PlanetData

Network of Excellence

FP7 – 257641

D17.1 Call 2: Linked Map Read-write Linked Data enabled OGC WMS server

Coordinator: Jesús Barrera (GEOSLAB)
With contributions from: Francisco J Lopez-Pellicer (UNIZAR)

1st Quality reviewer: Adrian Brasoveanu (MODUL)
2nd Quality reviewer: Loris Bozzato (FBK)

Deliverable nature:	Prototype (P)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	M41
Actual delivery date:	M43
Version:	1.0
Total number of pages:	25
Keywords:	WMS, Linked Data, REST, Update

Abstract

This deliverable describes the work made on designing and developing the first prototype of a Linked Map Service (LMS). This service acts as a reverse proxy for Web Map Servers (WMS) and, at the same time, it behaves as a Linked Data frontend. Thus, from the point of view of a Geographic Information System (GIS) client, such as a Web map client, the LMS is a WMS server that accepts requests that conforms to the Open Geospatial Consortium WMS implementation specification version 1.3.0. It will be placed in front of a set of WMS servers and will determine where to route a particular request. However, for semantic enabled clients, the LMS server is a well-behaved Linked Data fronted that enables RDF and HTML browsers to navigate the content of the datastores accessible by the server and allows users to make, for example, comments and edits about geographic data.

Executive summary

The Linked Map project aims to demonstrate the enormous potential of linking together different sources of geographic information under the paradigm of Linked Data. Linked geographic data can be published by means of a Linked Data map. A Linked Data map is an interactive Web application that mixes maps and Linked Data. Two illustrative examples are the projects Opendatamap[1] and LinkedGeoData[2]. However, this combination has a slow pace of adoption even with success stories such as Ordnance Survey [1]. This is partially due to the fact that Linked data is perceived as a disruptive technology that is not interoperable with existing infrastructure investments based on standard geographic web services, such as Open Geospatial Consortium (OGC) Web map services (WMS). Additionally, it is very common in web applications with maps that users may add and edit content in the map. Today many Linked Data based sites usually offer only read-only access to the data. These circumstances may be hampering the adoption of Linked data in this context.

The Linked Map project has as one of its goals the development of a Linked Data extension for the WMS 1.3.0 specification. This deliverable describes the work made on designing and developing the first prototype of a Linked Map Service (LMS), a service that addresses the interoperability between map services and Linked Data. This service acts as a reverse proxy for WMS servers. At the same time, it behaves as a Linked Data front end. Thus, from the point of view of a GIS client, the LMS is a WMS server that accepts requests that conforms to the OGC WMS implementation specification version 1.3.0. It will be placed in front of a set of WMS servers and will determine where to route a particular request. However, for semantic enabled clients, the LMS server is a well-behaved Linked Data fronted that enables RDF and HTML browsers to navigate the content of the datastores accessible by the server and allows users to make, for example, comments and edits about geographic data served by the proxied WMS servers.

One of the goals of the development of LMS is the development of a maintainable and extensible code base based on well-known Java libraries. For such purpose, the Linked Map project has decided to use *Spring Boot*[3] (core), *Thymeleaf*[4] (view) and *Pubby*[5] (data access).

[1] http://opendatamap.ecs.soton.ac.uk/

[2] http://linkedgeodata.org/

[3] http://projects.spring.io/spring-boot/

[4] http://www.thymeleaf.org/

[5] http://wifo5-03.informatik.uni-mannheim.de/pubby/

Document Information

IST Project Number	FP7 - 257641	Acronym	PlanetData
Full Title	PlanetData		
Project URL	http://www.planet-data.eu/		
Document URL	http://wiki.planet-data.eu/web/D17.1		
EU Project Officer	Leonhard Maqua		

Deliverable	Number	D17.1	Title	Call2: Linked Map Read-write Linked Data enabled OGC WMS server
Work Package	Number	WP17	Title	Call2: Linked Map Read-write Linked Data WMS framework

Date of Delivery	Contractual	M41	Actual	M43
Status	version 1.0		final ☒	
Nature	prototype ☒ report □ dissemination □			
Dissemination level	public ☒ consortium □			

Authors (Partner)	Jesús Barrera (GEOSLAB), Francisco J Lopez-Pellicer (UNIZAR)			
Responsible Author	Name	Jesús Barrera	E-mail	jesusb@geoslab.es
	Partner	GEOSLAB	Phone	+34 976 065152

Abstract (for dissemination)	This deliverable describes the work made on designing and developing the first prototype of a Linked Map Service (LMS). This service acts as a reverse proxy for Web Map Servers (WMS). At the same time, it behaves as a Linked Data frontend. Thus, from the point of view of a Geographic Information System (GIS) client, such as a Web map client, the LMS is a WMS server that accepts requests that conforms to the Open Geospatial Consortium WMS implementation specification version 1.3.0. It will be placed in front of a set of WMS servers and will determine where to route a particular request. However, for semantic enabled clients, the LMS server is a well-behaved Linked Data fronted that enables RDF and HTML browsers to navigate the content of the datastores accessible by the server and allows users to make, for example, comments and edits about geographic data.
Keywords	WMS, Linked Data, REST, Update

Version Log			
Issue Date	Rev. No.	Author	Change
2014/03/25	0.1	Francisco J Lopez-Pellicer	Template instantiation
2014/04/05	0.2	Francisco J Lopez-Pellicer	First draft
2014/04/07	0.3	Jesús Barrera	Comments and review
2014/04/08	0.4	Francisco J Lopez-Pellicer	Adjustments
2014/04/08	0.5	Jesús Barrera	First version for QA
2014/04/17	0.7	Francisco J Lopez-Pellicer	Version after external reviews
2014/04/28	1.0	Francisco J Lopez-Pellicer	Final version

Table of Contents

Executive summary 3
Document Information 4
Table of Contents 5
List of figures 6
List of tables 7
Abbreviations 8
1 Introduction 9
1.1 The Linked Data map challenge 9
1.2 The Linked Map Service 9
1.3 Requirements 10
1.4 Development methodology 11
2 System architecture and design 13
2.1 Description 13
2.2 Architecture 13
2.3 Implementation 14
2.4 API 14
2.5 Third party libraries used 15
2.5.1 Spring Boot 15
2.5.2 Thymeleaf 15
2.5.3 Pubby 15
3 Status of server implementation 17
3.1 Release 0.8 17
3.2 Release 0.9 17
3.3 Release 1.0 17
4 Reviewing requirements and architecture plan 19
5 Conclusions 21
Appendix I LMS API 22
References 25

List of figures

Figure 1 – Linked Map service in context 10
Figure 2 – LMS interactions 13
Figure 3 – LMS components....14
Figure 4 – LMS capabilities page 22
Figure 5 – Embedded client for testing in LMS capabilities document 23

List of tables

Table 1 – LMS releases....12
Table 2 – Release comparison....19

Abbreviations

BCN25 Base Cartográfica Numérica 1:25.000 (Numeric Cartographic Base 1:25.000)

BTN25 Base Topográfica Nacional 1:25.000 (National Topographic Base 1:25.000)

LDP Linked Data Platform

LMS Linked Map Service

NGBE Nomenclátor Geográfico Básico de España (Basic Gazetteer of Spain)

OGC Open Geospatial Consortium

OSM OpenStreetMap

VGI Volunteer Geographic Information

WMS OGC Web map service

1 Introduction

This deliverable describes the work made on designing and developing the first prototype of a Linked Map Service (LMS), a service that addresses the interoperability between map services and Linked Data. This service acts as a reverse proxy for servers that implements the Open Geospatial Consortium Web Map Service (WMS) implementation specification version 1.3.0 [2] and, at the same time, it is able to dereference URIs using Linked Data best practices and techniques [3].

1.1 The Linked Data map challenge

A Linked Data map is an interactive Web application that mixes maps and Linked Data. Two illustrative examples are the projects Opendatamap[6] and LinkedGeoData[7]. Opendatamap makes use mainly of authoritative geographic information datasets that describe the University of Southampton and surroundings. These datasets contain data from UK government agencies such as Ordnance Survey and Royal Mail. LinkedGeoData publishes as Linked Data the content of OpenStreetMap, a well-known Volunteered Geographic Information (VGI) initiative that aims to map the Earth. The mix of geographic information Linked Data has enormous potential. However, this combination has a slow pace of adoption in very large producers of geographic information (NASA, ESA, national mapping agencies, cadastres, etc.) even with success stories such as Ordnance Survey [1]. Today, Linked Data is perceived as a disruptive technology that is not interoperable with existing infrastructure investments based on standard geographic web services, such as WMS. Additionally, Linked Data approaches usually offer read-only access to the data in a context where user edits are quite common. We believe that these circumstances are hampering the adoption of Linked data in this context.

1.2 The Linked Map Service

One of the goals of the Linked Map project is the development of a semantic enablement for OGC WMS in the context of the Semantic Web. This semantic enablement is named Linked Map Service (LMS) and can be characterized as a service that mixes:

- A Semantic Web interface with the semantics of an existing OGC Web interface (in our case the WMS standard).
- An interface compliant to an existing OGC Web service standard (in our case the WMS standard) that makes easy the access to Semantic Web services

A LMS is a reverse proxy of a remote WMS that at the same time can return RDF descriptions of served resources by content negotiation (service metadata, maps, data portrayed in maps). Also, a LMS implements direct URI and RESTful access and update to RDF data. This interface is different from the WMS interface. Both views are heavily and seamless interlinked. A WMS client can discover in the headers of a WMS response RFC 5988 `Link` headers [4] pointing to RDF descriptions of the response content. A semantic client can discover after dereferencing a resource URI links to alternate representations as images via WMS requests in the response headers (and in the content of the response).

Figure 1 shows the role of the LMS as an integrator of the semantic descriptions and maps portrayed by map services. Users through web map clients can access transparently to remote WMS servers. At the same time, users can access to enriched geodata published as Linked Data. Provenance information helps to understand that the geodata is about the content portrayed in maps. In addition, users can make rich annotations in the map using the read/write Linked Data that the LMS instance offers.

[6] http://opendatamap.ecs.soton.ac.uk/

[7] http://linkedgeodata.org/

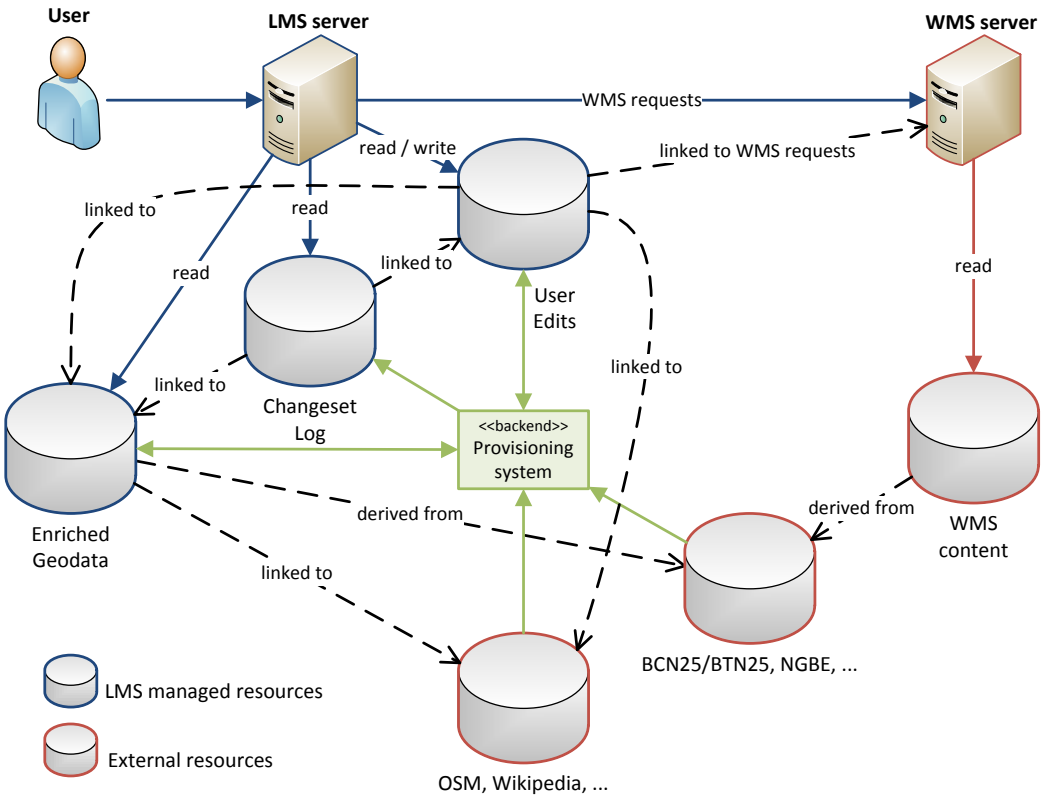

Figure 1 – Linked Map service in context

1.3 Requirements

There were some requirements defined in Section 7 *Linked Map Service API* of D15.1, "*Call 2: Linked Map requirements definition and conceptual architecture*" [5], (from R35 to R70) to be implemented in LMS:

- Core requirements:
  - R35. WMS 1.3.0 proxy server.
  - R36. Direct URI resolution of resources.
  - R37. REST API for resources.
- URI space requirements:
  - R38. LMS Ontology.
  - R39. Resource URI.
  - R40. Container URI.
- URI template requirements:
  - R41. Identifier URI template for real-world objects.
  - R42. Identifier URI template for definitions.
  - R43. Document URI template.
  - R44. Representation URI template.
  - R45. Online resource URI template.
  - R46. Representation URI template is behaving as an online resource URI template.
- WMS 1.3.0 proxy server requirements:

- R47. Reverse proxy for remote WMS servers.
- R48. Be conform to WMS specification.
- R49. Overloading WMS requests.

- Direct URI requirements:
  - R50. Redirection of URIs for identifiers to document URIs.
  - R51. Redirection of URIs for identifiers to representation URIs.
  - R52. Status code of representation URI response.
  - R53. Resource representation in HTML.
  - R54. Resource representation in TURTLE.
  - R55. Content negotiation with Accept header of document URIs.
  - R56. Content negotiation with Accept header of representation URIs.
  - R57. Representation URIs may trigger a WMS request.
  - R58. Access control.
- Extended REST API requirements:
  - R59. Support of query parameters in document and representation URIs.
  - R60. Restrict resource modification to users and user edits.
  - R61. Support of POST requests restricted to users and user edits.
  - R62. Support of DELETE requests restricted to users and user edits.
  - R63. Support of PUT/PATCH requests restricted to users and user edits.
- Resource discovery requirements:
  - R64. Advertise LMS presence with a Link header.
  - R65. Add a Link header to the right URI for retrieving the resource.
  - R66. Add a Link header to the primary topic of the representation.
  - R67. Add a Link header to the preferred representation of a resource.
  - R68. Add to WMS responses a Link header pointing to representation URIs.
  - R69. Add to responses Link headers pointing to alternate representations.
  - R70. Each Link header added must be also present in the RDF representation.

From this starting point, during WP17 the main requirements have been implemented, and other have suffered some changes caused by the feedback during the development of releases 0.8, 0.9 and 1.0 (see Section 3). These releases do not comprise the code that will be used by the Linked Map platform (D18.1 [6] and D18.2 [7]). Hence D18.1 and D18.2 will cover the fulfilment of requirements not fully closed by release 1.0 or changed due to the feedback obtained from the users of the Linked Map platform.

1.4 Development methodology

The development methodology used for the development of LMS is a mix of the iterative software engineering lifecycle (1 month cycle) and test driven development. The development of LMS is product of the activities of the WP17, hence the development must also adhere to internal quality management practices described in D21.1 *Call 2: Linked Map Project Handbook* [8]. For example, the development methodology should apply the following SPICE [9], capacity level 2, processes: ENG.1 Requirements Elicitation, MAN.3 Project Management, SUP.1 Quality Assurance and SUP.8. Configuration Management.

The process of requirements elicitation started on WP15 and has been supported with the tool Bugzilla. Bugzilla has been also used as the tool to track the development of the LMS in the project management

process. Regarding to our internal quality assurance process, we should note that it is similar to the Planet Data quality assurance process and it is targeted to ensure that outcomes send to Planet Data reviewers are already aligned with Planet Data objectives. Finally, the configuration management process has defined the LMS release policy (Table 1), and uses SVN as the version control system, Maven as the dependency manager, and Bugzilla as the configuration manager.

Table 1 – LMS releases

Release	Description	Date
0.8	Read Linked Data WMS service	Early April 2014
0.9	Read Linked Data WMS service	Late April 2014
1.0	Read Linked Data WMS service	Late May 2014

The LMS release 1.0 was initially planed to late April 2014 but due to the complexity of some issues and the need of synchronization with the development of the LMS client (D17.2 [10]) the release has been postponed to late May. The deliverable on the LMS client (D17.2) will document LMS release 1.0.

2 System architecture and design

This section describes the LMS systems, the involved components and details of the implemented logic.

2.1 Description

Figure 2 shows the interactions of a LMS instance. WMS clients perform WMS requests to the LMS instance as it is advertising himself as a WMS server. WMS requests received from WMS clients are sent to remote WMS servers on behalf of them and server responses are sent back to the clients along with Web links, a kind of HTTP header, that points to alternative representations of the WMS response. LMS clients dereference URIs that are either identifier URIs, document URIs or WMS KVP requests (i.e. API requests). The LMS instance returns either the most suitable representation or redirects to it. If the most suitable representation requires data, the data are obtained from the corresponding data repository. However, if the most suitable representation is an image, a request is made to a remote WMS for obtaining it. Both responses contain Web links that point to alternative representations of the requested resource. The LMS instance enables clients to update a subset of the URI space managed by the LMS instance. These updates are stored in a data repository.

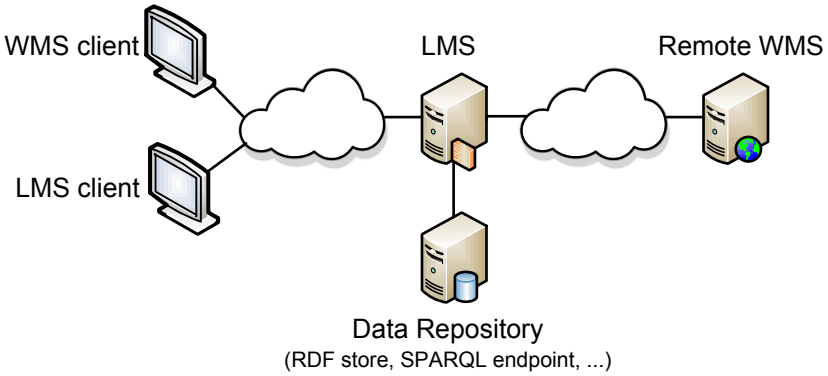

Figure 2 – LMS interactions

2.2 Architecture

At a high-level view nine component modules can be identified (Figure 3). These are the Web controllers module, different endpoint logic modules, and the data management module. These modules have been developed in the project.

The Web controllers module offers to users a web application that exposes the following interfaces:

- The interface specification WMS 1.3.0 (KVP and XML style).
- The interface for accessing RDF data (GET).
- The interface for modifying RDF data (POST, PUT, PATCH and DELETE)
- The interface for discovering the capabilities of the LMS that exposes the configuration.

The mapping between URIs patterns and supported interfaces is defined by configuration.

There are seven Endpoint classes that implement the business logic of LMS, including 303 redirects, domain specialization, data extraction and response adaptation. Their names are self-descriptive and identify a HTTP verb (GET, POST, DELETE, PUT, PATCH) or a WMS request style (KVP, XML). Each module is responsible for providing functionality related to local or remote data manipulation. KVP and XML modules can access to remote WMS instances for retrieving metadata and georeferenced images.

The data management module manages the access to local and remote RDF stores and mainly reuses existing *Pubby* codebase.

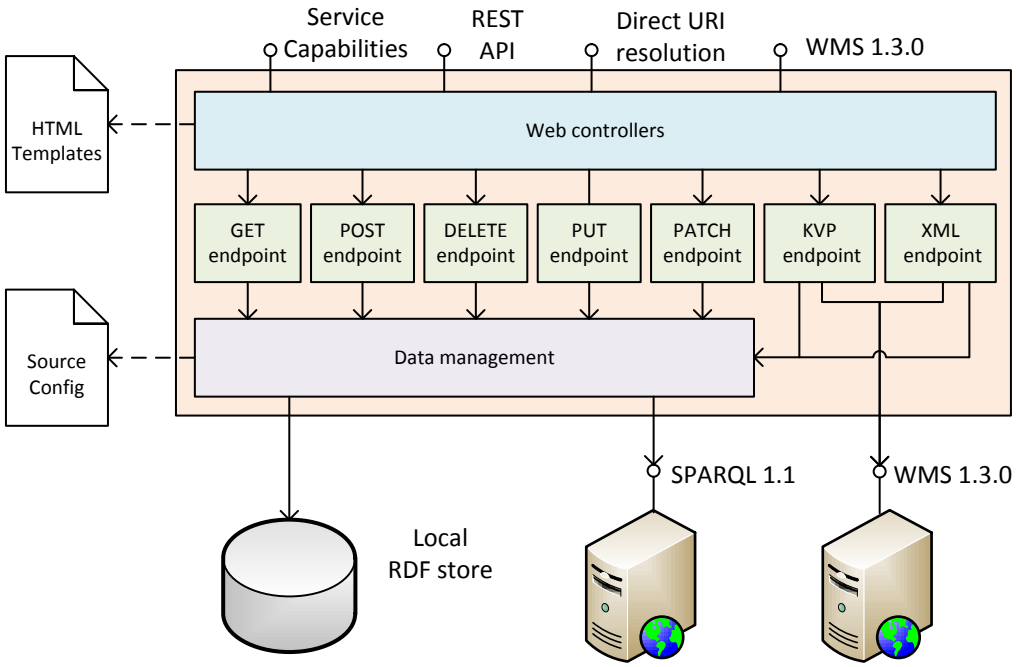

Figure 3 – LMS components

2.3 Implementation

Web controllers are written in the Java language with Spring Annotations provided by Spring Boot. The Spring Boot framework enables LMS instances to run standalone with an embedded Tomcat or Jetty server or to be deployed in a servlet container. The instance is backed locally by Apache Jena and can connect remotely to SPARQL endpoints. RDF data is serialized using the Turtle language [11] by default. Responses are compressed if the requester supports compression.

LMS has been implemented to serve different kind of resources and representations that are grouped in containers. Thus URIs valid on a LMS instance can follow the URI patterns:

- `/{kind}/{container}{.extension}{?query}` for containers, and
- `/{kind}/{container}/{resource}{.extension}{?query}` for resources.

These URI templates follow the RFC 6570 [12] that provides a mechanism for abstracting a space of URIs. The text enclosed between '`{`' and '`}`' denotes an expression with an optional operator (`.`, `?`) that will be expanded with values assigned to variable names (`kind`, `container`, `resource`, `extension`, `query`). The URI space includes optional path selectors and query parameters. Path selectors are used only in URIs that denotes documents. Query parameters are used in proxied WMS requests that follow the WMS KVP style or, for example, when the resource can be paged [13]. Requests to the URI space managed by a LMS instance are mapped to external entities by the data management module after extracting information using these patterns.

2.4 API

Appendix I describes the actual API in detail. The LMS API is composed by reverse proxy operations and operations oriented to the following resources: service capabilities, read only containers, writeable containers, read only resources and writeable resources. The interface is generic although only resources of type users and user edits may be writeable.

The API documentation will be available online as the future instance of the LMS 1.0 that will be deployed at the Linked Map website is able to publishes human readable documentation about itself. The API and the contents of this instance may change due to the evolution of the project.

2.5 Third party libraries used

One of the goals of the development of LMS is the development of a maintainable and extensible codebase based on well-known Java libraries. For such purpose, the Linked Map project has decided to use *Spring Boot*[8] (core), *Thymeleaf*[9] (view) and *Pubby*[10] (data access).

2.5.1 Spring Boot

Spring Boot is part of the execution layer of the Spring IO Platform (formerly Spring Framework). Spring IO Platform is a popular open source application framework and inversion of control container for the Java platform. *Spring Boot* simplifies the development of Spring-based projects and implements out of the box features such as automated health checking and metrics endpoints.

Spring Boot is a core component of LMS. LMS leverages of *Spring Boot*:

- Configuration of application components, done mainly via inversion of control and coding by convention.
- Model-view-controller framework, used for the implementation of web services.
- Embedded servlets containers, for standalone deployment of instances.
- Authentication and authorisation.
- Testing tools for being used during the development.

2.5.2 Thymeleaf

Thymeleaf is an XML/XHTML/HTML5 template engine written in Java suited for serving as the view layer of web applications. It enables the development of templates that can be correctly displayed by browsers so they can be used as static prototypes of the view layer.

Thymeleaf is the template engine used by LMS for serving human readable views of resources. LMS leverages of *Thymeleaf*:

- Template engine for XML, XHTML and HTML5
- Natural templates for prototyping, i.e. templates that can be rendered statically in a browser without running an application server.
- Parsed template cache.
- URL rewriting capabilities for adding context.
- Spring IO integration, including security and internationalization.

Also, a *Thymeleaf* template is served as human readable capabilities document of the LMS instance.

2.5.3 Pubby

Pubby is a Linked Data interface to local or remote SPARQL protocol servers. *Pubby* maps URIs dereferenceables into requests to description about resources stored in local or SPARQL-based repositories. A notable feature of Pubby is that it can rewrite URIs found in local or SPARQL-exposed datasets into a different namespace.

Pubby is only used for supporting data access. LMS leverages of Pubby:

- Data source configuration.
- Access to local RDF stores.

[8] http://projects.spring.io/spring-boot/

[9] http://www.thymeleaf.org/

[10] http://wifo5-03.informatik.uni-mannheim.de/pubby/

- Access to remote SPARQL 1.1 endpoints.
- Mapping between local and remote URIs.
- Rewriting of retrieved URIs.
- Algorithm for content negotiation, which is more complex that the provided by *Spring Boot*.

Note that *Spring Boot* and *Thymeleaf* have superseded several features offered by *Pubby*. The Linked Data interface, the 303 redirects and the content negotiation logic provided by *Pubby* have been replaced by *Spring Boot* based code. *Thymeleaf* has replaced the template engine provided by *Pubby*.

Pubby is not used as library. *Pubby* code that implements the features used by LMS has been moved to the Java package `es.unizar.iaaa.lms.pubby`.

3 Status of server implementation

This section covers the requirements defined in Section 7 *Linked Map Service API* of D15.1 (from R35 to R70) that were covered by the releases 0.8 (March 2014), 0.9 (April 2014) 1.0 (May 2014) of the LMS. Release 1.0 does not comprise the code that will be used by the Linked Map Platform (D18.1 and D18.2). D18.1 and D18.2 will cover the fulfilment of requirements not fully closed by release 1.0.

3.1 Release 0.8

LMS 0.8 accepts requests that conform to the WMS 1.3.0 implementation specification, and then works as proxy server to remote WMS servers. LMS 0.8 is able to dereference URIs using Linked Data best practices and techniques.

The URI space managed by this release is subject to the rules defined in Section 6 *Designing URIs for the Linked Map project* of D15.1. The URI space is organized in such way that URI patterns clearly identify if the URI refers to a resource or a container of resources. This release is also able to manage different kinds of URI templates that are attached by configuration to specialized Endpoint classes. URI templates supported are those defined in Section 6 *Designing URIs for the Linked Map project* of D15.1.

As it was stated above, this release acts as a transparent reverse proxy server for WMS 1.3.0 servers. That is, an instance is able to sit in front of a set of remote WMS servers, to advertise WMS endpoints, to determine the remote server to route a WMS request to those endpoints, and to return its response to the requester. The support in this release is restricted to the HTTP KVP style defined in the WMS implementation specification 1.3.0.

This release implements a well-behaved read only Linked Data interface for data stored in RDF files, Apache Jena-based RDF stores, and SPARQL endpoints. LMS internally uses a tailored and stripped from rendering code version of Pubby as data access manager. URI redirection, content negotiation and resource representation in this release has been developed using Spring Boot 1.0.

3.2 Release 0.9

LMS 0.9 is an intermediate step prior to release 1.0. This version advertises the support of the LMS model and the relations of the resources managed by adding specific HTTP `Link` headers in all responses to request made to the URI space managed by a LMS instance (R64 to R70). Additionally, the release partially addresses the description of the URI space managed by the LMS instance encoded by means of an ontology (R38).

3.3 Release 1.0

LMS 1.0 will support a read-write REST API interface for accessing, updating, creating and deleting some resources (R37). This interface will implement practices and techniques described in *Linked Data Platform* version 1.0 [13]. This release will provide a description of the URI space managed by the LMS instance encoded in an ontology (R38).

This release will support representation URI templates that behave as online resource URI templates in the circumstances defined in R46. This requirement will require the replacement of the logic for deciding the Endpoint instance that should process the request. The new logic is planed to be based on accept and reject URI templates rules.

The two main features of this planned release are a full-fledged support of a write REST API based on best practices and techniques discussed in LDP 1.0, and the overloading of WMS requests (R49) for intertwining WMS and Linked Data interfaces. The intertwining logic is triggered by some URI templates (R46) or as a consequence of content negotiation (R57).

Additionally, this release is planned to support:

- WMS requests that conform to the HTTP XML style defined in the WMS implementation specification 1.3.0.
- Advertising of deleted resources (i.e, `410 Gone` status code)

- Triggering a WMS request for a raw representation of the resource.
- Authentication and access control will be also implemented. The technology to be utilized is Spring Security.
- Query parameters in GET requests for Linked Data.

4 Reviewing requirements and architecture plan

Deliverable D15.1 organized LMS requirements in seven sections. These sections group core requirements, URI space requirements, URI template requirements, WMS 1.3.0 proxy server requirements, direct URI requirements, and extended REST API requirements and resource discovery requirements. Table 2 presents the full list of LMS requirements along with their coverage by the existing releases of LMS. The implementation of the release 1.0 will be described in D17.2. The LMS release 1.0 is a key component of the Linked Map Platform (D18.1 and D18.2) and therefore we express our commitment to implement it before its launch.

Table 2 – Release comparison

Requirement	0.8	0.9	1.0
Core requirements			
R35. WMS 1.3.0 proxy server	✔	✔	✔
R36. Direct URI resolution of resources	✔	✔	✔
R37. REST API for resources			✔
URI space requirements			
R38. LMS Ontology		partial	✔
R39. Resource URI	✔	✔	✔
R40. Container URI	✔	✔	✔
URI template requirements			
R41. Identifier URI template for real-world objects	✔	✔	✔
R42. Identifier URI template for definitions	✔	✔	✔
R43. Document URI template	✔	✔	✔
R44. Representation URI template	✔	✔	✔
R45. Online resource URI template	✔	✔	✔
R46. Representation URI template behaving as online resource URI template			✔
WMS 1.3.0 proxy server requirements			
R47. Reverse proxy for remote WMS servers	KVP	KVP	✔
R48. Be conform to WMS specification	KVP	KVP	✔
R49. Overloading WMS requests			✔
Direct URI requirements			
R50. Redirection of URIs for identifiers to document URIs	✔	✔	✔
R51. Redirection of URIs for identifiers to representation URIs	✔	✔	✔
R52. Status code of representation URI response	200, 404	200, 404	✔
R53. Resource representation in HTML	✔	✔	✔
R54. Resource representation in Turtle	✔	✔	✔
R55. Content negotiation with `Accept` header of document URIs	✔	✔	✔
R56. Content negotiation with `Accept` header of representation URIs	✔	✔	✔
R57. Representation URIs may trigger a WMS request			✔
R58. Access control			✔

Requirement	0.8	0.9	1.0
Extended REST API requirements			
R59. Support of query parameters in document and representation URIs			✔
R60. Restrict resource edition to users and user edits			✔
R61. Support of POST requests restricted to users and user edits			✔
R62. Support of DELETE requests restricted to users and user edits			✔
R63. Support of PUT/PATCH requests restricted to users and user edits			✔
Resource discovery requirements			
R64. Advertise LMS presence with a `Link` header		✔	✔
R65. Add a Link header to the right URI for retrieving the resource		✔	✔
R66. Add a Link header to the primary topic of the representation		✔	✔
R67. Add a Link header to the preferred representation of a resource		✔	✔
R68. Add to WMS responses a Link header pointing to representation URIs		✔	✔
R69. Add to responses Link headers pointing to alternate representations		✔	✔
R70. Each Link header added must be also present in the RDF representation		✔	✔

5 Conclusions

This deliverable has described the work made on the first prototype of a LMS, a service that addresses the interoperability between map services and Linked Data. LMS acts as a reverse proxy for servers that implements the WMS implementation specification version 1.3.0 and, at the same time, it is able to dereference URIs using Linked Data best practices and techniques.

This service, along with its client (that will be described in D17.2) will be deployed at the Linked Map platform (that will be described in D18.1). The sources of the releases are found on the Linked Map website. The development of LMS will continue along the project to adapt the system to the feedback from the development of the client and the platform.

Appendix I LMS API

The LMS API is composed by reverse proxy operations and operations oriented to the following resources: service capabilities, read only containers, writeable containers, read only resources and writeable resources. The interface is generic although only resources of type users and user edits may be writeable.

Reverse proxy: reverse proxy for remote WMS.

GET (KVP) `/api/servers/{resource}{?query}`

GetCapabilities, GetMap and GetFeatureInfo HTTP GET calls to a WMS server identified by `resource`. The `query` fragment encodes the requested operation.

POST (XML) `/api/servers/{resource}`

GetCapabilities, GetMap and GetFeatureInfo HTTP POST calls to a WMS server identified by `resource`. The body of the request encodes in XML the requested operation.

In both cases, the variable `resource` is extracted from the URI path and then it is mapped to a URI that identifies a resource in a RDF store. The endpoint address of the WMS server is a property of such resource.

Service capabilities: Self-descriptive information about the instance for humans.

GET /

Returns a human readable description of the capabilities of the LMS instance.

This request returns a HTML page dynamically generated from the configuration of the LMS instance. This page acts as documentation (Figure 4) and as sandbox (Figure 5) for interacting with the LMS instance. It is inspired by Swagger-UI[11].

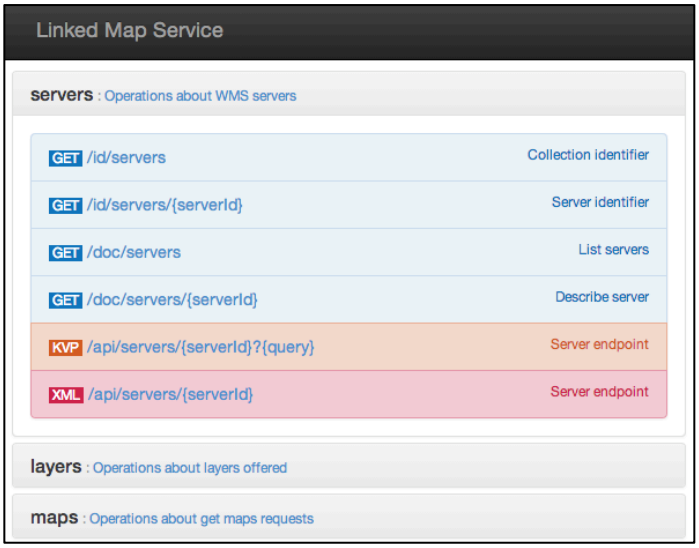

Figure 4 – LMS capabilities page

[11] https://github.com/wordnik/swagger-ui

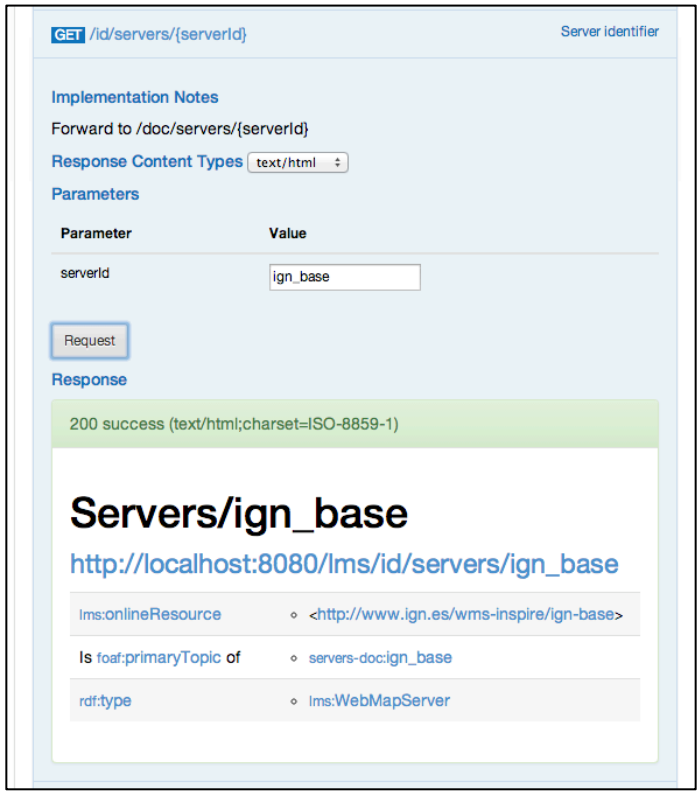

Figure 5 – Embedded client for testing in LMS capabilities document

Read only container: list of the contents of a container

GET `/doc/{container}{.extension}{?query}`

Query for the list of resources contained in a container.

The variable `container` is extracted and mapped to a container in a local RDF store or a SPARQL based RDF store. The `query` fragment may restrict the data returned. The parameters depend of the kind of resource. The variable `extension` may be used for selecting the response content type if the requester does not uses content negotiation.

Writeable container: list of the contents of a writeable container.

GET `/doc/{container}{.extension}{?query}`

Query for the list of resources contained in a container.

POST `/doc/{container}`

Add a new resource to the container.

The container may reject the operation if the submitted data is not compatible with the resources that the container may contain or if the operation is not authorized. The submitted body is a set of RDF data. The result informs the location of the newly added resource.

Read only resource: information about a resource

GET `/doc/{container}/{resource}{.extension}{?query}`

Query for the description of a resource.

The resource is identified by the variables `container` and `resource`. Both variables are mapped to a resource in a local RDF store or a SPARQL based RDF store. The `query` fragment may restrict the data returned. The parameters depend of the kind of resource. The variable `extension` may be used for selecting the response content type if the requester does not uses content negotiation.

Writeable resource: information about a writeable resource.

GET `/doc/{container}/{resource}{.extension}{?query}`

Query for the description of a resource.

DELETE `/doc/{container}/{resource}`

Delete the description of a resource and all its representations.

The request URI is mapped to a resource URI in a local RDF store or a SPARQL based RDF store. The corresponding store must not be read only (e.g. support SPARQL UPDATE).

PUT `/{kind}/{container}/{resource}`

Replace the description of a resource.

The request URI is mapped to a resource URI in a local RDF store or a SPARQL based RDF store. The resource URI must exist, the corresponding store must not be read only, and the new representation should be in form of triples in a processable format.

PATCH `/{kind}/{container}/{resource}`

Modify for the description of a resource.

The request URI is mapped to a resource URI in a local RDF store or a SPARQL based RDF store. The resource URI must exist, the corresponding store must not be read only (e.g. support SPARQL UPDATE) and the representation of changes should be in form of triples added and triples deleted in a processable format (e.g. support SPARQL UPDATE).

References

[1] J. Goodwin, C. Dolbear, and G. Hart, “Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web,” *Transactions in GIS*, vol. 12, pp. 19–30, Dec. 2008.

[2] J. de la Beaujardiere, Ed., “OpenGIS® Web Map Server Implementation Specification,” Open Geospatial Consortium Inc., OGC 06-042, Mar. 2006.

[3] T. Heath and C. Bizer, *Linked Data: Evolving the Web Into a Global Data Space*, vol. 1, no. 1. Morgan & Claypool Publishers, 2011.

[4] M. Nottingham, “Web Linking,” RFC 5988, Oct. 2010.

[5] F. J. Lopez-Pellicer and J. Barrera, “D15.1 *Call 2: Linked Map requirements definition and conceptual architecture*,” PlanetData, 2014.

[6] J. Barrera and F. J. Lopez-Pellicer, “D18.1 Call 2: Linked Data Platform Alpha version ,” PlanetData.

[7] J. Barrera and F. J. Lopez-Pellicer, “D18.2 Call 2: Linked Data Platform Beta version .”

[8] F. J. Lopez-Pellicer and J. Barrera, “D21.1 Call 2: Linked Map Project Handbook ,” PlanetData, 2014.

[9] H. Van Loon, *Process Assessment and ISO/IEC 15504*. Springer, 2004.

[10] J. Barrera and F. J. Lopez-Pellicer, “D17.2 Call 2: Linked Map Read-write Linked Data enables OGC WMS client,” PlanetData.

[11] E. Prud'hommeaux and G. Carothers, Eds., “RDF 1.1 Turtle,” W3C Recommendation 25 February 2014.

[12] J. Gregorio, R. T. Fielding, M. Hadley, M. Nottingham, and D. Orchard, “URI Template,” RFC 6570.

[13] S. Speicher, J. Arwe, and A. Malhotra, Eds., “Linked Data Platform 1.0,” *W3C*, 30-Jul-2013. [Online]. Available: http://www.w3.org/TR/2013/WD-ldp-20130730/. [Accessed: 28-Jan-2014].