB.

3.7 Geometry2RDF

Geometry2RDF[6][16] is a library for generating RDF files for geometrical information (which could be available in GML or WKT). The GML and WKT is manipulated with GeoTools. Geometry2rdf was the first tool that allows for the conversion of geospatial information that resides in a spatially-enabled relational database into an RDF graph. It takes as input data stored in a spatially enabled DBMS like Oracle Spatial or MySQL, and utilizes the libraries Jena and GeoTools to produce an RDF graph. Geometry2RDF follows the direct mapping approach, and allows the user to configure the properties that connect a URI to the serialization of a geometry and allows for the conversion of the coordinates to the desired coordinate reference system. Geometry2RDF follows the direct mapping approach that is not expressive enough to deal with the specificities of the geospatial domain.

3.8 TripleGeo

TripleGeo[7][19] is a utility developed by the Institute for the Management of Information Systems at Athena Research Center under the EU/FP7 project GeoKnow[8]: This generic purpose, open-source tool can be used for integrating features from geospatial databases into RDF triples. TripleGeo can directly access both geometric representations and thematic attributes either from standard geographic formats or widely used DBMSs. It can also reproject input geometries on-the-fly into a different Coordinate Reference System, before exporting the resulting triples into a variety of notations. It supports the GeoSPARQL standard , although it can extract geometries into other vocabularies as well.

TripleGeo is based on open-source utility geometry2rdf.

[6] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/technologies/151-geometry2rdf

[7] https://web.imis.athena-innovation.gr/redmine/projects/geoknow_public/wiki/TripleGeo

[8] http://geoknow.eu

3.9 GeomRDF

GeomRDF[10] ia a tool that helps users to convert spatial data from traditional GIS formats to RDF model easily. It generates geometries represented as GeoSPARQL WKT literal but also as structured geometries that can be exploited by using only the RDF query language, SPARQL. GeomRDF was implemented as a module in the RDF publication platform Datalift[9].

3.10 Conclusion - Match-up

Geometry2RDF has preceded the proposal of GeoSPARQL and stRDF, so it does not produce RDF graphs according to these vocabularies. In contrast TripleGeo as a newer utility can perform various transformations following the GeoSPARQL standard, but no the stSPARQL. GeomRDF also shares the same principles and architecture as a utility with TripleGeo, transforming following GeoSPARQL, but is integrated with Datalift. GeoTriples can produce transformation from various formats to GeoSPARQL or stRDF via R2RML mapping and is released as an open source project under the Mozilla Public License v2.0[10], the source code[11] became accessible in the Github repository, after the development of the transformation tool so the source code could not be reviewed for extension possibilities. [23]'s XSLT can be used as a base, as it produces valid RDF documents, and can be extended in order to follow the standards.

[9] http://datalift.org/

[10] https://www.mozilla.org/MPL/

[11] https://github.com/LinkedEOData/GeoTriples

Chapter 4

System Architecture

4.1 Multi-tier Architecture

Multi-tier architecture[1] is a client – server architecture in which the presentation, the application processing, and the data management are logically separate processes. For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture. The most widespread use of multi-tier architecture is the three-tier architecture.

Multi-tier application architecture provides a model for developers to create a flexible and reusable application. By breaking up an application into tiers, developers only have to modify or add a specific layer, rather than have to rewrite the entire application over. There should be a presentation tier, a business or data access tier, and a data tier.

The concepts of layer and tier are often used interchangeably. However, one fairly common point of view is that there is indeed a difference, and that a layer is a logical structuring mechanism for the elements that make up the software solution, while a tier is a physical structuring mechanism for the system infrastructure

In the web development field, three-tier is often used to refer to websites, commonly electronic commerce websites, which are built using three tiers:

1. A front-end web server serving static content, and potentially some cached dynamic content. In web based application, Front End is the content rendered by the browser. The content may be static or generated dynamically.

[1] `http://en.wikipedia.org/wiki/Multitier_architecture`

2. A middle dynamic content processing and generation level application server, for example Ruby on Rails, Java EE, ASP.NET, PHP, ColdFusion, Perl, Python platform.

3. A back-end database or data store, comprising both data sets and the database management system software that manages and provides access to the data.

4.2 Technologies

In this section we will describe the technologies that will be used in the development phase of this tool.

4.2.1 Java - JEE

Java

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

JEE

JEE (Java Enterprise Edition)[2] follows the distributed multi-tiered application approach which means the entire application may not reside at a single location, but distributed. And the applications is divided into various tiers. J2ee application is composed of various components which can be created by the different developers and then assembled together. These components can be installed on different machines across various tiers. And the distribution of application components across the solely different tiers depends upon the functionality it performs.

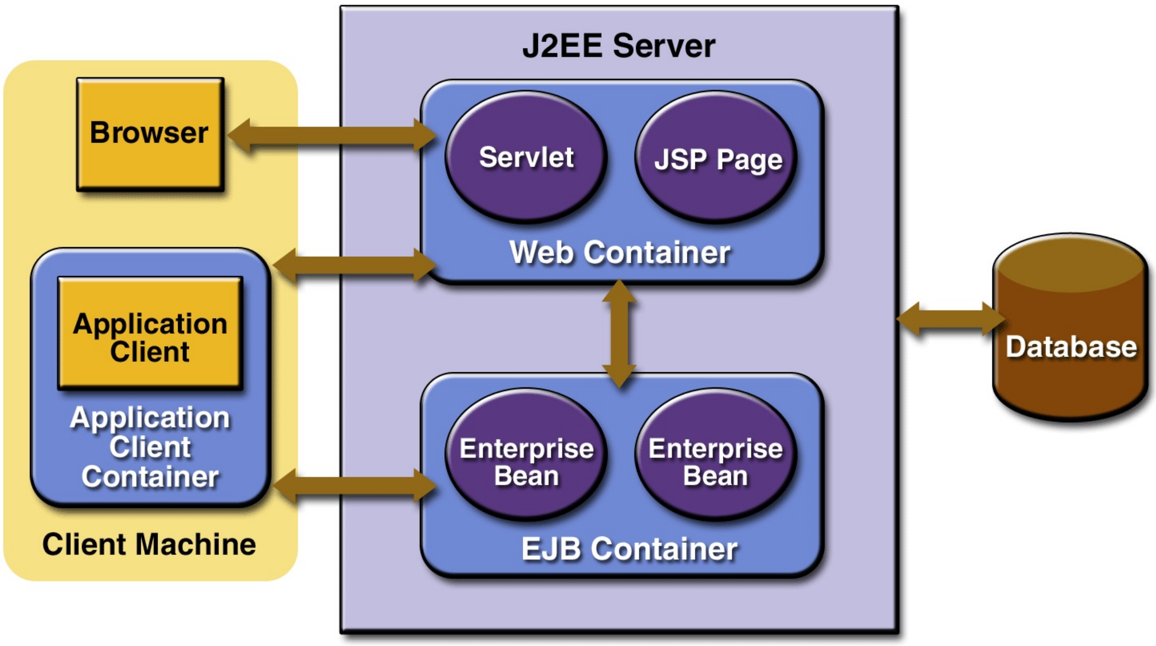

Figure 4.1: JEE Architecture

J2EE application tiers are as follows

Client Tier There are two type of clients in this tier , a web client, generally a browser, sends a request to the server and renders the webpage sent back by the server. However, it may not perform any complex tasks such as querying a database or performing any complicated business tasks and hence is also referred to as thin client. The complex tasks are performed on the server. Or an application client, a

[2] http://www.deknight.com/java/jee-architecture-java-enterprise-edition-architecture.html

standalone application that doesn't run in browsers. It can access components in the business tier directly.

Web tier In Java this tier can be created with Servlets and JavaServer Pages (JSP). A servlet receives HTTP requests from the client, processes them, and returns the output. It can also generate dynamic responses. Similar to servlets, JSP also create dynamic web pages. JSP pages are converted into servlets and executed within a servlet container. They are used to display the results processed by a servlet.

Business Tier This tier implements the business logic or the functional process logic that defines the business rules and constraints. For example business process such as withdrawing amount from the bank.

Enterprise Information tier This tier encompasses all enterprise backend resources such as databases, BackOffice/legacy systems, ERP implementations, XML or flatfiles

4.2.2 XSLT

XSLT (eXtensible Stylesheet Language Transformations)[3] is a language for transforming XML documents into other XML documents, or other formats such as HTML for web pages, plain text or into XSL Formatting Objects, which may subsequently be converted to other formats, such as PDF PostScript and PNG.

The original document is not changed; rather, a new document is created based on the content of an existing one. Typically, input documents are XML files, but anything from which the processor can build an XQuery and XPath Data Model can be used, for example relational database tables, or geographical information systems.

The XSLT processor takes one or more XML source documents, plus one or more XSLT stylesheets, and processes them to produce an output document. In contrast to widely-implemented imperative programming languages like C, XSLT is declarative. This makes a given XSLT program more resilient to change to the input it is likely to receive, useful in a language used for information processing applications. The basic processing paradigm is pattern matching.

[3] https://en.wikipedia.org/wiki/XSLT

The processor follows a fixed algorithm. First, assuming a stylesheet has already been read and prepared, the processor builds a source tree from the input XML document. It then processes the source tree's root node, finds the best-matching template for that node in the stylesheet, and evaluates the template's contents. Instructions in each template generally direct the processor to either create nodes in the result tree, or to process more nodes in the source tree in the same way as the root node. Output derives from the result tree.

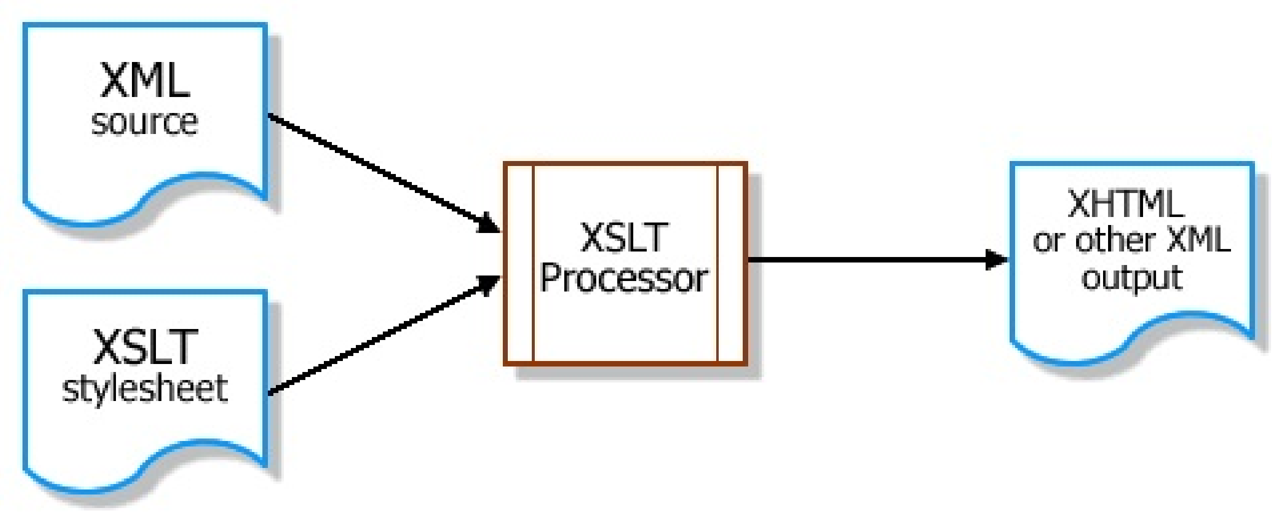

Figure 4.2: XSL Transformation

4.2.3 HTML5

HTML5[4] is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web. As of October 2014 this is the final and complete fifth revision of the HTML standard of the World Wide Web Consortium (W3C). The previous version, HTML 4, was standardised in 1997.

Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices (web browsers, parsers, etc.). HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

Following its immediate predecessors HTML 4.01 and XHTML 1.1, HTML5 is a re-

[4] https://en.wikipedia.org/wiki/HTML5

sponse to the fact that the HTML and XHTML in common use on the World Wide Web are a mixture of features introduced by various specifications, along with those introduced by software products such as web browsers, those established by common practice, and the many syntax errors in existing web documents. It is also an attempt to define a single markup language that can be written in either HTML or XHTML syntax.

4.2.4 CSS

Cascading Style Sheets (CSS)[5] is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts.This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as semantically insignificant tables that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C)

[5] https://en.wikipedia.org/wiki/Cascading_Style_Sheets

4.3 Conclusion

The technologies and architecture that will be used in this development of this new tool are the latest and most supported by the industry and the community.

Chapter 5

System design

5.1 Typical process of transforming geo data to linked geo data

When a researcher needs to use linked geodata for a specific reason , in most cases , there won't be available as linked data but probably there will be an abundant sea of open geodata available in other formats as SHP[9] or GML. The first thing a researcher must do after reviewing and selecting his dataset is transforming it to linked data following a geospatial standard like GeoSPARQL or stRDF. To accomplice this a tool is needed. Most of the available options are utilities using command line interfaces or minimal graphical or web user interfaces

5.2 GML to RDF transformation methods

- **OpenRefine**[1] is an open source tool that enables transformation of data from various formats like CSV, XML, XLS e.t.c. to another. Extension of this tools has been made to enable the creation of linked data like LODRefine.The transformations are made via manually mapping the objects of your dataset. So a GML to RDF transformation is possible but neither useful for multiple datasets with different data nor able to support the geospatial standards. OpenRefine provides a useful Web UI but can not provide

[1] `http://openrefine.org/`

the needed functionality.

- **Python scripting** Another option for data transformation is manually creating a script in Python[2] or other computer languages that will provide the transformation needed. By mannually scripting you can export the data following any standard you prefer. Usually this kind of scripts need to be rewritten for each different type of data and mostly via CLI.

- **Direct Transformation** Using tools like Geometry2RDF you can perform direct transformation from GML to RDF, this kind of tools, demand complex coding in order to be extended , and also require a database in order to temporary store the data. Any attempt to extend this king of tools will probably result in rewriting a large percentage of the codebase.

- **XSLT** GML is based on the XML file format as is also RDF/XML, so using a XSL stylesheet transformation is an easy solution with many fast and open source implementations. The output result of a transformation of this kind will be a valid RDF file. By changing the XSL file creating a RDF that implements the GeoSPARQL or stRDF standard is straightforward. Using a xsl transformer library enables the creation of a Web or Graphical UI .

5.3 Proposed solution for transforming GML to stRDF and GeoSPARQL

The solution that is proposed implements a Web UI with a user friendly environment, the user will upload the GML file that needs to be transformed, fill in the necessary data, select between the available vocabularies (GML, GeoSPARQL, stRDF) and also between the available formats (RDF/XML, Turtle, N3, N-Triples). After the successful transformation the user will be able to download the file for inspection and usage.

Using a Web UI via an application server enables us to provide this transformation as an open public service and helps users that don't have a high technical expertise to transform

[2] https://www.python.org/

their geodata to linked geodata. This will not restrict other users from downloading the source code and implementing, extending or using the solution locally.

In order to create an open, easy to use and extendable interface and implementation, a user management system will not be used because the identification of users is not crucial. So a database management system (DBMS) will not be used, configuration options, if needed, can be stored in simple files.

This system follows the multitier Client - Server architecture[3], and will includes a Presentation Tier using a Web Server and the Logic Tier, performing the transformations, that will be handled by the Application Server.

The transformations will be carried out either from direct XSL transformations , e.g. for performing vocabulary transformation, or by using available libraries to perform them, e.g. for performing XML to N-triples format transformation.

5.4 Conclusion

Proposing a solution like the previously presented is made following the main criteria of user-friendliness, ease of development and available resource. If different aims were set a different solution would probably arise.

[3] https://en.wikipedia.org/wiki/Multitier_architecture

Chapter 6

System Implementation

In this chapter we will describe the transformation process and how it is implemented, as it is described in the previous chapter.

6.1 Transformation Workflow

Firstly we will describe the transformation workflow and the stages that are necessary for a successful GML to RDF transformation. There are two essential and two optional stages.

- Preprocessing
- GML to RDF
- RDF to GeoSPARQL / stRDF (optional)
- RDF to RDF (optional)

6.1.1 Preprocessing stage

In this stage the input GML file is cleared of unnecessary tags that do not provide any added value in the data. And could possibly make the transformation fail. This preproccesing is done using an XSLT file that you can find in Appendix A.1

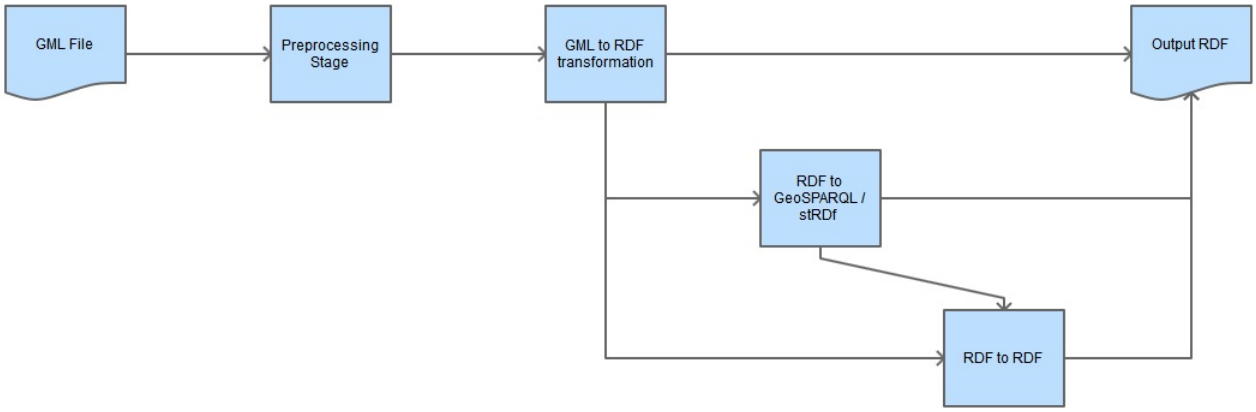

Figure 6.1: Transformation Workflow

6.1.2 GML to RDF transformation stage

In this stage the main GML to RDF transformation is carried out using a modified XSLT file that is based on [23]. The file was modified in order to produce RDF files enabling the usatge of the GeoSPARQL/stRDF standards. The output file includes the geospatial information in a strdf:hasGeometry tag and also all the other valid information accompanying the dataset. The XSLT file can be found in Appendix A.2

6.1.3 RDF to GeoSPARQL / stRDF transformation stage

In this optional stage the file has been already transformed to a valid RDF/XML file with embedded geodata. If the user has selected to transform the data to the GeoSPARQL or the stRDF vocabulary then this XSL transformation occurs. For each of the aforementioned vocabularies, a different XSL file was created following their standards. A XSLT file from 3.8, GML2WKT[1], was used as a base in order to create the two different transformation stylesheets, one for each vocabulary. These two XSLT files can be found in Appendix A.3 A.4

[1] https://github.com/GeoKnow/TripleGeo/blob/master/xslt/GML2WKT.xsl

6.1.4 RDF to RDF format transformation stage

At this optional stage the RDF/XML file that was produced by the previous stages can now be transformed to other RDF formats like n-triples and N3. If the user has selected this type of format for its output file the, RDF/XML file is passed through the RDF2RDF converter[2] and transformed to the desired format.

Here you can see a transformed snippets from a RDF/XML and a N3 file

```
<rdf:Description rdf:about="http://strdf.di.uoa.gr/ontology#d15e3"
rdf:type="http://www.opengis.net/gml#elegxomenes_kynhgetikes_perioxes">
<ogr:geometryProperty rdf:resource="#d15e5"/>
<ogr:apofash>163921/2035/12-7-1979</ogr:apofash>
<ogr:area>420447.481</ogr:area>
<ogr:area_strem>420.45</ogr:area_strem>
<ogr:ektash>250</ogr:ektash>
<ogr:epopteuous>ΔΑΣΑΡΧΕΙΟ ΣΚΟΠΕΛΟΥ</ogr:epopteuous>
<ogr:fek>744/B/4-9-1979</ogr:fek>
<ogr:nomos>ΜΑΓΝΗΣΙΑΣ</ogr:nomos>
<ogr:panida_thi>ΑΙΓΑΓΡΟΣ</ogr:panida_thi>
<ogr:perifereia>ΘΕΣΣΑΛΙΑ</ogr:perifereia>
<ogr:perimeter>2875.821</ogr:perimeter>
<ogr:thesh>ΝΗΣΟΣ ΑΓ. ΓΕΩΡΓΙΟΣ</ogr:thesh>
<strdf:hasGeometry rdf:datatype= "http://strdf.di.uoa.gr/ontology#WKT" >
 MULTIPOLYGON((( 482871.277100000006612 4331947.822999999858439,
  482968.395000000018626 4331905.615899999625981,
.
.
.
 482841.718999999982771 4331952.043700000271201,
 482875.49890000000596 4331952.043700000271201,
 482871.277100000006612 4331947.822999999858439)))
 ;http://www.opengis.net/def/crs/EPSG/0/2100
  </strdf:hasGeometry>
</rdf:Description>
```

RDF/XML format

```
strdf:d15e3 a <http://www.opengis.net/gml#elegxomenes_kynhgetikes_perioxes> ;
ogr:geometryProperty <unknown:namespace#d15e5> ;
ogr:apofash "163921/2035/12-7-1979" ;
ogr:area "420447.481" ;
ogr:area_strem "420.45" ;
ogr:ektash "250" ;
ogr:epopteuous "ΔΑΣΑΡΧΕΙΟ ΣΚΟΠΕΛΟΥ" ;
ogr:fek "744/B/4-9-1979" ;
ogr:nomos "ΜΑΓΝΗΣΙΑΣ" ;
ogr:panida_thi "ΑΙΓΑΓΡΟΣ" ;
```

[2] http://www.l3s.de/~minack/rdf2rdf/

```
ogr:perifereia "ΘΕΣΣΑΛΙΑ" ;
ogr:perimeter "2875.821" ;
ogr:thesh "ΝΗΣΟΣ ΑΓ. ΓΕΩΡΓΙΟΣ" ;
strdf:hasGeometry "MULTIPOLYGON
(((482871.277100000006612 4331947.822999999858439,
 482968.395000000018626 4331905.615899999625981,
.
.
.
482871.277100000006612 4331947.822999999858439)))
 ;http://www.opengis.net/def/crs/EPSG/0/2100"^^strdf:WKT .
```

N3 format

6.2 Necessary Java Libraries

In this section we will describe the Java Libraries that are essential for the project in order to successfully transform the datasets. Firstly a Java library is just a collection of class definitions. The reason behind is simply code reuse, i.e. get the code that has already been written by other developers. The classes and methods normally define specific operations in a domain specific area. For example, there are some libraries of mathematics which can let developer just call the function without redo the implementation of how an algorithm works. The libraries that was used in this project are :

- **RDF2RDF**[3] converts RDF files from any format to any format. It is wrapped into one single jar file for easy usage.
- **Kernow**[4] is a tool designed to make it faster and easier to repeatedly run XSLT transforms, XQuery and XML Schema using Saxon[5]
- **UploadBean**[6] is a JAVA component that allows to upload files from browser. This bean can be integrated in any JSP/Servlets application.
- **C**om **O**reilly **S**ervlet (**COS**)[7] is a utility class to handle multipart/form-data requests, the kind of requests that support file uploads. This class can receive arbitrarily large

[3] http://www.l3s.de/~minack/rdf2rdf/
[4] http://kernowforsaxon.sourceforge.net/
[5] http://saxon.sourceforge.net/
[6] http://www.javazoom.net/jzservlets/uploadbean/uploadbean.html
[7] http://www.servlets.com/cos/

files (up to an artificial limit you can set), and fairly efficiently too.

6.3 Front End Development

In this section we will describe how the front end of the tool was developed. The use of a front end framework was decided to enable easier and faster developement, without the need to start from scratch. This is also the most common strategy when developing a new web tool. The Bootstrap V3 Framework[8] was chosen for this reason. Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. Bootstrap was also programmed to support both HTML5 and CSS3.

A basic responsive template was downloaded using the `http://www.initializr.com/` to kick-start the development of the tool.

This basic HTML page was cut in three pieces in order to make it more reusable.

- **Header** includes the header tags for the HTML page and also the navigation bar that is shown in every page. This navigation bar is responsive and adjust its size relatively to the screen size or orientation.

- **Body** is used as a container for the content we want to display to the user on each page of the tool.

- **Footer** includes the footer tags of the page, the sources for javascript files used in the bootstrap framework and closing HTML tag of the document

6.4 Basic Web Pages

The web application consists of three basic web pages. These are responsible for submitting the data and GML, transforming the data and returning the transformed file.

6.4.1 Index

This web page is the index page of the web application. The user must fill out the necessary information in this page. The data that the user must provide except the GML file are

[8] `http://getbootstrap.com/`

- **Geo Property** is the geospatial property that embeds the geometry information of the file, e.g. ogr:geometryProperty
- **Namespaces** the application has a list of preloaded namespaces. If the users GML document has more namespaces loaded, the user must insert them.
- **Global Reference System**, the user must fill out the GRS of the file in order to produce the correct input for system like Strabon.
- **Output Filetype**, the user can select between RDF/XML, N-Triples, Turtle and N3
- **Vocabulary**, the user can select between three different vocabularies GML, geoSPARQL and stRDF
- the user also must submit her valid GML file.

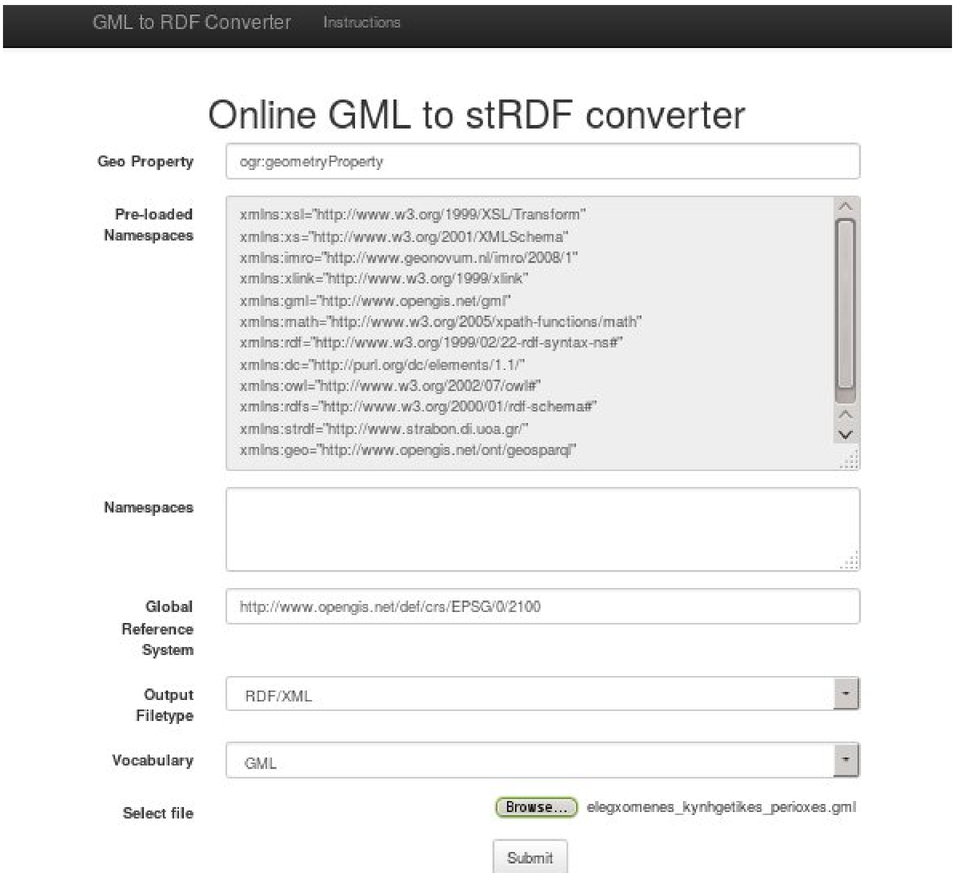

Figure 6.2: Index Form Page

6.4.2 File Upload and transformation

After the submission of the data, all the selected and aforementioned transformation takes place in the upload webpage. The GML file is firstly stored locally and then the transformations are applied. After a successful transformation the user can download the file from the appropriate link and then she can perform a new transformation.

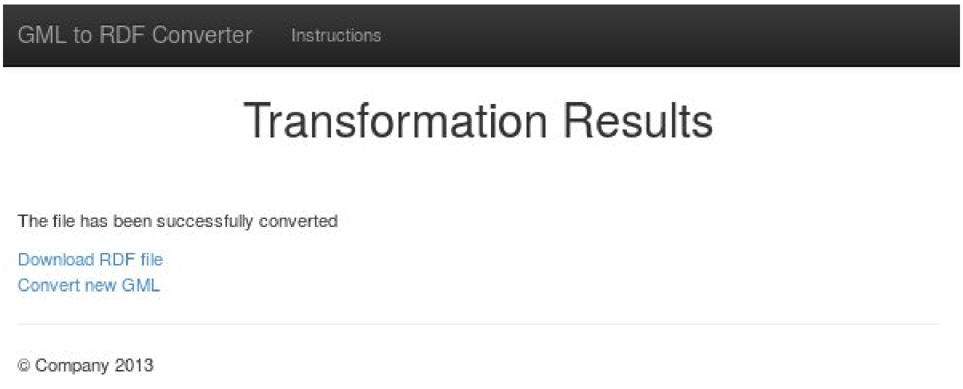

Figure 6.3: Upload and proccessing page

In case of a failed transformation mainly because of a missing namespace an error message appears and the user must submit the correct namespaces.

6.4.3 Output download

This web page serves for returning the output file to the user and then delete from the servers file system, so it won't fill its hard drives. After downloading the file, the user can transform a new GML file or check the results of the transformation. This web page takes as input from GET variables the name of the file, the name is a random numerical string, and also the file type of the document, e.g. rdf, n3, ttl etc.

6.5 Conclusion

These simple web pages that are used in the transformation tool make it very easy for non experts to transform GML files to RDF and in various formats and vocabularies. Also the tool is build in a way that it helps users perform the transformations. The tool is available for download as open source from `https://github.com/bekatoros/gml2rdf`, installation instructions are also available there.

Chapter 7

Output Results

The aim of this chapter is to demonstrate the data that was produced using the the GML to RDF transformation system.

7.1 Finding Geospatial Data

The geospatial data in GML format that was used for testing purposes needed to comply to some criteria

1. embedded information, to enrich the final linked data that is produced with valuable metadata

2. open, the used dataset must be open and freely available for everyone to use , to avoid copyright issues and have the ability to publicly provide the results.

In order to find geospatial data matching this criteria the search was aimed at open national databanks like `http://geodata.gov.gr` and `htt://ckan.okfn.gr`. These sites provide open geospatial data in various formats like gml or kml etc and also the most datasets have embedded information.

7.2 Natura 2000 Protected areas in Greece

The first dataset that was examined and transformed to RDF was about the protected by Natura 2000 Network areas in Greece. Natura 2000 [1] is a network of nature protection areas in the territory of the European Union. The network includes both terrestrial and marine sites. Natura 2000 protects around 18

An example of the dataset is shown here (Table 7.1) The dataset include information about the name of the Natura area, the unique code, its perimeter's length in meters and space in hectares.

[1] `http://en.wikipedia.org/wiki/Natura_2000`

```
<gml:featureMember>
  <ogr:natura fid=”F0”>
    <ogr:geometryProperty><gml:Polygon><gml:outerBoundaryIs>
    <gml:LinearRing><gml:coordinates>400527.093799999915063,4482831.5 400818.125,
    4482535.5 400761.218799999915063,4482031.5 400687.218799999915063,
    4481500.5 400708.906299999915063,4480947.5 400714.8125,
    4480444.5 400291.875,4480661.5 399837.5625,4480652.5 399352.843799999915063,
    4480667.5 399153.468799999915063,4480715.5 399091.0,4480730.5 398964.625,
    4480760.5 399071.3125,4481139.5 399153.718799999915063,
    4481619.5 398961.718799999915063,4481993.5 398980.125,
    4482396.5 398892.718799999915063,4482822.5 399334.4375,
    4482832.5 399699.156299999915063,4482789.5 400055.593799999915063,
    4482947.5 400322.093799999915063,4482928.5 400527.093799999915063,4482831.5
    </gml:coordinates></gml:LinearRing></gml:outerBoundaryIs>
    </gml:Polygon></ogr:geometryProperty>
    <ogr:OBJECTID>0</ogr:OBJECTID>
    <ogr:CODE>GR1220005</ogr:CODE>
    <ogr:AREA>3771983.6090500001</ogr:AREA>
    <ogr:PERIMETER>8161.1978623400</ogr:PERIMETER>
    <ogr:HECTARES>377.1980000000</ogr:HECTARES>
    <ogr:SITETYPE>SPASCI</ogr:SITETYPE>
    <ogr:PERIPHERY>ΚΕΝΤΡΙΚΗ ΜΑΚΕΔΟΝΙΑ</ogr:PERIPHERY>
    <ogr:PREFECTURE>ΘΕΣΣΑΛΟΝΙΚΗ</ogr:PREFECTURE>
    <ogr:NAME_LATIN>LIMNOTHALASSA ANGELOCHORIOU</ogr:NAME_LATIN>
  </ogr:natura>
 </gml:featureMember>
```

Table 7.1: Natura GML Fraction

The table 7.2 demonstrates the result of the GML to RDF transformation using the GeoSPARQL vocabulary and the RDF/XML file format.

```
<rdf:Description rdf:about=”http://strdf.di.uoa.gr/ontology#d3e3”
                    rdf:type=”http://www.opengis.net/gml#natura”>
    <ogr:geometryProperty rdf:resource=”#d3e5”/>
    <ogr:OBJECTID>0</ogr:OBJECTID>
    <ogr:CODE>GR1220005</ogr:CODE>
    <ogr:AREA>3771983.6090500001</ogr:AREA>
    <ogr:PERIMETER>8161.1978623400</ogr:PERIMETER>
    <ogr:HECTARES>377.1980000000</ogr:HECTARES>
    <ogr:SITETYPE>SPASCI</ogr:SITETYPE>
    <ogr:PERIPHERY>ΚΕΝΤΡΙΚΗ ΜΑΚΕΔΟΝΙΑ</ogr:PERIPHERY>
    <ogr:PREFECTURE>ΘΕΣΣΑΛΟΝΙΚΗ</ogr:PREFECTURE>
    <ogr:NAME_LATIN>LIMNOTHALASSA ANGELOCHORIOU</ogr:NAME_LATIN>
    <strdf:hasGeometry rdf:datatype=”http://strdf.di.uoa.gr/ontology#WKT”>
      <rdf:Description>
       <geo:asWKT rdf:datatype=”http://www.opengis.net/ont/geosparql#wktLiteral”>
       &lt;http://www.opengis.net/def/crs/EPSG/0/2100&gt;
       POLYGON((400527.093799999915063 4482831.5,400818.125 4482535.5,
       400761.218799999915063 4482031.5, 400687.218799999915063 4481500.5,
       400708.906299999915063 4480947.5, 400714.8125 4480444.5,
       400291.875 4480661.5,399837.5625 4480652.5,
       399352.843799999915063 4480667.5,399153.468799999915063,
       4480715.5,399091.0 4480730.5,398964.625 4480760.5,
       399071.3125 4481139.5, 399153.718799999915063 4481619.5,
       398961.718799999915063 4481993.5, 398980.125 4482396.5,
       398892.718799999915063 4482822.5, 399334.4375 4482832.5,
       399699.156299999915063 4482789.5,400055.593799999915063 4482947.5,
       400322.093799999915063 4482928.5, 400527.093799999915063 4482831.5))
       </geo:asWKT>
       <rdf:type rdf:resource=”http://www.opengis.net/gml#Polygon”/>
      </rdf:Description>
    </strdf:hasGeometry>
  </rdf:Description>
```


Table 7.2: Natura RDF Fraction

After inserting the output RDF in the Strabon system we can execute queries. You can see an example query and its output in KML format. (Table 7.3, 7.4 and Figure 7.1) This query will provide us the geometry of the area with unique code GR2320001.

```
PREFIX strdf: <http://strdf.di.uoa.gr/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql>
PREFIX ogr: <http://ogr.maptools.org/>

SELECT ?result
WHERE {
        ?a ogr:CODE "GR2320001" .
        ?a strdf:hasGeometry ?f.
        ?f <http://www.opengis.net/ont/geosparql#asWKT> ?result
}
limit 1
```

Table 7.3: Natura stSPARQL query

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://www.opengis.net/kml/2.2'>
<Folder>
<Placemark>
<name>GR2320001</name>
<Polygon>
    <outerBoundaryIs>
        <LinearRing>
            <coordinates>
            21.384391432425858,38.10539935844492 21.38331552896756,
            .
            .
            .
            ,38.109550990198905 21.384391432425858,38.10539935844492
            </coordinates>
        </LinearRing>
    </outerBoundaryIs>
</Polygon>
</Placemark>
</Folder>
</kml>
```

Table 7.4: Natura KML result

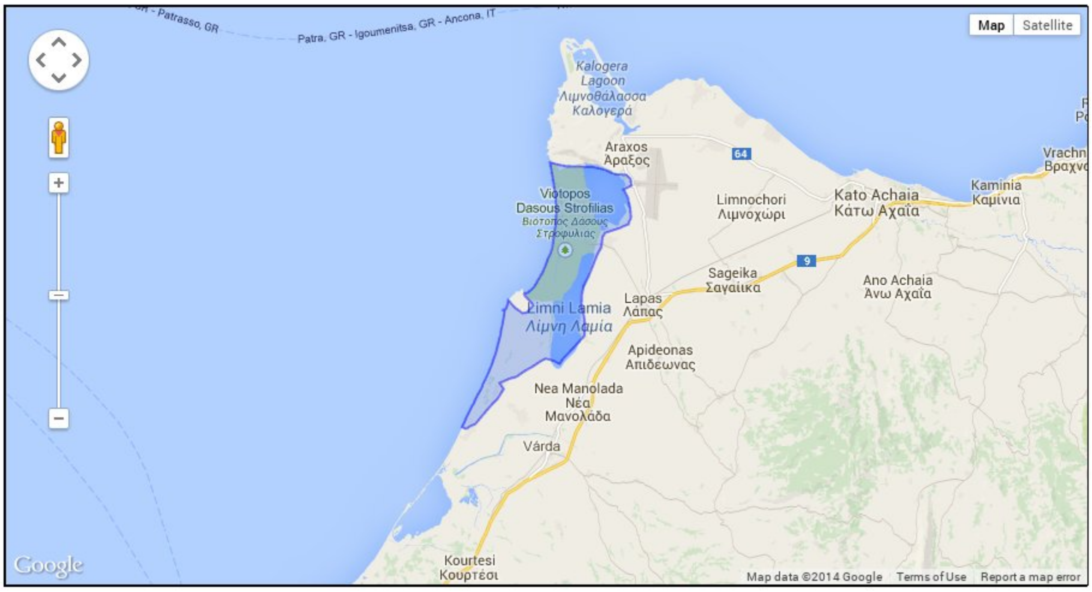

Figure 7.1: Natura area

7.3 Wildlife Refuges in Greece

Wildlife refuges are areas that are protected by national law and is illegal to hunt the fauna that resides in them.

An example of the dataset is shown here (Table 7.5)

```
<gml:featureMember>
  <ogr:katafygia_agrias_zwhs fid=”F6”>
    <ogr:geometryProperty><gml:MultiPolygon srsName=”EPSG:2100”>
    <gml:polygonMember><gml:Polygon><gml:outerBoundaryIs>
    <gml:LinearRing><gml:coordinates>786769.120470919995569,
    4085865.706098780035973 788104.389848164049909,4086214.713971099816263
    789015.829721012967639,4086685.776499540079385 789474.651353440014645,
    788511.399128542980179,4084316.158577340189368 788455.597272490966134,
    .
    .
    .
    4085462.821490400005132 786769.120470919995569,4085865.706098780035973
    </gml:coordinates></gml:LinearRing></gml:outerBoundaryIs>
    </gml:Polygon></gml:polygonMember></gml:MultiPolygon>
    </ogr:geometryProperty>
    <ogr:apofash>3659/12-7-00</ogr:apofash>
    <ogr:area>21933940.764</ogr:area>
    <ogr:area_strem>21933.94</ogr:area_strem>
    <ogr:ektash>20950</ogr:ektash>
    <ogr:epopteyoys>ΔΑΣΑΡΧΕΙΟ ΘΗΡΑΣ</ogr:epopteyoys>
    <ogr:nomos>ΔΩΔΕΚΑΝΗΣΟΥ</ogr:nomos>
    <ogr:perifereia>ΑΙΓΑΙΟΥ</ogr:perifereia>
    <ogr:perimeter>35014.719</ogr:perimeter>
    <ogr:thesh>ΑΓ. ΦΩΚΑΣ - ΨΑΛΙΔΙ - ΚΑΣΤΕΛΟ - ΗΡΑΚΛΗΣ</ogr:thesh>
  </ogr:katafygia_agrias_zwhs>
</gml:featureMember>
```

Table 7.5: Wildlife Refuges GML Fraction

The table 7.6 demonstrates the result of the GML to RDF transformation using the GeoSPARQL vocabulary and the RDF/XML file format.

```
<rdf:Description rdf:about="http://strdf.di.uoa.gr/ontology#d3e176"
    rdf:type="http://www.opengis.net/gml#katafygia_agrias_zwhs">
     <ogr:geometryProperty rdf:resource="#d3e178"/>
     <ogr:apofash>3659/12-7-00</ogr:apofash>
     <ogr:area>21933940.764</ogr:area>
     <ogr:area_strem>21933.94</ogr:area_strem>
     <ogr:ektash>20950</ogr:ektash>
     <ogr:epopteyoys>ΔΑΣΑΡΧΕΙΟ ΘΗΡΑΣ</ogr:epopteyoys>
     <ogr:nomos>ΔΩΔΕΚΑΝΗΣΟΥ</ogr:nomos>
     <ogr:perifereia>ΑΙΓΑΙΟΥ</ogr:perifereia>
     <ogr:perimeter>35014.719</ogr:perimeter>
     <ogr:thesh>ΑΓ. ΦΩΚΑΣ - ΨΑΛΙΔΙ - ΚΑΣΤΕΛΟ - ΗΡΑΚΛΗΣ</ogr:thesh>
     <strdf:hasGeometry rdf:datatype="http://strdf.di.uoa.gr/ontology#WKT">
        <rdf:Description>
           <geo:asWKT rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral">
           &lt;http://www.opengis.net/def/crs/EPSG/0/2100&gt;
           MULTIPOLYGON(((786769.120470919995569 4085865.706098780035973,
           788104.389848164049909 4086214.713971099816263,
           .
           .
           .
           786986.128599122050218 4085462.821490400005132,
           786769.120470919995569 4085865.706098780035973))) </geo:asWKT>
           <rdf:type rdf:resource="http://www.opengis.net/gml#MultiPolygon"/>
        </rdf:Description>
     </strdf:hasGeometry>
  </rdf:Description>
```

Table 7.6: Wildlife refugees RDF Fraction

After inserting the output RDF in the Strabon system we can execute queries. You can see an example query and its output in KML format. (Table 7.7, 7.8 and Figure 7.2)

```
PREFIX strdf: <http://strdf.di.uoa.gr/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql>
PREFIX ogr: <http://ogr.maptools.org/>

SELECT ?result
WHERE {
        ?a ogr:nomos "ΤΡΙΚΑΛΩΝ" .
        ?a strdf:hasGeometry ?f.
        ?f <http://www.opengis.net/ont/geosparql#asWKT> ?result
}
```

Table 7.7: Wildlife stSPARQL query

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://www.opengis.net/kml/2.2'>
<Folder>
<Placemark>
<name>Result0</name>
<MultiGeometry>
    <Polygon>
        <outerBoundaryIs>
            <LinearRing>
                <coordinates>21.521487158064158,
                39.30216403545025 21.521661268619418,39.30157272193126 21.521534887564258,
                39.30081138378854 21.521384658920564,39.30018341259812 21.52126091678776,
                .
                .
                .
                39.30252626792019 21.520336381022798,39.30260775555942 21.521121659758194,
                39.302609617049306 21.521414015319046,39.302467085844434 21.521487158064158,
                39.30216403545025</coordinates>
            </LinearRing>
        </outerBoundaryIs>
    </Polygon>
</MultiGeometry>
</Placemark>
</Folder>
</kml>
```

Table 7.8: Wildlife refuges KML result

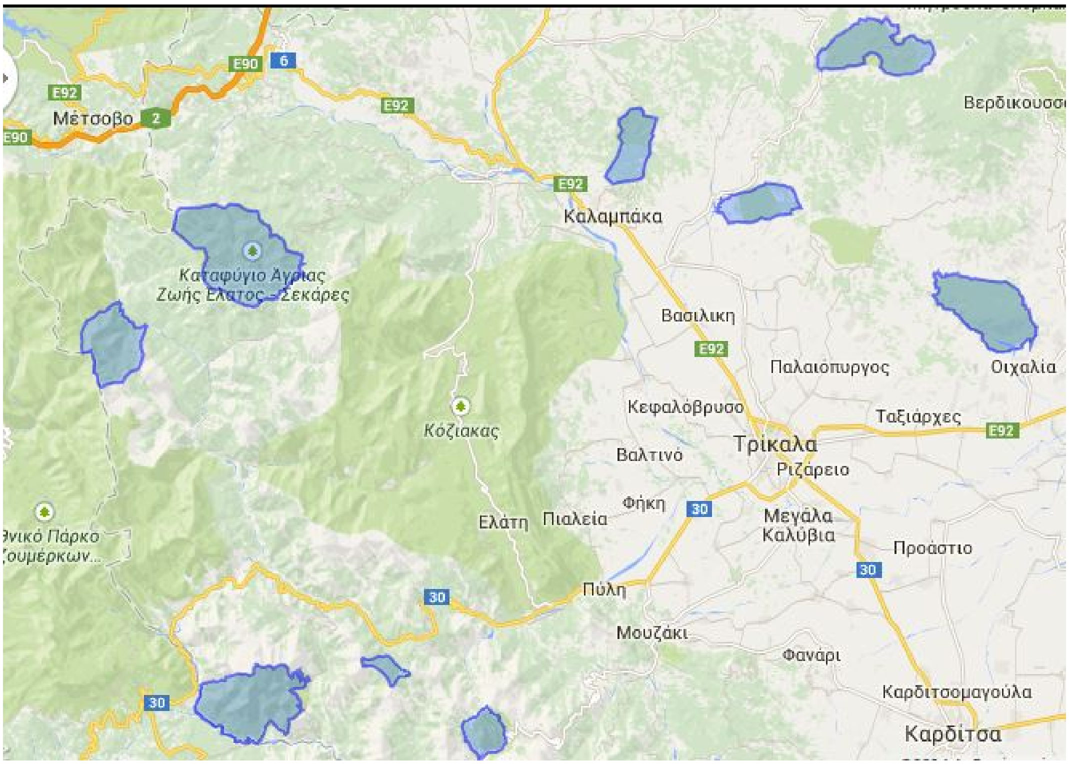

Figure 7.2: Wildlife refuges area

7.4 Blue Flags in Greece, 2010

The Blue Flag [2] is a certification by the Foundation for Environmental Education (FEE) that a beach or marina meets its stringent standards.

The Blue Flag is a trademark owned by FEE which is a not-for-profit, non-governmental organisation consisting of 65 organisations in 60 member countries in Europe, Africa, Oceania, Asia, North America and South America.

FEE's Blue Flag criteria include standards for water quality, safety, environmental education and information, the provision of services and general environmental management criteria. The Blue Flag is sought for beaches and marinas as an indication of their high environmental and quality standards.

An example of the dataset is shown here (Table 7.9) In this snippet we can see the name of the beach that was awarded the blue flag, and also the municipality, region and other informations about the beach.

[2] http://en.wikipedia.org/wiki/Blue_Flag_beach

```
<gml:featureMember>
 <ogr:blue_flags_2010_final fid="F0">
  <ogr:geometryProperty><gml:Point srsName="EPSG:2100">
  <gml:coordinates>494309.222795462002978,3930054.613042179960757</gml:coordinates>
  </gml:Point></ogr:geometryProperty>
  <ogr:c_edpp>94180401</ogr:c_edpp>
  <ogr:codeau>1</ogr:codeau>
  <ogr:commune>NEAS KYDONIAS</ogr:commune>
  <ogr:dd>ΚΡΗΤΗ-ΣΤΑΛΟΥ</ogr:dd>
  <ogr:descript>ΕΝΑΝΤΙ ΚΕΝΤΡΟΥ ΚΑΤΙ ΑΛΛΟ</ogr:descript>
  <ogr:dhmos>ΚΡΗΤΗ-Ν.ΚΥΔΩΝΙΑΣ</ogr:dhmos>
  <ogr:latitude>35.5166666666667</ogr:latitude>
  <ogr:longitude>23.9388888888889</ogr:longitude>
  <ogr:nomos>ΧΑΝΙΩΝ</ogr:nomos>
  <ogr:numind>GR4340180494180401</ogr:numind>
  <ogr:prelev>STALOS  - ENANTI KENTROU \'KATI ALLO\'</ogr:prelev>
  <ogr:province>Chania</ogr:province>
  <ogr:region>KRITI</ogr:region>
  <ogr:watername>ΣΤΑΛΟΣ</ogr:watername>
 </ogr:blue_flags_2010_final>
</gml:featureMember>
```

Table 7.9: Blue Flags GML Fraction

The table 7.10 demonstrates the result of the GML to RDF transformation using the GeoSPARQL vocabulary and the RDF/XML file format.

```
<rdf:Description rdf:about="http://strdf.di.uoa.gr/ontology#d21e3"
                 rdf:type="http://www.opengis.net/gml#blue_flags_2010_final">
    <ogr:geometryProperty rdf:resource="#d21e5"/>
    <ogr:c_edpp>94180401</ogr:c_edpp>
    <ogr:codeau>1</ogr:codeau>
    <ogr:commune>NEAS KYDONIAS</ogr:commune>
    <ogr:dd>ΚΡΗΤΗ-ΣΤΑΛΟΥ</ogr:dd>
    <ogr:descript>ΕΝΑΝΤΙ ΚΕΝΤΡΟΥ ΚΑΤΙ ΑΛΛΟ</ogr:descript>
    <ogr:dhmos>ΚΡΗΤΗ-Ν.ΚΥΔΩΝΙΑΣ</ogr:dhmos>
    <ogr:latitude>35.5166666666667</ogr:latitude>
    <ogr:longitude>23.9388888888889</ogr:longitude>
    <ogr:nomos>ΧΑΝΙΩΝ</ogr:nomos>
    <ogr:numind>GR4340180494180401</ogr:numind>
    <ogr:prelev>STALOS  - ENANTI KENTROU \'KATI ALLO\'</ogr:prelev>
    <ogr:province>Chania</ogr:province>
    <ogr:region>KRITI</ogr:region>
    <ogr:watername>ΣΤΑΛΟΣ</ogr:watername>
    <strdf:hasGeometry rdf:datatype="http://strdf.di.uoa.gr/ontology#WKT">
      <rdf:Description>
        <geo:asWKT rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral">
        &lt;http://www.opengis.net/def/crs/EPSG/0/2100&gt;
        POINT(494309.222795462002978 3930054.613042179960757)
        </geo:asWKT>
        <rdf:type rdf:resource="http://www.opengis.net/gml#Point"/>
      </rdf:Description>
    </strdf:hasGeometry>
  </rdf:Description>
```

Table 7.10: Blue Flags RDF Fraction

After inserting the output RDF in the Strabon system we can execute queries. You can see an example query and its output in KML format. (Table 7.11, 7.12 and Figure 7.3)

In this query we want to retrieve all the beaches that have been awarded a blue flag in the region of Rethymnon.

```
PREFIX strdf: <http://strdf.di.uoa.gr/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql>
PREFIX ogr: <http://ogr.maptools.org/>
SELECT ?result
WHERE {
        ?a  ogr:nomos "ΡΕΘΥΜΝΗΣ" .
        ?a strdf:hasGeometry ?f.
        ?f <http://www.opengis.net/ont/geosparql#asWKT> ?result
}
```

Table 7.11: Blue flags stSPARQL query

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://www.opengis.net/kml/2.2'>
<Folder>
<Placemark>
<name>Result6</name>
<Point>
   <coordinates>24.37499999519137,35.19166702734546</coordinates>
</Point>
</Placemark>
</Folder>
</kml>
```

Table 7.12: Blue flags KML result

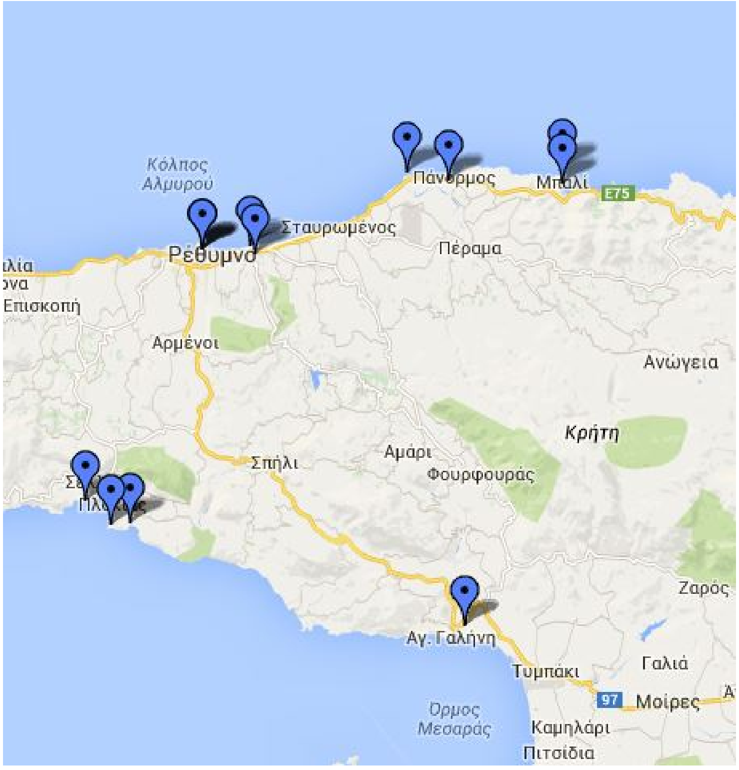

Figure 7.3: Blue flags of Rethymno

7.5 National Parks of Greece

A national park is a park in use for conservation purposes. [3] Often it is a reserve of natural, semi-natural, or developed land that a sovereign state declares or owns. Although individual nations designate their own national parks differently, there is a common idea: the conservation of wild nature for posterity and as a symbol of national pride.

An example of the dataset is shown here (Table 7.13)

[3] http://en.wikipedia.org/wiki/National_park

```
  <gml:featureMember>
  <ogr:ethikoi_drymoi fid="F3">
    <ogr:geometryProperty><gml:MultiPolygon srsName="EPSG:2100">
    <gml:polygonMember><gml:Polygon><gml:outerBoundaryIs>
    <gml:LinearRing><gml:coordinates>500866.614220653020311,
    4171297.4179223398678 501011.428300787985791,
    4171511.91110653989017 501283.111885040998459,
    .
    .
    .
    4171173.257922229822725 500866.614220653020311,4171297.4179223398678
    </gml:coordinates></gml:LinearRing>
    </gml:outerBoundaryIs></gml:Polygon>
    </gml:polygonMember></gml:MultiPolygon>
    </ogr:geometryProperty>
    <ogr:area>4909187.875</ogr:area>
    <ogr:area_strem>4909.19</ogr:area_strem>
    <ogr:category>ΠΥΡΗΝΑΣ</ogr:category>
    <ogr:perimeter>11599.418</ogr:perimeter>
    <ogr:textstring>ORIA_PYRHNA_SOYNIOY</ogr:textstring>
    <ogr:thesh>ΔΡΥΜΟΣ_ΣΟΥΝΙΟΥ</ogr:thesh>
  </ogr:ethikoi_drymoi>
</gml:featureMember>
```

Table 7.13: National Parks GML Fraction

The table 7.14 demonstrates the result of the GML to RDF transformation using the GeoSPARQL vocabulary and the RDF/XML file format.

```
<rdf:Description rdf:about="http://strdf.di.uoa.gr/ontology#d9e69"
            rdf:type="http://www.opengis.net/gml#ethikoi_drymoi">
  <ogr:geometryProperty rdf:resource="#d9e71"/>
  <ogr:area>4909187.875</ogr:area>
  <ogr:area_strem>4909.19</ogr:area_strem>
  <ogr:category>ΠΥΡΗΝΑΣ</ogr:category>
  <ogr:perimeter>11599.418</ogr:perimeter>
  <ogr:textstring>ORIA_PYRHNA_SOYNIOY</ogr:textstring>
  <ogr:thesh>ΔΡΥΜΟΣ_ΣΟΥΝΙΟΥ</ogr:thesh>
  <strdf:hasGeometry rdf:datatype="http://strdf.di.uoa.gr/ontology#WKT">
    <rdf:Description>
      <geo:asWKT rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral">
      &lt;http://www.opengis.net/def/crs/EPSG/0/2100&gt;
      MULTIPOLYGON(((500866.614220653020311 4171297.4179223398678,
      .
      .
      .
      500866.614220653020311 4171297.4179223398678)))
      </geo:asWKT>
      <rdf:type rdf:resource="http://www.opengis.net/gml#MultiPolygon"/>
    </rdf:Description>
  </strdf:hasGeometry>
</rdf:Description>
```

Table 7.14: National Parks RDF Fraction

After inserting the output RDF in the Strabon system we can execute queries. You can see an example query and its output in KML format. (Table 7.15, 7.16 and Figure 7.4) In this query we want to retrieve the geometry of Parnitha's national park.

```
PREFIX strdf: <http://strdf.di.uoa.gr/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql>
PREFIX ogr: <http://ogr.maptools.org/>

SELECT ?results
WHERE {
        ?a ogr:thesh "ΔΡΥΜΟΣ_ΠΑΡΝΗΘΑΣ" .
        ?a strdf:hasGeometry ?f.
        ?f <http://www.opengis.net/ont/geosparql#asWKT> ?results
}
```

Table 7.15: National Parks stSPARQL query

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://www.opengis.net/kml/2.2'>
<Folder>
<Placemark>
<name>Result0</name>
<MultiGeometry>
    <Polygon>
        <outerBoundaryIs>
            <LinearRing>
                <coordinates>
                  23.65651404234795,38.12273668916609 23.655833157471974,
                  .
                  .
                  .
                38.121875723226644 23.65651404234795,38.12273668916609</coordinates>
            </LinearRing>
        </outerBoundaryIs>
    </Polygon>
</MultiGeometry>
</Placemark>
</Folder>
</kml>
```

Table 7.16: National parks refuges KML result

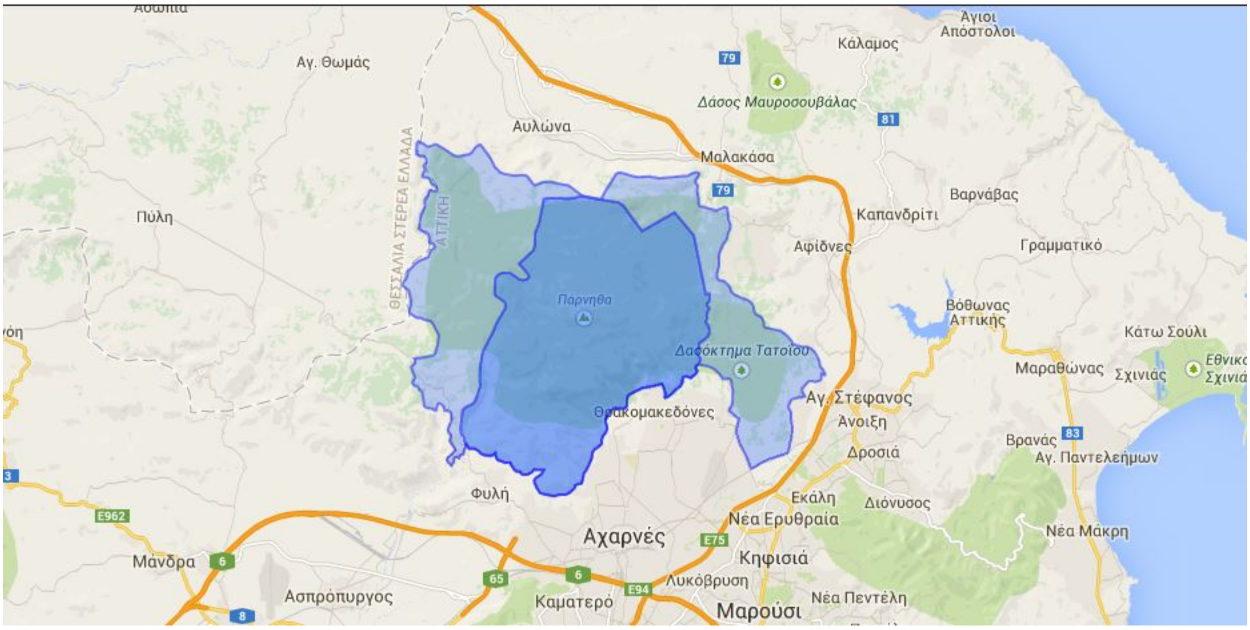

Figure 7.4: National Park of Parnitha

7.6 Controlled Hunting Areas in Greece

This dataset consists of the controlled hunting areas in Greece, the data that accompanies them include the name of the area, the law in which this area is declared as as a hunting are, the region and also the game that can be hunted in it. An example of the dataset is shown here (Table 7.17)

```
<gml:featureMember>
  <ogr:elegxomenes_kynhgetikes_perioxes fid="F0">
    <ogr:geometryProperty><gml:MultiPolygon srsName="EPSG:2100">
    <gml:polygonMember><gml:Polygon><gml:outerBoundaryIs>
    <gml:LinearRing><gml:coordinates>482871.277100000006612,
    4331947.822999999858439 482968.395000000018626,
    .
    .
    .
    4331952.043700000271201 482871.277100000006612,
    4331947.822999999858439</gml:coordinates>
    </gml:LinearRing></gml:outerBoundaryIs>
    </gml:Polygon></gml:polygonMember>
    </gml:MultiPolygon></ogr:geometryProperty>
    <ogr:apofash>163921/2035/12-7-1979</ogr:apofash>
    <ogr:area>420447.481</ogr:area>
    <ogr:area_strem>420.45</ogr:area_strem>
    <ogr:ektash>250</ogr:ektash>
    <ogr:epopteuous>ΔΑΣΑΡΧΕΙΟ ΣΚΟΠΕΛΟΥ</ogr:epopteuous>
    <ogr:fek>744/B/4-9-1979</ogr:fek>
    <ogr:nomos>ΜΑΓΝΗΣΙΑΣ</ogr:nomos>
    <ogr:panida_thi>ΑΙΓΑΓΡΟΣ</ogr:panida_thi>
    <ogr:perifereia>ΘΕΣΣΑΛΙΑ</ogr:perifereia>
    <ogr:perimeter>2875.821</ogr:perimeter>
    <ogr:thesh>ΝΗΣΟΣ ΑΓ. ΓΕΩΡΓΙΟΣ</ogr:thesh>
  </ogr:elegxomenes_kynhgetikes_perioxes>
 </gml:featureMember>
```


Table 7.17: Hunting areas GML Fraction

The table 7.18 demonstrates the result of the GML to RDF transformation using the GeoSPARQL vocabulary and the RDF/XML file format.

```
<rdf:Description rdf:about="http://strdf.di.uoa.gr/ontology#d3e3"
   rdf:type="http://www.opengis.net/gml#elegxomenes_kynhgetikes_perioxes">
    <ogr:geometryProperty rdf:resource="#d3e5"/>
    <ogr:apofash>163921/2035/12-7-1979</ogr:apofash>
    <ogr:area>420447.481</ogr:area>
    <ogr:area_strem>420.45</ogr:area_strem>
    <ogr:ektash>250</ogr:ektash>
    <ogr:epopteuous>ΔΑΣΑΡΧΕΙΟ ΣΚΟΠΕΛΟΥ</ogr:epopteuous>
    <ogr:fek>744/B/4-9-1979</ogr:fek>
    <ogr:nomos>ΜΑΓΝΗΣΙΑΣ</ogr:nomos>
    <ogr:panida_thi>ΑΙΓΑΓΡΟΣ</ogr:panida_thi>
    <ogr:perifereia>ΘΕΣΣΑΛΙΑ</ogr:perifereia>
    <ogr:perimeter>2875.821</ogr:perimeter>
    <ogr:thesh>ΝΗΣΟΣ ΑΓ. ΓΕΩΡΓΙΟΣ</ogr:thesh>
    <strdf:hasGeometry rdf:datatype="http://strdf.di.uoa.gr/ontology#WKT">
      <rdf:Description>
        <geo:asWKT rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral">
        &lt;http://www.opengis.net/def/crs/EPSG/0/2100&gt;
        MULTIPOLYGON(((482871.277100000006612 4331947.822999999858439,
        .
        .
        .
        482871.277100000006612 4331947.822999999858439))) </geo:asWKT>
        <rdf:type rdf:resource="http://www.opengis.net/gml#MultiPolygon"/>
      </rdf:Description>
    </strdf:hasGeometry>
</rdf:Description>
```

Table 7.18: Hunting areas RDF Fraction

After inserting the output RDF in the Strabon system we can execute queries. You can see an example query and its output in KML format. (Table 7.19, 7.20 and Figure 7.5) In this query we search for areas in Macedonia periphery in which wild boar can be hunted.

```
PREFIX strdf: <http://strdf.di.uoa.gr/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql>
PREFIX ogr: <http://ogr.maptools.org/>

SELECT ?results
WHERE {
        ?a  ogr:panida_thi "ΑΓΡΙΟΧΟΙΡΟΣ" .
        ?a  ogr:perifereia "ΜΑΚΕΔΟΝΙΑ" .
        ?a strdf:hasGeometry ?f.
        ?f <http://www.opengis.net/ont/geosparql#asWKT> ?results
}
```

Table 7.19: Hunting areas stSPARQL query

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://www.opengis.net/kml/2.2'>
<Folder>
<Placemark>
<name>Result0</name>
<MultiGeometry>
    <Polygon>
        <outerBoundaryIs>
            <LinearRing>
                <coordinates>21.775989702493597,
                40.88254867865887 21.777865451161823,
                .
                .
                .
                40.88084989196388 21.775989702493597,
                40.88254867865887</coordinates>
            </LinearRing>
        </outerBoundaryIs>
    </Polygon>
</MultiGeometry>
</Placemark>
</Folder>
```

Table 7.20: Hunting areas KML result

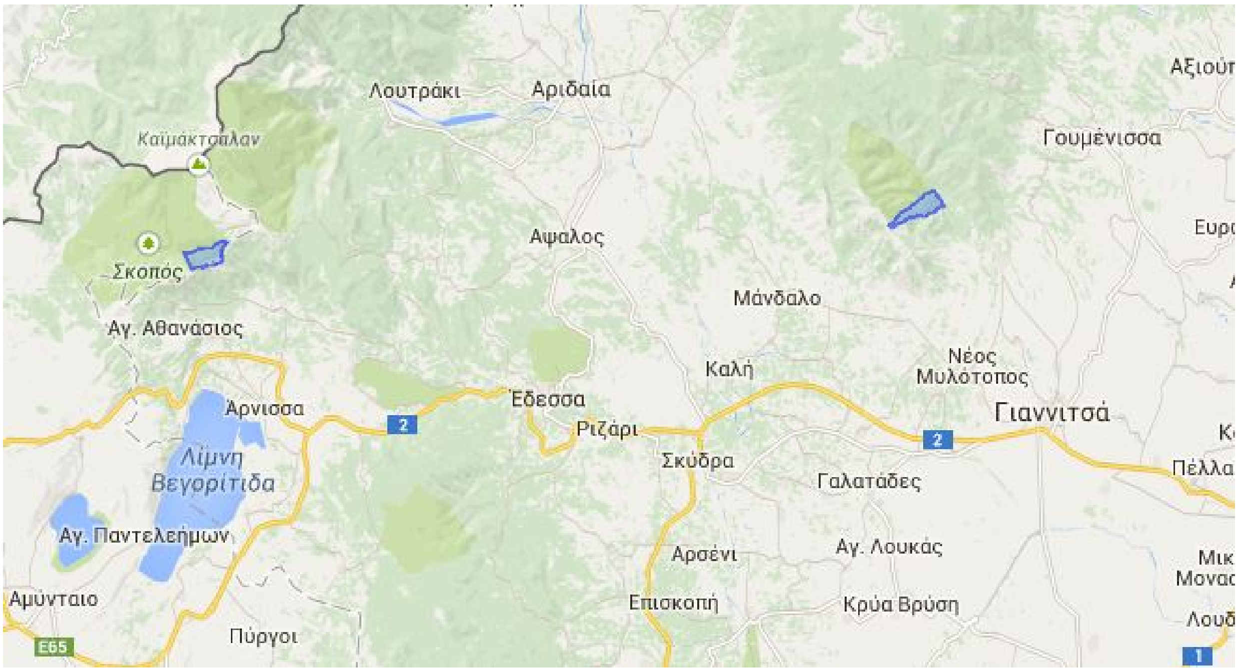

Figure 7.5: Wildboard Hunting areas in Macedonia

7.7 Conclusion

Transforming geospatial data in GML format using this tool is simple, due to the user-friendly interface and produces RDF files with all the useful information available in the original files. Enabling the creation of complex queries using the data for data scientist to use in their research.

Chapter 8

Conclusion Future Work

This tool was built with the intention to be used by inexperienced users, providing usability and simplicity. The tool can be extended to include more geospatial data file-types like Shapefiles, an extension like this can allow an even larger amount of geospatial data to be used by data scientist . Furthermore a Java library or a programming API can be developed based on this tool to allow easier reuse in other applications.

The development of this tool covered the main goals that were set at the beginning of this master's thesis, as a simple web based tool it can help data scientist transform geodata and enable them to create knowledge out of them.

Appendix

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:imro="http://www.geonovum.nl/imro/2008/1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:math="http://www.w3.org/2005/xpath-functions/math"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:strdf="http://strdf.di.uoa.gr/ontology#"
xmlns:geo="http://www.opengis.net/ont/geosparql#"
xmlns:ogr="http://ogr.maptools.org/"
exclude-result-prefixes="xs" version="2.0">
<xsl:output indent="yes"/>
<xsl:strip-space elements="*"/>

<xsl:template match="@*|node()">
        <xsl:copy>
                <xsl:apply-templates select="@*|node()" />
        </xsl:copy>
</xsl:template>
        <xsl:template match="gml:boundedBy"></xsl:template>
</xsl:stylesheet>
```

Table A.1: Preprocessing XSLT file

```
<?xml version="1.0" encoding="UTF-8"?><xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:imro="http://www.geonovum.nl/imro/2008/1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:math="http://www.w3.org/2005/xpath-functions/math"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:strdf="http://strdf.di.uoa.gr/ontology#"
xmlns:geo="http://www.opengis.net/ont/geosparql#"
xmlns:ogr="http://ogr.maptools.org/"
exclude-result-prefixes="xs"
version="2.0">
<xsl:output indent="yes"/>
<xsl:strip-space elements="*"/>
<xsl:template match="/ "><rdf:RDF>
<xsl:apply-templates select="//*[count(ancestor::*) mod 2 = 0] "/>
</rdf:RDF></xsl:template>
<xsl:template match="*[count(ancestor::*) mod 2 = 0 ]  ">
<rdf:Description rdf:about="http://strdf.di.uoa.gr/ontology#{if (@gml:id)
then @gml:id else generate-id(.)}"  rdf:type="http://www.opengis.net/gml#{local-name()}">
<xsl:apply-templates/>
<xsl:apply-templates select="@srsName"/>
<xsl:if test="ogr:geometryProperty">
<strdf:hasGeometry rdf:datatype="http://strdf.di.uoa.gr/ontology#WKT" >
<xsl:copy-of select="child::*/child::*"/>
</strdf:hasGeometry></xsl:if>
</rdf:Description>
</xsl:template>
<xsl:template match="@srsName">
<gml:srsName rdf:resource="http://www.strabon.di.uoa.gr#{.}"/>
</xsl:template>
<xsl:template match="gml:description">
<rdfs:comment><xsl:value-of select="text()"/></rdfs:comment>
</xsl:template>
<xsl:template match=" *[count(ancestor::*) mod 2 != 0  and not(child::*)  ]">
<xsl:element name="{name()}">
<xsl:value-of select="text()"/>
</xsl:element>
</xsl:template>
<xsl:template match="  *[count(ancestor::*) mod 2 != 0   and child::*  ]">
<xsl:for-each select="*">
<xsl:element name="{parent::*/name()}">
<xsl:attribute name="rdf:resource" select="concat('#', if (@gml:id) then
@gml:id else generate-id(.))"></xsl:attribute>
</xsl:element></xsl:for-each></xsl:template>
<xsl:template match="*[count(ancestor::*) mod 2 != 0 and normalize-space(@xlink:href)  ]">
<xsl:element name="{name()}">
<xsl:attribute name="rdf:resource" select="@xlink:href"/>
</xsl:element></xsl:template>
<xsl:template match="//*[(ancestor::ogr:geometryProperty)]">
</xsl:template></xsl:stylesheet>
```

Table A.2: GML to RDF XSLT file

```
<?xml version="1.0" encoding="UTF-8"?><xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:imro="http://www.geonovum.nl/imro/2008/1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:math="http://www.w3.org/2005/xpath-functions/math"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:strdf="http://strdf.di.uoa.gr/ontology#"
xmlns:geo="http://www.opengis.net/ont/geosparql#"
xmlns:ogr="http://ogr.maptools.org/"
exclude-result-prefixes='xs'
version='2.0'>
<xsl:output indent='yes'/>
<xsl:strip-space elements='*'/><xsl:variable name='epsg'  >&lt;http://www.opengis.net/def/crs/EPSG/0/2100&gt; </xsl:variable>
<xsl:template match="@*|node()">
<xsl:copy>
<xsl:apply-templates select="@*|node()" />
</xsl:copy>
</xsl:template>
<xsl:template match="gml:Box">
<xsl:param name="separator" select="' '" />
<xsl:call-template name="gml:srsName"/>
<xsl:variable name="coords" select="tokenize(replace(replace(gml:coordinates,'\s+$',''),'^\s+',''), '[,\s\n\r]+')" />
<xsl:variable name="MINX"><xsl:value-of select="$coords[1]"/></xsl:variable>
<xsl:variable name="MINY"><xsl:value-of select="$coords[2]"/></xsl:variable>
<xsl:variable name="MAXX"><xsl:value-of select="$coords[3]"/></xsl:variable>
<xsl:variable name="MAXY"><xsl:value-of select="$coords[4]"/></xsl:variable>
<rdf:Description><geo:asWKT rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" >
<xsl:value-of select="$epsg" />
```

```
<xsl:value-of select="concat(' POLYGON((', $MINX, ' ', $MINY, ', '
, $MAXX, ' ', $MINY, ', ', $MAXX, ' ', $MAXY, ', ', $MINX, ' ',
 $MAXY, ', ', $MINX, ' ', $MINY, ')) ^^geo:wktLiteral')"/>
</geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#Polygon"/>
</rdf:Description>        </xsl:template>
<xsl:template match="gml:Envelope">
<xsl:param name="separator" select="' '"/>
<xsl:call-template name="gml:srsName"/>
<xsl:variable name="MINX"><xsl:value-of select="normalize-space(substring-before(gml:lowerCorner, $separator))"/></xsl:variable>
<xsl:variable name="MINY"><xsl:value-of select="normalize-space(substring-after(gml:lowerCorner, $separator))"/></xsl:variable>
<xsl:variable name="MAXX"><xsl:value-of select="normalize-space(substring-before(gml:upperCorner, $separator))"/></xsl:variable>
<xsl:variable name="MAXY"><xsl:value-of select="normalize-space(substring-after(gml:upperCorner, $separator))"/></xsl:variable>
<rdf:Description><geo:asWKT    rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" >
<xsl:value-of select="$epsg" />
<xsl:value-of select="concat(' POLYGON((', $MINX, ' ', $MINY, ', ', $MAXX, ' ',
 $MINY, ', ', $MAXX, ' ', $MAXY, ', ', $MINX, ' ', $MAXY, ', ', $MINX, ' ',
  $MINY, ')) ^^geo:wktLiteral')"/>
</geo:asWKT><rdf:type rdf:resource="http://www.opengis.net/gml#Polygon"/>
</rdf:Description>
</xsl:template>
<xsl:template match="gml:Point">
<xsl:call-template name="gml:srsName"/>
<rdf:Description><geo:asWKT    rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" ><xsl:value-of select="$epsg" />
<xsl:text> POINT</xsl:text>
<xsl:call-template name="gml:posList"/>
</geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#Point"/>
</rdf:Description>
</xsl:template>
<xsl:template match="gml:LineString">
<xsl:call-template name="gml:srsName"/>
<rdf:Description><geo:asWKT    rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" ><xsl:value-of select="$epsg" />
<xsl:text> LINESTRING</xsl:text>
<xsl:call-template name="gml:posList"/>
</geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#LineString"/>
</rdf:Description>        </xsl:template>
<xsl:template match="gml:Polygon">
<xsl:call-template name="gml:srsName"/>
<rdf:Description><geo:asWKT    rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" ><xsl:value-of select="$epsg" />
<xsl:text> POLYGON(</xsl:text>
<xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
<xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
<xsl:text>) </xsl:text></geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#Polygon"/>
</rdf:Description>        </xsl:template>
<xsl:template match="gml:MultiPoint">
<xsl:call-template name="gml:srsName"/>
<rdf:Description><geo:asWKT    rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" ><xsl:value-of select="$epsg" />
<xsl:text> MULTIPOINT(</xsl:text>
<xsl:for-each select="gml:pointMember//gml:Point|gml:pointMembers//gml:Point">
<xsl:if test="not(position()=1)">
<xsl:text>,</xsl:text>
</xsl:if>
<xsl:apply-templates />
</xsl:for-each>
<xsl:text>) </xsl:text></geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#MultiPoint"/>
</rdf:Description>        </xsl:template>
<xsl:template match="gml:MultiLineString">
<xsl:call-template name="gml:srsName"/>
<rdf:Description><geo:asWKT    rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" ><xsl:value-of select="$epsg" />
<xsl:text> MULTILINESTRING(</xsl:text>
<xsl:for-each select="gml:lineStringMember">
<xsl:call-template name="gml:lineStringMember"/>
<xsl:if test="(position()!=last())">
<xsl:text>,</xsl:text>
</xsl:if>
</xsl:for-each>
<xsl:text>) </xsl:text></geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#MultiLineString"/>
</rdf:Description>        </xsl:template>
<xsl:template match="gml:Curve">
<xsl:call-template name="gml:srsName"/><rdf:Description> <geo:asWKT    rdf:datatype =
 "http://www.opengis.net/ont/geosparql#wktLiteral">
<xsl:value-of select="$epsg" />
<xsl:for-each select="gml:segments//gml:LineStringSegment">
<xsl:choose>
<xsl:when test="not(position()=1)">
<xsl:text>,</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:text> MULTILINESTRING(</xsl:text>
</xsl:otherwise>
</xsl:choose>
<xsl:apply-templates />
</xsl:for-each>
<xsl:text>) </xsl:text></geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#MultiLineString"/>
</rdf:Description>        </xsl:template>
<xsl:template match="gml:MultiPolygon">
<xsl:call-template name="gml:srsName"/>
<rdf:Description>
<geo:asWKT rdf:datatype        =
"http://www.opengis.net/ont/geosparql#wktLiteral" >
```

```
<xsl:value-of select="$epsg" />
<xsl:for-each select="gml:polygonMember//gml:Polygon">
<xsl:choose>
<xsl:when test="not(position()=1)">
<xsl:text>,</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:call-template name="gml:srsName"/>
<xsl:text> MULTIPOLYGON(</xsl:text>
</xsl:otherwise>
</xsl:choose>
<xsl:text>(</xsl:text>
<xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
<xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
<xsl:text>)</xsl:text>
</xsl:for-each>
<xsl:text>) </xsl:text></geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#MultiPolygon"/>
</rdf:Description>     </xsl:template>
<xsl:template match="gml:MultiSurface">
<rdf:Description><geo:asWKT   rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" ><xsl:value-of select="$epsg" />
<xsl:for-each select="gml:surfaceMember//gml:Surface|gml:surfaceMember//gml:Polygon">
<xsl:choose>     <xsl:when test="not(position()=1)">
<xsl:text>,</xsl:text>     </xsl:when>
<xsl:otherwise>
<xsl:call-template name="gml:srsName"/>
<xsl:text> MULTIPOLYGON(</xsl:text>
</xsl:otherwise>     </xsl:choose>     <xsl:text>(</xsl:text>
<xsl:apply-templates />     <xsl:text>)</xsl:text>
</xsl:for-each>     <xsl:text>) </xsl:text></geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#MultiPolygon"/>
</rdf:Description>     </xsl:template>
<xsl:template match="gml:MultiGeometry|gml:GeometryCollection">
<xsl:call-template name="gml:srsName"/>
<rdf:Description><geo:asWKT   rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral" ><xsl:value-of select="$epsg" />
<xsl:text> GEOMETRYCOLLECTION(</xsl:text>
<xsl:for-each select="gml:geometryMember">
<xsl:if test="not(position()=1)">
<xsl:text>,</xsl:text>     </xsl:if>
<xsl:apply-templates select="*[not(local-name()='gml:srsName')]" />
</xsl:for-each>     <xsl:text>) </xsl:text></geo:asWKT>
<rdf:type rdf:resource="http://www.opengis.net/gml#GeometryCollection"/>
</rdf:Description>     </xsl:template>
<xsl:template match="gml:pointMember">
<xsl:apply-templates select="gml:Point"/>
</xsl:template>
<xsl:template name="gml:lineStringMember">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template match="gml:LineStringSegment">
<xsl:apply-templates select="gml:posList"/>
</xsl:template>
<xsl:template match="gml:polygonMember|gml:patches//gml:PolygonPatch">
<xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
<xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
</xsl:template>
<xsl:template match="gml:exterior|gml:outerBoundaryIs">
<xsl:apply-templates select="gml:LinearRing"/>
</xsl:template>
<xsl:template match="gml:interior|gml:innerBoundaryIs">
<xsl:text>,</xsl:text>
<xsl:apply-templates select="gml:LinearRing"/>
</xsl:template>
<xsl:template match="gml:LinearRing">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template match="gml:pos">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template match="gml:coordinates">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template name="gml:posList">
<xsl:call-template name="gml:processPosList"/>
</xsl:template>
<xsl:template match="gml:posList">
<xsl:call-template name="gml:processPosList"/>
</xsl:template>
<xsl:template name="gml:processPosList">
<xsl:text>(</xsl:text>
<xsl:for-each select="tokenize(replace(replace(.,'\s+$',''),'^\s+',''), '[,\s\n\r]+')">
<xsl:variable name="t" select="." />
<xsl:value-of select="concat($t, if (position() = last()) then '' else if ((position() mod 2) = 0) then ', ' else ' ')" />
</xsl:for-each>
<xsl:text>)</xsl:text>     </xsl:template>
<xsl:template name="gml:srsName">
</xsl:template>     </xsl:stylesheet>
```

Table A.3: GeoSPARQL vocabulary XSLT file

```
<?xml version=”1.0” encoding=”UTF-8”?><xsl:stylesheet
xmlns:xsl=”http://www.w3.org/1999/XSL/Transform”
xmlns:xs=”http://www.w3.org/2001/XMLSchema”
xmlns:imro=”http://www.geonovum.nl/imro/2008/1”
xmlns:xlink=”http://www.w3.org/1999/xlink”
xmlns:gml=”http://www.opengis.net/gml”
xmlns:math=”http://www.w3.org/2005/xpath-functions/math”
xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
xmlns:dc=”http://purl.org/dc/elements/1.1/”
xmlns:owl=”http://www.w3.org/2002/07/owl#”
xmlns:rdfs=”http://www.w3.org/2000/01/rdf-schema#”
xmlns:strdf=”http://strdf.di.uoa.gr/ontology#”
xmlns:geo=”http://www.opengis.net/ont/geosparql#”
xmlns:ogr=”http://ogr.maptools.org/”
exclude-result-prefixes=’xs’
version=’2.0’>
<xsl:output indent=’yes’/>
<xsl:strip-space elements=’*’/><xsl:variable name=’epsg’  >;http://www.opengis.net/def/crs/EPSG/0/2100</xsl:variable>
<xsl:template match=”@*|node()”>
<xsl:copy>
<xsl:apply-templates select=”@*|node()” />
</xsl:copy>
</xsl:template>
<xsl:template match=”gml:Box”>
<xsl:param name=”separator” select=”’ ’” />
<xsl:call-template name=”gml:srsName”/>
<xsl:variable name=”coords” select=”tokenize(replace(replace(gml:coordinates,’\s+$’,’’),’^\s+’,’’), ’[,\s\n\r]+’)” />
<xsl:variable name=”MINX”><xsl:value-of select=”$coords[1]”/></xsl:variable>
<xsl:variable name=”MINY”><xsl:value-of select=”$coords[2]”/></xsl:variable>
<xsl:variable name=”MAXX”><xsl:value-of select=”$coords[3]”/></xsl:variable>
<xsl:variable name=”MAXY”><xsl:value-of select=”$coords[4]”/></xsl:variable>
<xsl:value-of select=”concat(’ POLYGON((’, $MINX, ’ ’, $MINY, ’, ’, $MAXX, ’ ’,
 $MINY, ’, ’, $MAXX, ’ ’, $MAXY, ’, ’, $MINX, ’ ’, $MAXY, ’, ’, $MINX, ’ ’,
  $MINY, ’))’)”/>
<xsl:value-of select=”$epsg” />
</xsl:template>
<xsl:template match=”gml:Envelope”>
<xsl:param name=”separator” select=”’ ’”/>
<xsl:call-template name=”gml:srsName”/>
<xsl:variable name=”MINX”><xsl:value-of select=”normalize-space(substring-before(gml:lowerCorner, $separator))”/></xsl:variable>
<xsl:variable name=”MINY”><xsl:value-of select=”normalize-space(substring-after(gml:lowerCorner, $separator))”/></xsl:variable>
<xsl:variable name=”MAXX”><xsl:value-of select=”normalize-space(substring-before(gml:upperCorner, $separator))”/></xsl:variable>
<xsl:variable name=”MAXY”><xsl:value-of select=”normalize-space(substring-after(gml:upperCorner, $separator))”/></xsl:variable>
<xsl:value-of select=”concat(’ POLYGON((’, $MINX, ’ ’, $MINY, ’, ’, $MAXX,
’ ’, $MINY, ’, ’, $MAXX, ’ ’, $MAXY, ’, ’, $MINX, ’ ’, $MAXY, ’, ’, $MINX,
’ ’, $MINY, ’))’)”/>
<xsl:value-of select=”$epsg” />
</xsl:template>
<xsl:template match=”gml:Point”>
<xsl:call-template name=”gml:srsName”/>
<xsl:text> POINT</xsl:text>
<xsl:call-template name=”gml:posList”/>
<xsl:value-of select=”$epsg” />
</xsl:template>
<xsl:template match=”gml:LineString”>
<xsl:call-template name=”gml:srsName”/>
<xsl:text> LINESTRING</xsl:text>
<xsl:call-template name=”gml:posList”/>
<xsl:value-of select=”$epsg” />
</xsl:template>
<xsl:template match=”gml:Polygon”>
<xsl:call-template name=”gml:srsName”/>
<xsl:text> POLYGON(</xsl:text>
<xsl:apply-templates select=”gml:exterior|gml:outerBoundaryIs”/>
<xsl:apply-templates select=”gml:interior|gml:innerBoundaryIs”/>
<xsl:text>)</xsl:text>
<xsl:value-of select=”$epsg” />
</xsl:template>
<xsl:template match=”gml:MultiPoint”>
<xsl:call-template name=”gml:srsName”/>
<xsl:text> MULTIPOINT(</xsl:text>
<xsl:for-each select=”gml:pointMember//gml:Point|gml:pointMembers//gml:Point”>
<xsl:if test=”not(position()=1)”>
<xsl:text>,</xsl:text>
</xsl:if>
<xsl:apply-templates />
</xsl:for-each>
<xsl:text>)</xsl:text>
<xsl:value-of select=”$epsg” />
</xsl:template>
<xsl:template match=”gml:MultiLineString”>
<xsl:call-template name=”gml:srsName”/>
<xsl:text> MULTILINESTRING(</xsl:text>
<xsl:for-each select=”gml:lineStringMember”>
<xsl:call-template name=”gml:lineStringMember”/>
<xsl:if test=”(position()!=last())”>
<xsl:text>,</xsl:text>
</xsl:if>
</xsl:for-each>
<xsl:text>)</xsl:text><xsl:value-of select=”$epsg” />
</xsl:template>
<xsl:template match=”gml:Curve”>
<xsl:call-template name=”gml:srsName”/>
<xsl:for-each select=”gml:segments//gml:LineStringSegment”>
<xsl:choose>
```

```
<xsl:when test="not(position()=1)">
<xsl:text>,</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:text> MULTILINESTRING(</xsl:text>
</xsl:otherwise>
</xsl:choose>
<xsl:apply-templates />
</xsl:for-each>
<xsl:text>) </xsl:text>
<xsl:value-of select="$epsg" />
</xsl:template>
<xsl:template match="gml:MultiPolygon">
<xsl:call-template name="gml:srsName"/>
<xsl:for-each select="gml:polygonMember//gml:Polygon">
<xsl:choose>
<xsl:when test="not(position()=1)">
<xsl:text>,</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:call-template name="gml:srsName"/>
<xsl:text> MULTIPOLYGON(</xsl:text>
</xsl:otherwise>
</xsl:choose>
<xsl:text>(</xsl:text>
<xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
<xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
<xsl:text>)</xsl:text>
</xsl:for-each>
<xsl:text>)  </xsl:text>
<xsl:value-of select="$epsg" />
</xsl:template>
<xsl:template match="gml:MultiSurface">
<xsl:for-each select="gml:surfaceMember//gml:Surface|gml:surfaceMember//gml:Polygon">
<xsl:choose>
<xsl:when test="not(position()=1)">
<xsl:text>,</xsl:text>
</xsl:when>
<xsl:otherwise>
<xsl:call-template name="gml:srsName"/>
<xsl:text> MULTIPOLYGON(</xsl:text>
</xsl:otherwise>          </xsl:choose>
<xsl:text>(</xsl:text>          <xsl:apply-templates />
<xsl:text>)</xsl:text>          </xsl:for-each>
<xsl:text>)  </xsl:text>  <xsl:value-of select="$epsg" />
</xsl:template>
<xsl:template match="gml:MultiGeometry|gml:GeometryCollection">
<xsl:call-template name="gml:srsName"/>
<xsl:text> GEOMETRYCOLLECTION(</xsl:text>
<xsl:for-each select="gml:geometryMember">
<xsl:if test="not(position()=1)">
<xsl:text>,</xsl:text>          </xsl:if>
<xsl:apply-templates select="*[not(local-name()='gml:srsName')]" />
</xsl:for-each>
<xsl:text>)</xsl:text><xsl:value-of select="$epsg" />
</xsl:template>          <xsl:template match="gml:pointMember">
<xsl:apply-templates select="gml:Point"/>
</xsl:template>
<xsl:template name="gml:lineStringMember">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template match="gml:LineStringSegment">
<xsl:apply-templates select="gml:posList"/>
</xsl:template>
<xsl:template match="gml:polygonMember|gml:patches//gml:PolygonPatch">
<xsl:apply-templates select="gml:exterior|gml:outerBoundaryIs"/>
<xsl:apply-templates select="gml:interior|gml:innerBoundaryIs"/>
</xsl:template>
<xsl:template match="gml:exterior|gml:outerBoundaryIs">
<xsl:apply-templates select="gml:LinearRing"/>
</xsl:template>
<xsl:template match="gml:interior|gml:innerBoundaryIs">
<xsl:text>,</xsl:text>
<xsl:apply-templates select="gml:LinearRing"/>
</xsl:template>
<xsl:template match="gml:LinearRing">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template match="gml:pos">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template match="gml:coordinates">
<xsl:call-template name="gml:posList"/>
</xsl:template>
<xsl:template name="gml:posList">
<xsl:call-template name="gml:processPosList"/>
</xsl:template>
<xsl:template match="gml:posList">
<xsl:call-template name="gml:processPosList"/>
</xsl:template>
<xsl:template name="gml:processPosList">
<xsl:text>(</xsl:text>
<xsl:for-each select="tokenize(replace(replace(.,'\s+$',''),'^\s+',''), '[,\s\n\r]+')">
<xsl:variable name="t" select="." />
<xsl:value-of select="concat($t, if (position() = last()) then '' else if ((position() mod 2) = 0) then ', ' else ' ')" />
</xsl:for-each>
```

```
<xsl:text>)</xsl:text>
</xsl:template>
<xsl:template name="gml:srsName">
</xsl:template></xsl:stylesheet>
```

Table A.4: stRDF vocabulary XSLT file

Bibliography

[1] OpenGIS implementation specification for geographic information - simple feature access - part 1: Common architecture, October 2006.

[2] Konstantina Bereta, Panayiotis Smeros, and Manolis Koubarakis. Representation and querying of valid time of triples in linked geospatial data. In Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *The Semantic Web: Semantics and Big Data*, volume 7882 of *Lecture Notes in Computer Science*, pages 259–274. Springer Berlin Heidelberg, 2013.

[3] 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.

[4] Dan Brickley and Ramanathan V Guha. Resource description framework (rdf) schema specification 1.0: W3c candidate recommendation 27 march 2000. 2000.

[5] Jeen Broekstra, Arjohn Kampman, and Frank Van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *The Semantic Web—ISWC 2002*, pages 54–68. Springer, 2002.

[6] Anthony G Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, 1(3):275–316, 1997.

[7] Simon Cox, Paul Daisey, Ron Lake, and Clemens Portele. OpenGIS geography markup language (GML) encoding specification. OpenGIS Encoding Specification OGC 02-023r4, Open Geospatial Consortium, Inc., December 2003.

[8] Max J Egenhofer. A formal definition of binary topological relationships. In *Foundations of data organization and algorithms*, pages 457–472. Springer, 1989.

[9] URL ESRI and White PaperdJuly. Esri shapefile technical description, 1998.

[10] Fayçal Hamdi, Nathalie Abadie, Bénédicte Bucher, and Abdelfettah Feliachi. Geomrdf: A geodata converter with a fine-grained structured representation of geometry in the web. 2014.

[11] Zoi Kaoudi. *Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks*. PhD thesis, PhD thesis, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece, 2011.

[12] Manolis Koubarakis, Manos Karpathiotakis, Kostis Kyzirakos, Charalampos Nikolaou, and Michael Sioutis. *Data Models and Query Languages for Linked Geospatial Data*. Springer, 2012.

[13] Manolis Koubarakis, Kostis Kyzirakos, Babis Nikolaou, Michael Sioutis, and Stavros Vassos. A data model and query language for an extension of rdf with time and space, 2011.

[14] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: a semantic geospatial dbms. In *The Semantic Web–ISWC 2012*, pages 295–311. Springer, 2012.

[15] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: A semantic geospatial dbms. In *International Semantic Web Conference (1)*, pages 295–311, 2012.

[16] Kostis Kyzirakos, Ioannis Vlachopoulos, Dimitrianos Savva, Stefan Manegold, and Manolis Koubarakis. Geotriples: a tool for publishing geospatial data as rdf graphs using r2rml mappings.

[17] Ora Lassila and Ralph R Swick. Resource description framework (rdf) model and syntax specification. 1999.

[18] Franck Michel, Johan Montagnat, et al. A survey of rdb to rdf translation approaches and tools. 2013.

[19] Kostas Patroumpas, Michalis Alexakis, Giorgos Giannopoulos, and Spiros Athanasiou. Triplegeo: an etl tool for transforming geospatial data into rdf triples. In *EDBT/ICDT Workshops*, pages 275–278, 2014.

[20] Matthew Perry and John Herring. OpenGIS geography markup language (GML) encoding specification. OpenGIS Standard OGC 11-052r4, Open Geospatial Consortium, Inc., 2012.

[21] Freddy Priyatna, Oscar Corcho, and Juan Sequeda. Formalisation and experiences of r2rml-based sparql to sql query translation using morph. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW ’14, pages 479–490, New York, NY, USA, 2014. ACM.

[22] Eric Prud’Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[23] Linda van den Brink, Paul Janssen, and Wilko Quak. From geo-data to linked data: Automated transformation from gml to rdf, 2013.