Enablers Deployment
- First Release

DECEMBER 2014

SDI4Apps

DELIVERABLE

Project Acronym: **SDI4Apps**
Grant Agreement number: **621129**
Project Full Title: **Uptake of Open Geographic Information Through Innovative Services Based on Linked Data**

D3.2.1 ENABLERS DEPLOYMENT – FIRST RELEASE

Revision no. 04

Authors: Martin Kuba (Masaryk University)
Tomáš Sapák (Masaryk University)
Karel Charvát (Czech Centre for Science and Society)
Raitis Berzins (Baltic Open Solutions Center)
Michal Kepka (University of West Bohemia)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY

Revision	Date	Author	Organisation	Description
01	19/12/2014	Tomáš Sapák, Martin Kuba	Masaryk University	Initial draft
02	22/12/2014	Martin Tuchyňa, Tomáš Kliment	SAŽP, E-PRO	Contribution with initial pilot deployment test
03	29/12/2014	Karel Charvát, Raitis Berzins, Michal Kepka	CCSS, BOSC, UWB	Specific GI enablers
04	30/12/2014	Tomáš Mildorf	UWB	Final revision and formatting

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Disclaimer:

Views expressed in this document are those of the individuals, partners or the consortium and do not represent the opinion of the Community.

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

This document describes the initial deployment of generic and specific enablers for the SDI4Apps platform in the cloud operated by CERIT-SC.

The generic enablers are the IaaS cloud operated by CERIT-SC, based on the OpenNebula cloud management system, KVM and Xen hypervisors, disk images with preinstalled Debian Linux operating system, Perun account management system, and powerful hardware machines.

The specific enablers are specifically prepared VM templates and disk images with preinstalled software suited for spatial data infrastructure. The software includes Postgres-XL clustered database with PostGIS extension, implementation of the Web Map Service named MapServer, nginx web server with caching and load-balancing capabilities, HAProxy load-balancer, metadata catalogue and visualisation tool. Some recommendations for configuration and usage of this software are also provided including a short pilot related deployment test.

1 DEPLOYED GENERIC ENABLERS

The IaaS cloud infrastructure of CERIT-SC is based on OpenNebula cloud management software. It has a web interface accessible at the URL https://cloud.metacentrum.cz/

User account management including applications for new accounts, password resets, personal data updates, group membership management etc. is provided by the Perun system, which is used for managing both the CERIT-SC and CESNET national e-infrastructures.

An application for a new account can be entered at http://bit.ly/sdi4apps, user identity can be verified using federated identity mechanisms of either the pan-European federation of educational and research institutions EDUgain, the Czech national federation of education and research institutions eduID.cz, or using the well-known social networks and identity providers Google+, Facebook, LinkedIn and MojeID.

The CERIT-SC cloud uses Xen and KVM hypervisors. It provides disk images with preinstalled Debian Linux, CentOS, SciLinux, and MS-Windows.

For auto scaling, OpenNebula offers a component named OneFlow, which allows setting up scaling policies based on metrics data from monitoring or time schedule.

To make usage simpler for new users, a new group named sdi4apps was created, which contains all participants of the SDI4Apps project who have applied for an account in the CERIT-SC cloud. Members of the group can see disk images and templates prepared specifically for this project. The disk images have preinstalled Debian Linux operating system with a selection of software suited for spatial data infrastructure.

Documentation for the CERIT-SC cloud is provided in the project's wiki at

http://redmine.ccss.cz/projects/sdi4apps/wiki/CERIT-SC_Cloud_Infrastructure

The CERIT-SC cloud uses a pool of physical machines with the following specifications (as of December 2014):

- 32 machines with 8 CPUs and 128GB RAM each
- 48 machines with 12 CPUs and 90GB RAM each
- 112 machines with 16 CPUs and 128GB RAM each
- 10 machines with 24 CPUs and 96GB RAM each

Should the need arise, more CPUs can be added from CESNET's MetaCentrum computing infrastructure collecting theoretically up to 10160 CPUs in total as of December 2014.

2 DEPLOYED SPECIFIC ENABLERS

2.1 Postgres-XL

Postgres-XL is deployed in the CERIT-SC cloud in the form of two virtual machine templates with corresponding virtual machine images.

Installed templates:

- sdi4apps-gtm
- sdi4aps-postgresxl

Installed images:

- sdi4apps-gtm
- sdi4apps-postgresxl

Both images are installed with Debian Wheezy OS, Postgres-XL 9.2rc and PostGIS 2.0.6. sdi4apps-gtm additionally contains the pgxc_ctl utility.

Sdi4apps-gtm serves two purposes. It accommodates Global Transaction Manager and is used to initialize, control and monitor the Postgres-XL cluster with the pgxc_ctl utility. The Sdi4apps-postgresxl template is used for instantiating clustered database nodes.

The current deployment consists of these virtual machines, which form Postgres-XL database clusters:

- sdi4apps-gtm
- sdi4apps-postgresxl-0
- sdi4apps-postgresxl-1
- sdi4apps-postgresxl-2

Any project participant can create from these templates their own instance of the cluster with his own number of nodes. The guide to setup *n* node cluster, is as follows:

1. Instantiate one sdi4apps-gtm and *n* sdi4apps-postgresxl virtual machines
2. Login to sdi4apps-gtm and edit in /home/osm/pgxc_ctl/pgxc_ctl.conf the following lines:
    - gtmMasterServer= sdi4apps-gtm-DNS_name
    - coordMasterServers=(sdi4apps-postgresxl-1-DNS_name)
    - datanodeMasterServers=(sdi4apps-postgresxl-1-DNS_name)
3. Generate the ssh key pair under the osm user on GTM:

    ```
    ssh-keygen –t rsa
    ```

4. Copy the content of /home/osm/.ssh/id_rsa.pub to /home/osm/.ssh/authorized_keys to all nodes (even to GTM)
5. Run under osm user on GTM:

    ```
    pgxc_ctl init all
    ```

    GTM and the cluster with first node (sdi4apps-postgresxl-1-DNS_name) will be initialized.
6. Run on GTM to add all remaining nodes to the cluster (replacing X accordingly):

    ```
    pgxc_ctl add coordinator master coordX sdi4apps-postgresxl-X-DNS_name 20004 20010 /home/osm/pgxc/nodes/coord
    ```

    ```
    pgxc_ctl add datanode master datanodeX sdi4apps-postgresxl-X-DNS_name 20008 20012 /home/osm/pgxc/nodes/dn_master datanode1
    ```

2.2 Scalable Web Map Service

A Web Map Service[1] (WMS) is a standard protocol defined by the Open Geospatial Consortium (OGC) for serving map images over the Internet that are generated by a map server using data from a GIS[2] database and GIS file formats.

The goal in the SDI4Apps infrastructure is to prepare a scalable cloud deployment of WMS services, which would allow faster response times.

The scalability of WMS could be achieved in two ways:

- scalability to many simultaneous requests from many clients
- scalability to process more data for a single request or to process the same amount of data in less time

The latter depends mainly on the scalability of the PostGIS database and thus can be achieved by the clustered Postgres-XL as described in the chapter2.1 .

The scalability to many simultaneous requests can be achieved by distributing the request processing to many CPUs.

Two software implementations of WMS were considered: MapServer and GeoServer. The software MapServer was selected, because while GeoServer is a server implemented in Java, which has some extension for clustering, however MapServer is a simple CGI[3] program and thus it is more scalable by simply running it multiple times independently.

The latest version of MapServer can be installed easily on Linux Debian 8 (Jessie) from its standard repositories.

MapServer generates a new image for every request, even if the requests are for the same image. We made experiments with a MapServer connected to a local PostGIS loaded with the OpenStreetMap database. Processing each request took many seconds, with CPU load divided alternately between the MapServer executable and the PostGIS database. A single request cannot be made faster in the single-threaded MapServer, but multiple simultaneous requests can be spread over multiple machines and the results can be cached. Thus a scalable setup needs a **HTTP cache** and a **load-balancer**. See the diagram in the Figure 1:

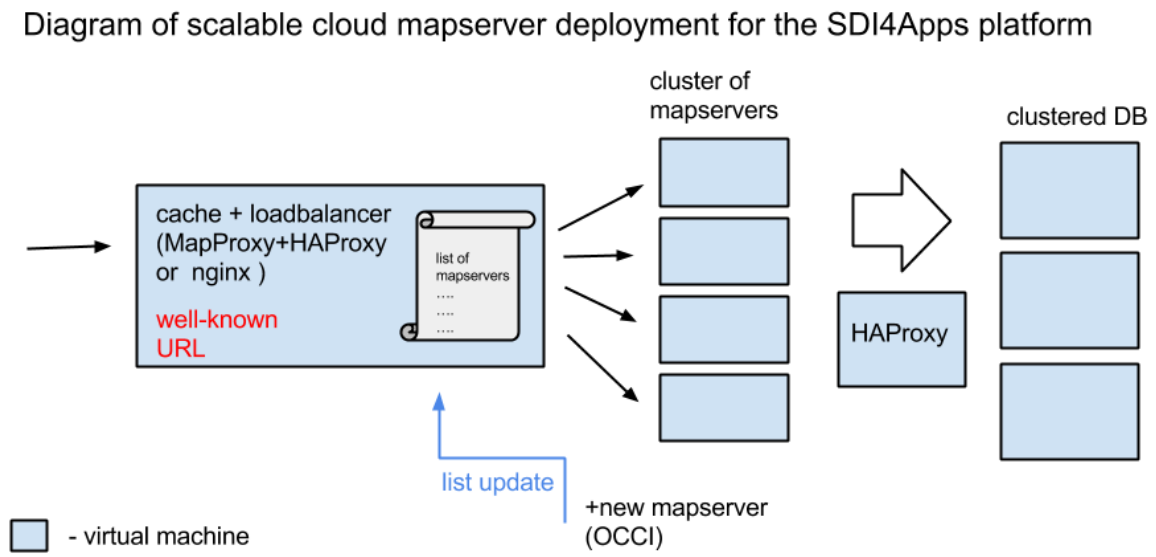

Figure 1: Diagram of scalable WMS

[1] http://en.wikipedia.org/wiki/Web_Map_Service
[2] Geographic Information System
[3] Common Gateway Interface – a standard interface between a web server and programs that generate web content

A single well-known URL would be published for clients, thus the service would have a single entry point.

The purpose of the **HTTP cache** is to reduce processing time for subsequent requests for the same URL. On the first request for a URL, the request is passed to the MapServer to generate its response and the response is cached. On subsequent requests for the same URL, the cached response is returned, reducing the response time significantly.

The purpose of the **load-balancer** is to evenly spread processing of HTTP requests to multiple CPUs located possibly on multiple machines when processing concurrent requests.

The Wikipedia page about WMTS[4] (Web Map Tile Service) states that the WMS standard proved to be difficult to implement for situations where short response times are important. That has led to the development of standards WMS-C (WMS Tile Cache), TMS (Tile Map Service) and WMTS (Web Map Tile Service) which restrict the set of tiles that can be requested.

The scalability of WMS would be achieved by internally load-balancing to multiple map servers. Multiple VMS with map servers can be started to enable scalability to more CPUs than a single computer may have. When a new VM with a map server would be started, the list of available map servers in the load-balancer must be updated.

We selected 3 pieces of software that fit the functional requirements for the proxy or load-balancer:

- **MapProxy** - a cache which can be seeded (preloaded with selected images), but does not load-balance
- **HAProxy** - a load-balancer on both HTTP and TCP network layers, but does not cache
- **nginx** - a web server that can act as both a cache and a load-balancer, however it cannot seed itself

An interesting idea is to use nginx for both cache and load-balancing, and seeding its cache using a MapProxy seeding requesting tiles from nginx.

2.2.1 Load-balancing Algorithms

HAPRoxy and nginx are both high-performance load-balancers, however they each support different set of load-balancing algorithms.

Nginx supports the following algorithms[5]:

- **round-robin** — requests to the application servers are distributed in a round-robin fashion
- **least-connected** — next request is assigned to the server with the least number of active connections
- **ip-hash** — a hash-function is used to determine what server should be selected for the next request (based on the client's IP address)
- **hash**— (since version 1.7.2) mapping is based on a hashed key value, the key can contain text, variables, and their combinations

HAProxy supports the following algorithms[6]:

- **round-robin**— Each server is used in turns, according to their weights. This algorithm is dynamic, which means that server weights may be adjusted on the fly for slow starts for instance. It is limited by design to 4095 active servers per backend.
- **static-rr** - Each server is used in turns, according to their weights. It is static, which means that changing a server's weight on the fly will have no effect. On the other hand, it has no design limitation on the number of servers

[4] http://en.wikipedia.org/wiki/Web_Map_Tile_Service
[5] http://nginx.org/en/docs/http/load_balancing.html
[6] http://cbonte.github.io/haproxy-dconv/configuration-1.5.html#4.2-balance

- **leastconn** - The server with the lowest number of connections receives the connection. Round-robin is performed within groups of servers of the same load.
- **first** - The first server with available connection slots receives the connection. The purpose of this algorithm is to always use the smallest number of servers so that extra servers can be powered off during non-intensive hours.
- **source** - The source IP address is hashed and divided by the total weight of the running servers. This ensures that the same client IP address will always reach the same server as long as no server goes down or up.
- **uri** - This algorithm hashes either the left part of the URI (before the question mark) or the whole URI and divides the hash value by the total weight of the running servers. This ensures that the same URI will always be directed to the same server as long as no server goes up or down.
- **url_param** - The URL parameter specified in argument will be looked up in the query string of each HTTP GET request. This is used to track user identifiers in requests and ensure that a same user ID will always be sent to the same server as long as no server goes up or down.
- **hdr(<name>)** - The HTTP header <name> will be looked up in each HTTP request.
- **rdp-cookie(<name>)** - The RDP cookie <name> (or "mstshash" if omitted) will be looked up and hashed for each incoming TCP request.

The HAProxy has a richer set of load-balancing algorithms, however for our purposes, the **least-connected** algorithm seems to be the best fit, and it is supported by both nginx and HAProxy.

Because the nginx software provides both a HTTP cache and a load-balancer, we recommend using it for both purposes. Alternatively a MapProxy with HAProxy may be used, but it is generally easier to maintain a single piece of software than two separate pieces of software.

2.2.2 Deployment on the CERIT-SC Cloud

The CERIT-SC cloud contains a VM template and a corresponding disk image for the deployment of scalable web map service.

The VM template is intended for two types of deployment. The first type is in many instances providing parallelized execution of the MapServer software. The second type is intended to be run as a single instance providing a well-known URL entry point backed by a cache and a load-balancer distributing HTTP request to the parallelized MapServers. The image has preinstalled nginx, HAProxy, Apache, MapServer and Postgres client and server.

VM instances in CERIT-SC cloud get IP addresses assigned randomly when they are created from a pool of IP addresses reserved for the cloud. A VM template may specify if it wants a public IP address, a private IP address (in the range 10.x.x.x) or both. The public IP addresses are mapped in reverse DNS to names cloudXX.cerit-sc.cz where XX is a number, which are not easily remembered. If a more easily remembered host-name is needed, dynamic DNS registrations can be used to assign a preselected host-name.

2.3 Metadata

Micka, developed by HSRS, is used as a metadata catalogue for SDI4Apps. MICKA is a complex system for metadata management (metadata creation, editing, storing, etc.) used for building SDI or geoportal solutions. It contains tools for editing and management of metadata for spatial information, web services and other sources (documents, web sites, etc.). It includes online metadata search engine, portrayal of spatial information and download of spatial data to local computer.
MICKA fully complies with the ISO 19115 standard. It can be integrated with map applications and it is multilingual. The web catalogue service uses OGC specifications (standards).
MICKA is compatible with obligatory standards for European SDI building (INSPIRE). Therefore it is ready to be connected with other nodes of prepared networked metadata catalogues (its compatibility with pilot European geoportal is continuously being tested).

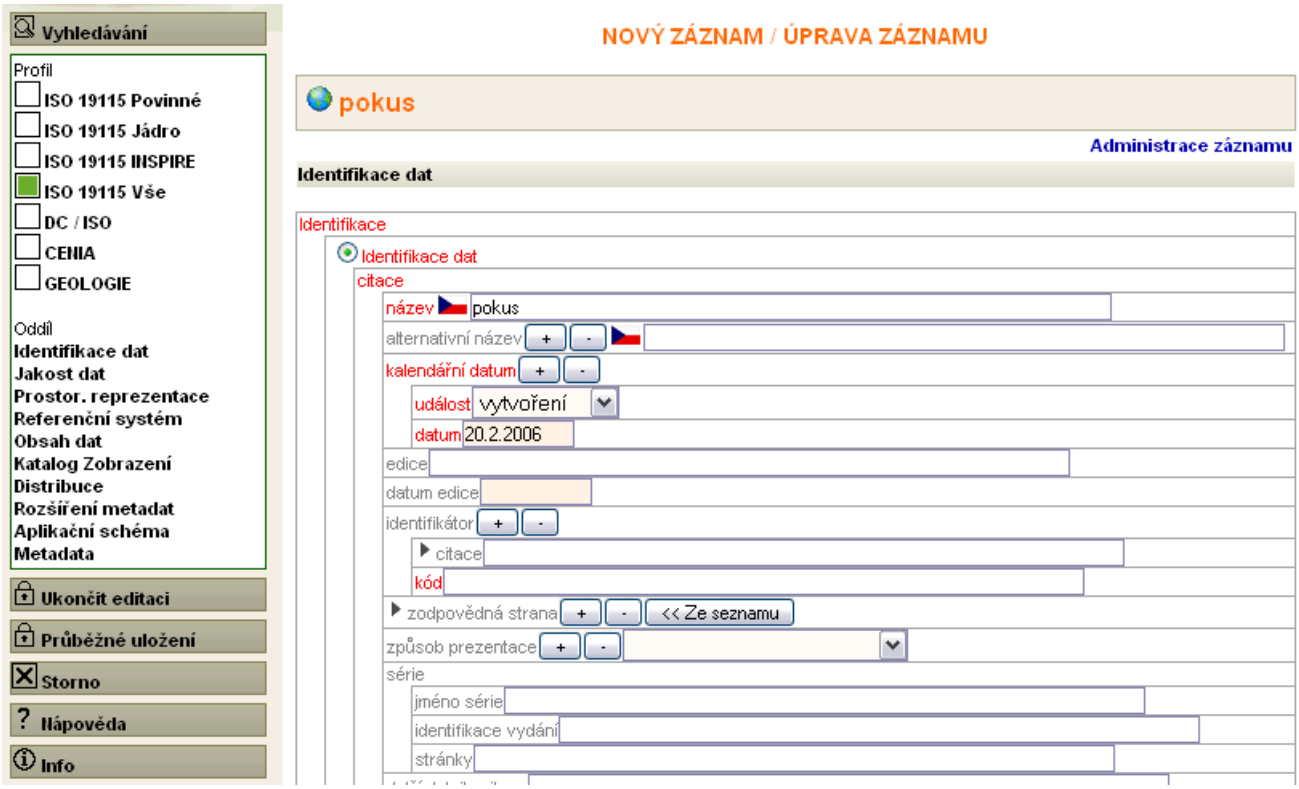

Figure 2 Metadata catalogue Micka

MICKA (Figure 2) is responsible for storing meta-information about available data and supporting discovery of existing geospatial data and services. Its role is very important for the platform. MICKA is a bridge to the current INSPIRE geoportal and services. MICKA supports internal management of geospatial data on the SD4Apps cloud. It helps to keep certain consistency of data. Since the metadata profile is compliant with international standards, it is much easier to search data. In addition, if metadata are stored as RDF then it is possible to query datasets from many sources (for example "show all datasets that were published by Eurostat", or more complicated query "show all datasets that where published in 2013 by Eurostat and are related to demographics").

Functions:

- Spatial data metadata (ISO 19115)
- Spatial services metadata (ISO 19119)
- Dublin Core metadata (ISO 15836)

- Feature catalogue support (ISO 19110)
- OGC CSW 2.0.2 support (catalogue service)
- User defined metadata profiles
- INSPIRE metadata profile
- Web interface for metadata editing
- Multilingual (both user interface and metadata records). Currently 16 languages supported. It is possible to dynamically extend the system for other languages.
- Context help (multilingual)
- Import from the following metadata formats are supported:
    - FGDC CSDGM,
    - ISO 19139,
    - OGC services (WMS, WFS, WCS, CSW)
    - Feature catalogue XML
- Export – ISO 19139, GeoRSS, HTML, PDF, JSON, GeoRSS, Atom, KML, OAI_PMH and OAI_MARC2, RDF
- Support of thesauri and gazetteers.
- Display of changes with GeoRSS
- Template base interface with possibilities to change according to user requirements
- Possibility of deep cooperation with any of map clients for display of on-line map services

The MICKA instance provides the following functionality:

- editing and validation of INSPIRE metadata,
- import metadata from existing services such as WMS, WFS, CSW, SOS and their editing and validation against the INSPIRE profile,
- advanced metadata search by several criteria,
- full discovery catalogue, supporting multi catalogue search.

A brief technical description of MICKA:

- Written in PHP (server part), ExtJS JavaScript framework (client part)
- Client components (may be used separately)
- Advanced search form – over all INSPIRE queryables. User required queryables can be added.

Configuration needed for initiation and runtime:

- Database setup and connection
- Shared libraries address
- OSM (or another map service) address
- Used thesauri address
- Visual (if not default)
- HTML output templates
- CSS, icons

Third party dependencies:

- PHP >= 5.2,
- Extensions: XSLT, mbstring

MICKA offers an API:

- OGC CSW 2.0.2 support according to standard
    - Querying (GetRecords, GetRecordById operations)
    - Editing (Transaction, Harvest operations)
- Extended functionality:
    - Additional queryables (listed in Capabilities document)
    - Additional format (JSON, RDF, Atom, GeoRSS, HTML)

2.4 Visualisation

Two visualisation clients HSLayers and HSLayers NG were deployed for the purposes of SDI4Apps.

2.4.1 HSLayers

Data visualisation is based on the current HSLayers product with extended functionality. HSLayers combines capabilities of ExtJS and OpenLayers and several helping scripts to establish truly Web GIS applications. HSLayers features include

- Portrayal of various types of data:
  - Raster: OGC WMS(-T), Image (PNG, JPEG, GIF), WCS
  - Vector: OGC WFS(-T), GML, GeoRSS, KML, GPX, GeoJSON, ML, WMTS, WCTS, OpenLS, GeoRSS, GML, INSPIRE View Service,
  - INSPIRE Direct Access Download Service,
  - INSPIRE Coordinate Transformation Service (WPS-CT).
  - Data sources from commercial servers: Google Maps, Virtual Earth, Yahoo Maps, …
  - The user interface adheres to current conventions in web map portals.
  - Information about queried objects in text bubbles.
- Dynamic adding of OGC (Open Geospatial Consortium) services into map - clients for WMS and WFS.
- Portrayal of independent data sources on the client side. Thematic maps can be composed on the basis of requests to various servers. It is thus not necessary to install a map server.
- Saving of thematic maps according to the WMC (Web Map Context) OGC specification on user's computer for repeated future use or for sharing between users.
- Extension of compute functions based on WPS (Web Processing Service) OGC service - according to user needs.
- Multilingual environment.
- Map requests to various types of data stored on various servers, with automatic processing of results.
- Work with micro-formats.
- Search on the map.
- Connection of the application with catalogue client (OGC CSW) in the geoportal, which enables display of the searched service from catalogue directly on the map.
- Edit function - snapping to chosen layers.
- Possibilities for advanced configuration of user requests.
- Advanced measuring of length and surfaces.
- Print of map compositions - possibility of large print outs (up to A0 format), user configuration of print settings.

An important new issue is support of Web Map Context (WMC). Web Map Context (WMC) describes how to save a map view comprised of many different layers from different Web servers. The expectation is to extend the WMC concept to allow storage of not only visualisation parameters, but also full analytical functionality of clients.

It consists of a map window, tree-based layer switcher, OWS client and other components including length and area measurement tools, module for printing of hard-copy maps and query tool. This whole environment enables the user to collect data from various data sources (WMS, KML, WFS, WMTS and others) and to prepare thematic maps. The maps can be styled using the SLD styler component. Layers can be made visible, transparent and their order within the layer stack can be changed.

2.4.2 HSLayers Next Generation (NG)

HSLayers NG is a new generation of HSLayers. It extends the OpenLayers 3 functionality and takes some ideas from the original HSLayers library, but doesn't use Ext3 as the frontend JavaScript framework and is more lightweight in general. It is still under development and published under the GNU/GPL licence version 3.

Dependecies

OpenLayers 3

OpenLayers is an open source (provided under the 2-clause BSD License) JavaScript library for displaying map data in web browsers. It provides an API for building rich web-based geographic applications similar to Google Maps and Bing Maps. The 3rd version of OpenLayers focuses on mobile devices and uses the newest HTML5 features for faster rendering of map tiles and vectors. It is also smaller in size reducing the loading times of applications.

RequireJS

RequireJS is used to separate HSLayers NG code into several modules. The module dependencies are automatically resolved and modules are loaded only when necessary improving the speed and quality of code.

AngularJS

HTML is great for declaring static documents, but it falters when using it for declaring dynamic views in web-applications. AngularJS enables extension of HTML vocabulary resulting in an environment which is extraordinarily expressive, readable, and quick to develop.

Bootstrap

Bootstrap is used for a unified UI appearance and responsive design so the application will look good on a big screen as well as on a mobile phone. Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

Design and Implementation

HSLayers is built in a modular way which enables the modules to be freely attached and removed as far as the dependencies for each of them are satisfied. The dependency checking is done automatically. The following paragraphs describe the main modules at the time of writing of this document.

Map

The map functionality is provided by OpenLayers3 and extended by some controls like navigation bar, scale line, attribution dialog, GPS and compass tracking etc. It supports multi-touch gestures, but the performance is highly dependent on the browser and mobile device hardware so can be a bit slower than in native applications.

Layer manager and legend

Layer manager is used for listing all the map layers, displaying or hiding them and setting the transparency. The user can view layers metadata and attribution by clicking on it. A legend is fetched from the server and displayed in a separate panel for all the WMS layers on the map. Grouping of layers in containers is also provided which enables a more user friendly and organized representation of layers for different topics.

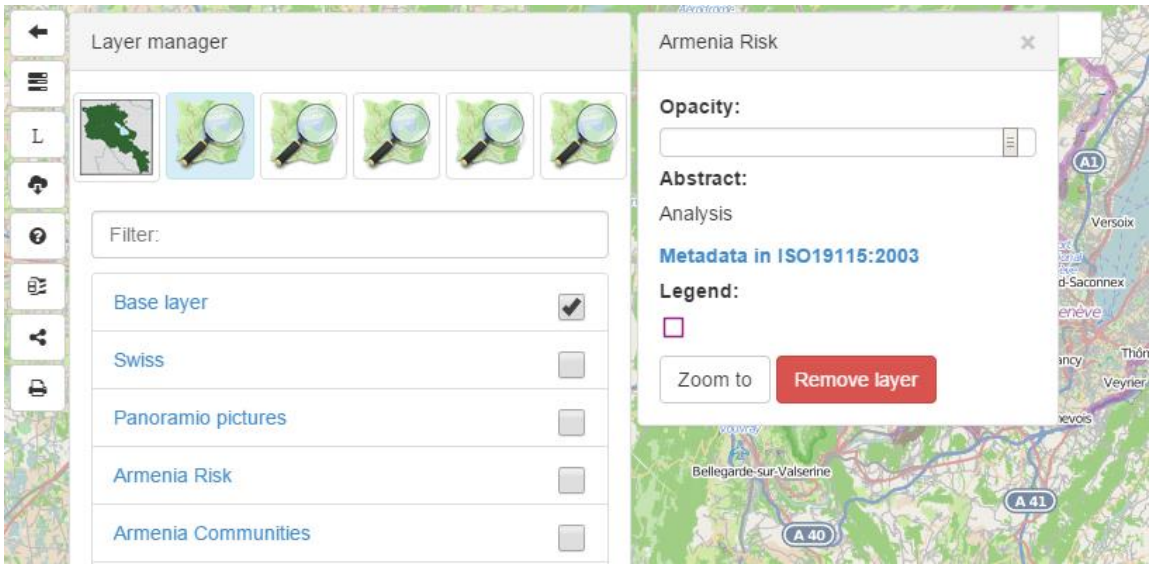

Figure 3 Layer manager

OGC Web Services context parser

The parser is used for GetCapabilities requests to different map servers and parsing the response. It can then be used for automatic or user initiated generation of map layers only by knowing the URL to the specific OGC standardized map service.

Query tool

The query tool generates a GetFeatureInfo request for every WMS layer on the map and displays the list of features and their attributes at the specified coordinate. For vector layers the attribute list is generated directly on client side without server interaction.

Search field

It provides a field for entering a name of a place and displaying a list of possible geographical names which begin with the phrase entered. Zooms the map to the place selected. It uses api.geonames.org service as the database, but in the future will be extended to different data sources.

Print dialog

Print dialog is used for printing the map with the users' browser print dialogue. The printing is done completely on client side by using HTML5 canvas graphics enabling a good performance. For it to work WMS server has to be configured to have Access-Control-Allow-Origin header (CORS support).

Permalink

Permalink provides the user with a URL which describes the current map state and view enabling the sharing and bookmarking of different map compositions. It also serves as a URL API when using HSLayers NG applications in an iframe or similar embedded environment.

Linked Open Data explorer

Eurostat explorer is a demo application (module) which queries Semantic Web data sources via SPARQL endpoints. It demonstrates the feasibility of automatic query building for Eurostat report data and displaying it on a map of NUTS2 regions (specified in GeoJSON file) according to the calculated transparency ratios.

On the server side it uses a Virtuoso Universal Server which is a middleware and database engine hybrid that combines the functionality of a traditional RDBMS, ORDBMS, virtual database, RDF, XML, free-text, web application server and file server functionality in a single system.

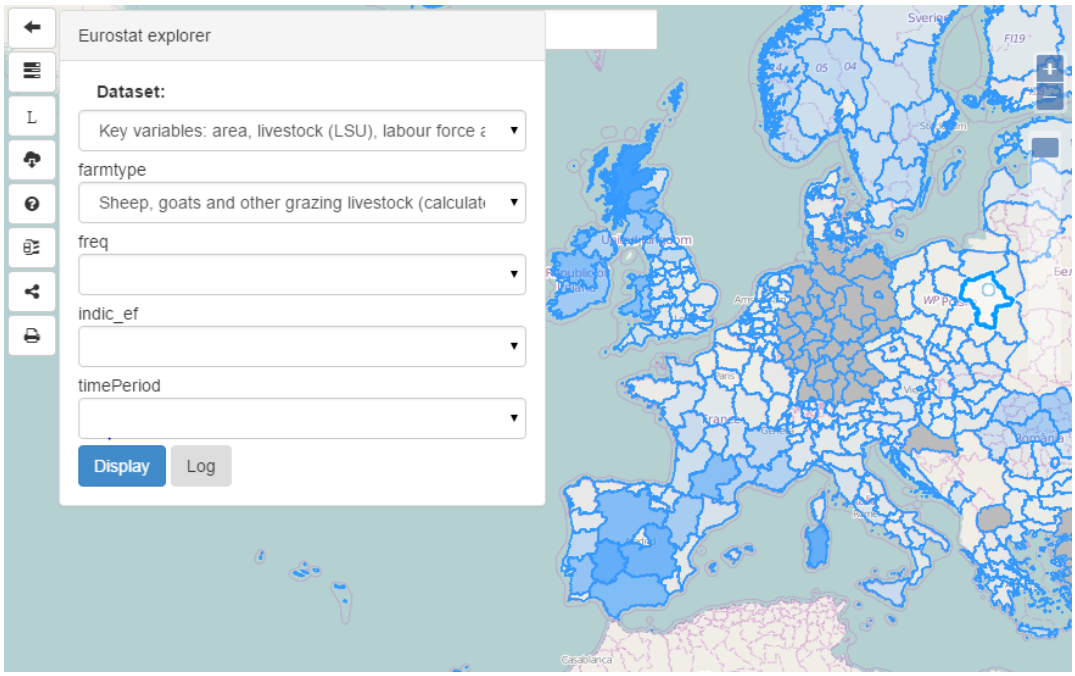

Figure 4 Linked Open Data explorer

Measurement tool

Provides the measurement of distances of polylines and areas of polygons in metric units.

Panoramio layer and info panel

Used mainly for touristic purposes this component creates a special layer which contains the thumbnails of scenic landscape photos from user generated and open database called "Panoramio". The photos are displayed in the place where they were taken. It uses the *Panoramios* API, to get the most popular images in the current map extent. The number of images returned is dependent on the screen size of the users' device.

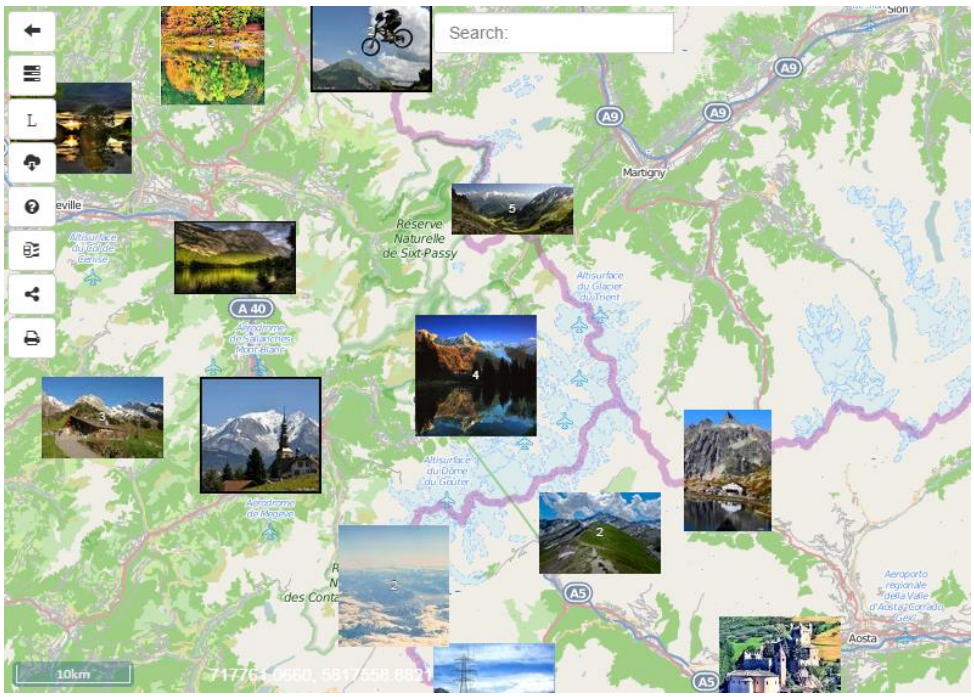

Figure 5 Panoramio layer and info panel

HSProxy

To overcome the cross domain request security limitation HSLayers NG uses a small CGI script serving as a proxy between the client map application and the server which sometimes has a different domain name.

2.5 Sensors

Two Enablers are used for sensors management:

- SensLog – a software developed by the University of West Bohemia and the Czech Centre for Science and Society
- Orion – Generic Enabler of FI-WARE[7]

2.5.1 SensLog

SensLog is a complex software framework for sensor networks. This integrated solution includes a data model and a server-side application which is capable to store, analyse and publish data in various ways over the network.

The main tasks of SensLog are:

1. receiving measured data,
2. store received data properly,
3. pre-process stored data for easier queries,
4. publish stored data through the system of web-services for various clients.

SensLog interface consists of web-services in proprietary form in JSON language and of standardised web-services using OGC SOS version 1.0.0.

SensLog is intended as a middleware between database and client applications with GUI. SensLog has extensive combinations of web services that are described below. Web GUI was designed to allow user to overview of measured data, provides list of units connected to any user. The web GUI allows to get unit position highlighted on map segment, to show measured values in form of charts for particular sensor connected to unit. The interface contains simple form to insert new unit positions.

List of units

URL:

/index.jsp

User gets list of available units according to the rights of access. The list of units contains list of numeric unit identifiers that are active links to pages with detailed unit description as well. Example of the list of units is shown below on Figure 6.

[7] http://catalogue.fi-ware.org/enablers/configuration-manager-orion-context-broker

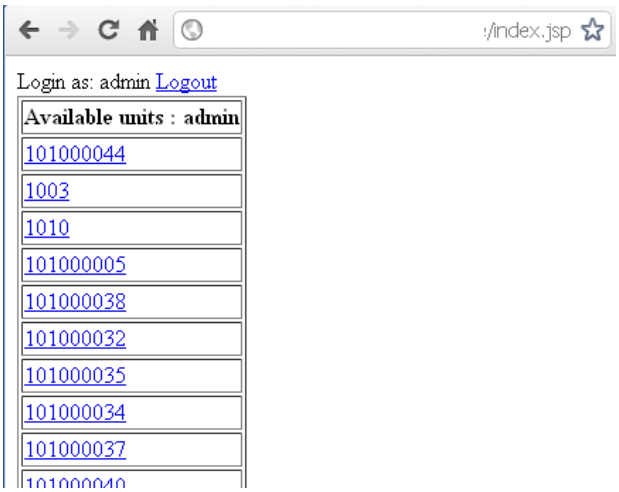

Figure 6 Example of the list of units

Visualisation of unit position and observations

URL:

/vypis.jsp?unit_id=*"unit_id"*

Parameters:

unit_id - numeric identifier of the units is mandatory parameter

The page with detailed information of the unit contains map segment with highlighted last position of the unit and several charts, one for every sensor connected to the unit. Every chart contains measured values during last 7 days. Access to this page is allowed by clicking on link in list of units described above or direct by entering URL with unit identifier parameter. Figure 7 contains part of visualisation page with map segment with red pointer highlighting last unit position. There is unit numeric identifier and short description of unit below the map segment. Charts for all connected sensors are below the map segment. Every chart contain unit and sensor identifier on top, time is on horizontal axis, observed values are on vertical axis of chart. Legend with sensor name is located under the chart.

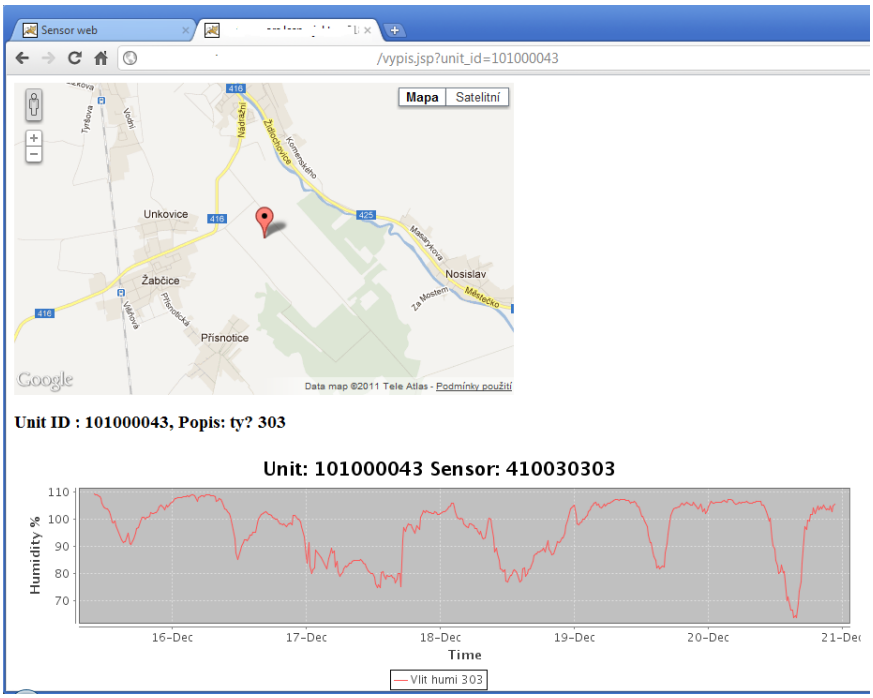

Figure 7 Example of unit details page

Insert new unit position

URL:

insert.jsp?unit_id=*"unit_id"*

Parameters:

unit_id - numeric identifier of the units is mandatory parameter

Authorized user is allowed to update unit position. There is page with simple form to insert new unit position, an example of page is shown on Fig. 7 below. The form in Figure 8 contains fields for inserting new unit position in WGS-84 system.

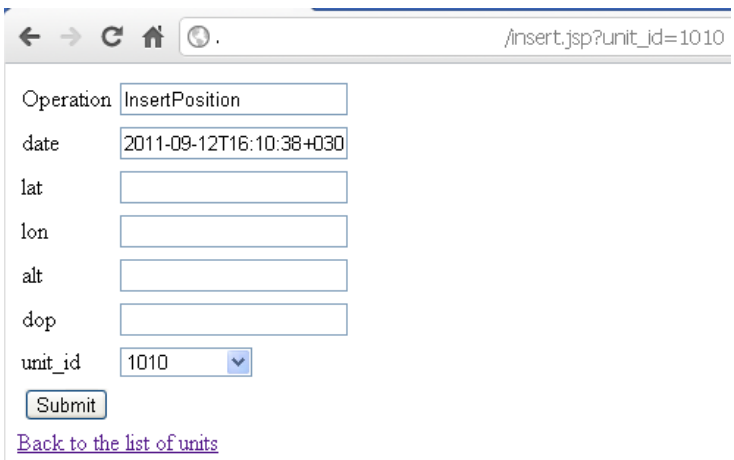

Figure 8 Example of insert unit position page

The fields of the form (Figure 8):

- *Operation* - Operation name remains unchanged "InsertPosition"

- *Date* - Time stamp of measured position, default value is the timestamp of page loading, user can change value but must respect the ISO 8601 pattern *YYYY-MM-DDThh:mm:ss±ZZZZ*.
- *Lat* - Latitude
- *Lon* - Longitude
- *Alt* - Altitude, optional
- *Dop* - Dilution of precision, optional

Unit_id - Unit numeric identifier, default value is from URL parameter, or can be selected from list.

Information about successful or unsuccessful inserting is returned to user after sending the form and then is user redirected to visualisation page of unit where should be map segment with new inserted position shown.

Standardised interface realised by the OGC SOS service is suitable for connection from third party clients. An example of the connection from SOS client to the SensLog is shown in Figure 9.

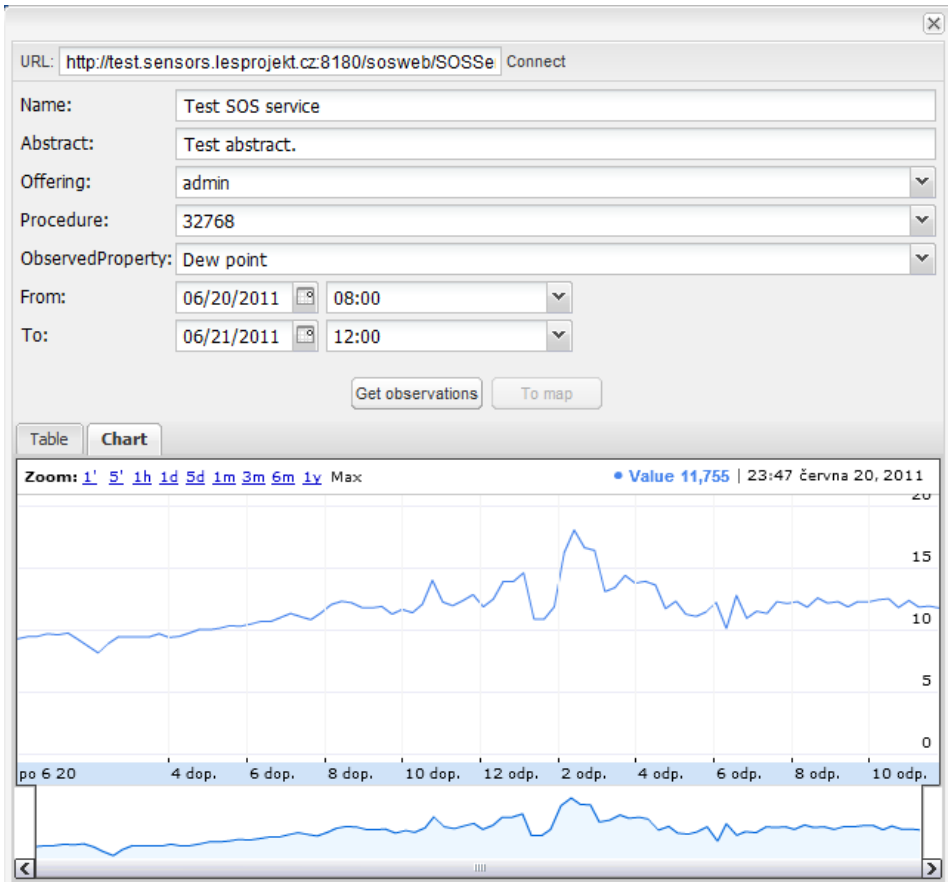

Figure 9 Example of client with connected SOS service

SensLog dependencies are solved by using Apache Maven that is a software project management and comprehension tool (http://maven.apache.org/). All required libraries are downloaded during compilation of the SensLog. SensLog runs in any web server and servlet container (e.g. Apache Tomcat http://tomcat.apache.org/) where can be deployed directly by Maven.

2.5.2 Orion Context Broker

The Orion Context Broker (Orion) is an implementation of the Configuration Manager GE. The Orion provides the NGSI 9 interfaces and within the Internet of Things (IoT) chapter, it is aimed to be used in combination with IoT Broker GE although Orion can be also used as a standalone component. The Orion actually supports both NGSI10 and NGSI9.

Using the NGSI9 interface clients can do several operations:

- Register context producer applications
- Discover context producers information
- Being notified when changes on context information availability

Implementation of the Orion uses MongoDB as main data storage and users don't access database directly. The Orion runs as a backend service daemon. Thus, it doesn't have any Graphical User Interface (GUI) and it is accessed through its REST API. The Orion REST API uses methods of HTTP and requests and responses in XML or JSON format (since version 0.9.0).

The Orion supports several groups of operations:

1. NGSI10 standard operations – these operations are derived directly from OMA NGSI specifications and allows:
    a. Update Context – entity creation
    b. Query Context
    c. Subscribe Context
    d. Update Context Subscription
    e. Unsubscribe Context
2. NGSI10 convenience operations – convenience operations are a set of operations that have been defined by FI-WARE project to ease the usage of NGSI implementations as a complement to the standard operations defined in the OMA NGSI specification. The convenience operations are an alternative to some standard operations. These operations allows:
    a. Entity creation
    b. Query context
    c. Update context
    d. Context subscriptions
3. NGSI9 standard operations – these operations are directly derived from the OMA NGSI specification and allows:
    a. Register Context
    b. Discover Context Availability
    c. Subscribe Context Availability
    d. Update Context Availability Subscription
    e. Unsubscribe Context Availability
4. NGSI9 convenience operations – described as part of the FI-WARE NGSI REST API NGSI9. The convenience operations are an alternative to some standard operations. These operations allows:
    a. Register Context
    b. Discover Context Availability
    c. Subscribe Context Availability
    d. Update Context Availability Subscription
    e. Unsubscribe Context Availability

Entity

An object with several attributes called an entity in the context of the Orion. The identification of each entity is provided by a unique elementId. The type of the entity is defined by element type which defined a class of the entity (e.g. an entity has id "1001" and type has "meteostation"). Each entity contains list of attributes elements and each attribute provides its name, type and value. The Orion doesn't perform any checking on data types. Apart from simple values for attribute values, it can be used complex structures or custom metadata. An example of real object can be meteorological station with 2 sensors, one for temperature and one for pressure. These station will be represented as an entity as follows in JSON format:

```
{"contextElements": [{
        "type": "meteostation",
        "isPattern": "false",
        "id": "1001",
```

```
            "attributes": [{
                "name": "temperature",
                "type": "float",
                "value": "23"
            }, {
                "name": "pressure",
                "type": "integer",
                "value": "720"
            }
    ] } ] }
```

An entity can be updated by several operations, it can be added other attributes or updated attributes values. Users can attach their own metadata to entity attributes. These metadata elements are processed by the Orion in a transparent way: it simply stores them in the database at updateContext and retrieve it in queryContext or notifyContext operations. An example of the metadata element follows:

```
    "metadatas": [{
        "name": "accuracy",
        "type": "float",
        "value": "0.8"
      } ]
```

Entities can have a location, specified by one of its attributes. The attribute that defines the location contains the "location" metadata. An example of entity with location follows:

```
    { "contextElements": [ {
            "type": "meteostation",
            "isPattern": "false",
            "id": "1001",
            "attributes": [ {
                "name": "temperature",
                "type": "float",
                "value": "23"
            }, {
                "name": "pressure",
                "type": "integer",
                "value": "720"
            }, {
                "name": "position",
                "type": "coords",
                "value": "40.418889, -3.691944",
                "metadatas": [
                {
                    "name": "location",
                    "type": "string",
                    "value": "WGS84"
                }
            ]}]}]}
```

It can be used different attributes to specify the location in different entities, e.g. "position", "coordinates" attribute. But only one attribute at a time can be defined as location. Current version only support WGS84 (which is the one used internally by the MongoDB database).

Query Context

Consumer of context has several operations to access the context information stored by Orion. Query context contains identifier of the entity and list of attributes that values should be returned. Instead of using exact entity identifier it can be used a regular expression for the entity identification and types of entities. Context can be queried either on entity types or on attributes names, but not on attribute values as such.

Notification

The Orion has ability to subscribe to context information if a defined event occurs. In this case the Orion send an asynchronous notification to user-defined application. There are two types of events when the Orion sends notification, it's called ONTIMEINTERVAL and ONCHANGE subscriptions. Each subscription is defined by:

- List of identification of entities and attributes which elements will be included in the notification message. It can be used lists of patterns to specify entities.
- The callback URL in reference element where notification should be sent. Only one reference can be included per subscribeContext request, but several subscriptions on the same context elements can be defined.
- Each subscription have a duration element and once that duration is expired, the subscription is simply ignored. The duration of the subscription can be extended by updating it. The duration is specified by ISO 8601 format.
- The trigger for the subscription is defined by element notifyCondition where child element type defines type of subscription and child element condValueList defines conditions of the triggering. In the case of ONTIMEINTERVAL condValueList contain only one element condValue whose values is a time interval defined by ISO 8601. A notification is sent with a frequency equal to that defined interval. In the case of ONCHANGE, notification is send when some attribute changes and the condValueList contains an actual list of condValue elements, each one with an attribute name.

Subscriptions can be updated or cancelled by several operations by given subscriptionId that is given in response to create subscription request.

Filters

The Orion implements several filters that can be used to filter the results in query operations. These filters are typically used with queryContext with patterns or the convenience operation to get all entities. There are implemented several filters:

- Existence type filter – in the current version, the only parameter than can be checked for existence is the entity type.
- No-Existence type filter – in the current version, the only parameter than can be checked for no-existence is the entity type
- Entity type filter – only entity with given type will be selected in queryContext
- Geo-location filter – filter defines the area and either only the entities located in that area or entities not included in the area are taken into account. Orion allows the following possibilities:
    - Area internal to a circumference, given its centre and radius.
    - Area external to a circumference, given its centre and radius.
    - Area internal to a polygon (e.g. a terrain zone, a city district), given its vertices.
    - Area external to a polygon (e.g. a terrain zone, a city district), given its vertices.
    - Area unions or intersections (e.g. the intersection of a circle and a polygon) are not supported in the current version.

2.6 Initial Pilot Deployment Test

In order to test the potential of the CERIT-SC cloud as well as related procedures from the pilot point of view, initial activities took place on behalf of the Slovakian Ecosystem Services Evaluation (ESS) pilot.

This set of test activities included registration into the cloud environment with provision of appropriate credentials via user account management. Based on an initial set of ESS pilot requirements an appropriate cloud environment has been used based on the Debian Linux operating system with a selection of the following spatial data infrastructure-related software components:

- GeoServer
- Mapshup

As input resources for the pilot deployment test, imagery data resources (GeoTIFF) were used. These resources represent outcomes of the data processing activities focused on evaluation of selected ecosystem services (forest biomass, tourism, carbon, livestock) as well as overall evaluation of ESS at national level.

GeoServer (Figure 10) provides functionality for the publication of the data resources as Web Map Service (WMS) as well as Web Coverage Service (WCS)[8]. A set of available WMS encodings including the OpenLayers and KML formats are available via Layer preview interface[9]. The Web coverage services can be accessed via web and desktop clients[10,11].

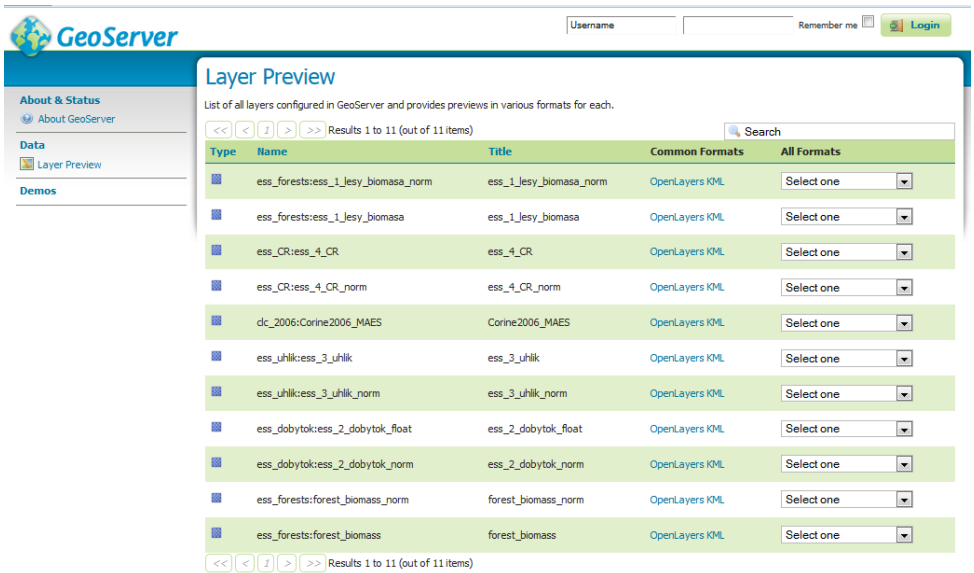

Figure 10: List of published ESS related data resources

Mapshup (Figure 11) acts as a web client allowing the visualisation of the published data resources as well as their combination with other resources via one single web graphical user interface. A dedicated project for this initial set of the data resources has also been created[12].

[8] http://www.opengeospatial.org/standards/wcs
[9] http://147.251.252.167:8080/geoserver/web/?wicket:bookmarkablePage=:org.geoserver.web.demo.MapPreviewPage
[10] http://147.251.252.167:8080/geoserver/ows?service=wcs&version=1.1.1&request=GetCapabilities
[11] http://147.251.252.167:8080/geoserver/wcs
[12] http://147.251.252.201/mapshup

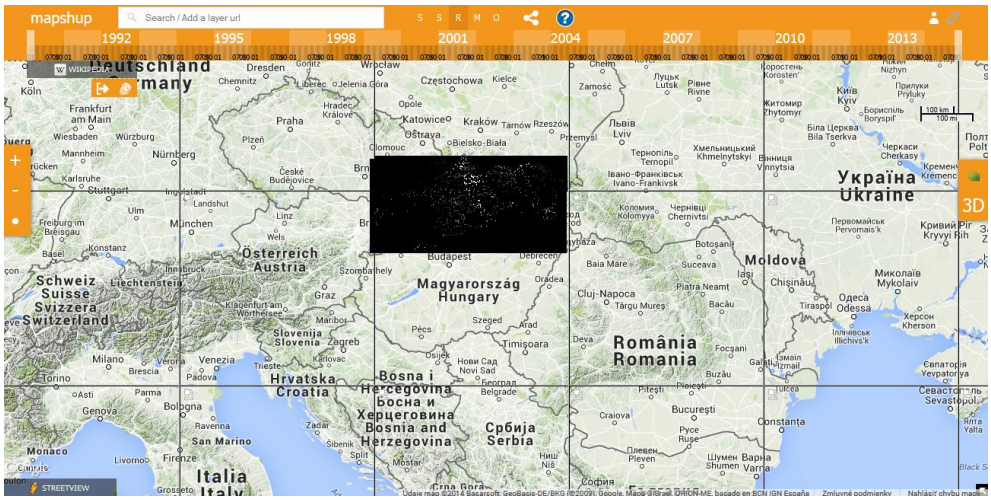

Figure 11 Mashup

This set of initial pilot-related test activities helped the project partners to get familiar with the cloud environment and directly test the deployment possibilities of generic and specific enablers for specific pilot requirements.

Future activities will be focused on exploration of the possibilities of cloud scalability with GeoServer / Map server Web GIS server; storing coverages in Postgres XL clustered GIS database as well as deployment of further specific enablers (supporting web-processing and semantic tools).

3 CONCLUSION

The initial deployment of generic and specific enablers for the SDI4Apps platform on the CERIT-SC cloud described in this document is based on the current understanding of pilot application needs, and will be refined over time as the pilot application requirements get better specified following the iterative development methodology adopted by the SDI4Apps project.

REFERENCES

FIWARE Catalogue - Configuration Manager - Orion Context Broker. URL <http://catalogue.fi-ware.org/enablers/configuration-manager-orion-context-broker>

Publish/Subscribe Broker - Orion Context Broker - Installation and Administration Guide. URL <https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Publish/Subscribe_Broker_-_Orion_Context_Broker_-_Installation_and_Administration_Guide>