

ENABLERS DEPLOYMENT

- FINAL RELEASE

SEPTEMBER 2016



DELIVERABLE

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

D.3.2. ENABLERS DEPLOYMENT - FINAL RELEASE

Revision no. 04

Authors: Karel Charvat (CCSS)
 Runar Bergheim (Avinet)
 Marin Kuba (MU)
 Jan Jezek (UWB)
 Michal Kepka (UWB)
 Dmitrij Kozuch (HSRS)
 Stepan Kafka (HSRS)
 Simon Leitgeb (HSRS)
 Alfredo Iembo (Hyper)
 Daniele Tarini (Hyper)

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	30/12/2015	Karel Charvat	CCSS	Initial draft
02	20/03/201	Karel Charvat, Runar Bergheim, Marin Kuba, Jan Jezek, Michal Kepka, Dmitrij Kozuch, Stepan Kafka, Simon Leitgeb, Alfredo Iembo, Daniele Tarini	CCSS, BOSC, MU, UWB, Avinet	Contribution
03	31/03/2016	Karel Charvat	CCSS	Revision
04	30/09/2016	Daniele Tarini	HYPER	Final check and minor text revisions

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This deliverable contains original unpublished work except where clearly indicated otherwise.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

Revision History	3
Table of Contents	4
List of Figures.....	5
Executive Summary	6
1 Introduction	7
2 Basic Cloud Enablers Update	8
3 Specific Enablers for SDI Domain Update	9
4 Enablers for Mobility and Sensors Update	10
4.1 Mobile Thematic Maps.....	10
4.2 SensLog	12
4.2.1 SensLog and VGI	14
4.3 IoT Discovery.....	16
4.4 VGI module.....	19
5 s4a.js	21
5.1 Deployment and installation	21
5.2 Dependencies	21
5.3 Architecture	22
5.4 Main features	23
6 References to Platform	24
7 GitHub script	27
8 Conclusion	29

LIST OF FIGURES

Figure 1 - Mobile Thematic Map	11
Figure 2 - Google Play Web Page for Mobile Thematic Maps.....	12
Figure 3 - Overview diagram of SensLog structure	13
Figure 4 - Example of current observed values of selected unit	13
Figure 5 - Example of 7-day history of air pressure sensor of selected unit.....	13
Figure 6 - Register form example	17
Figure 7 - Update form example.....	18
Figure 8 - Query form example	19
Figure 9 - VGI App.....	20
Figure 10 - the above diagram shows and explains the dependencies between s4a.js and other components	22
Figure 11 - The above diagram shows the principal packages of the s4a.js library	23
Figure 12 - Map viewer	24
Figure 13 - Sensors viewer	25
Figure 14 - Spatial and semantic query builder	26
Figure 15 - Example of script output	28

EXECUTIVE SUMMARY

The report describes the final version of the SDI4Apps portal and its installation as Virtual server. The current version including all components is now available to be installed in any cloud environment. The main focus during the last year was on improving the functionality of HSLayers NG and transfer it to mobile environment, improving of SensLog and support for voluntary collected data.

1 INTRODUCTION

During the third year, the enablers integrated into one operational platform were updated and improved. This platform is now available on Masaryk University cloud (portal.sdi4apps.eu). It is now also available a virtual server, which could be easily deployed to any cloud. This opens possibilities for different users to run their own platform.

2 BASIC CLOUD ENABLERS UPDATE

The basic cloud enablers deployment has been upgraded according to the deliverable D3.4.1 "Updated Platform Release - Y2".

3 SPECIFIC ENABLERS FOR SDI DOMAIN UPDATE

The specific enablers deployment has been upgraded according to the deliverable D3.4.1 "Updated Platform Release - Y2".

4 ENABLERS FOR MOBILITY AND SENSORS UPDATE

4.1 Mobile Thematic Maps

Mobile Thematic Map is a result of the HS-Layers framework and mobile specific code integration. It utilises PhoneGap framework, which packs HTML, CSS3 and JavaScript code into an installation package. Apache Cordova is the software developer uses to compile the application with PhoneGap. Because PhoneGap is based on rendering the application via web view, not the platforms UI framework, the source code does not have to be platform specific and can still benefit from the native device APIs. As the consequence of employing web views, performance and other issues may arise.

One such issue is the fact that Android devices prior to V4.4.3 KitKat uses the default Android browser to render the web view, which does not provide GPU acceleration and restricts CPU support if the application calls for GPU acceleration. This results in severe performance decrease and has to be dealt with. Subsequent Android versions use Chromium for web view rendering, which does support GPU acceleration. The Crosswalk webview plug-in for Cordova solves this issue by incorporating Chromium directly in the application package. That increases the package size by approximately 20 MB, which could be undesirable.

Another matter addressed was usage of proxy service by HS-Layers. The service could not be utilised in the application, because it is written in Python, which is unsupported by Cordova by default. Cordova instead uses domain whitelisting for various types of HTTP requests (images, stylesheets, scripts, etc.), so a simple URL may be used for search queries and JSON or image requests if whitelisted properly. HS-Layers components that make HTTP requests needed to be modified so they make a request with simple URL in the mobile version, but still serve the URL to the proxy service on desktop. A JavaScript mobile variable was defined in the index.html file of the Cordova application and a conditional statement in each component verifies if this variable is defined.

Geolocation component also needed to be rewritten to use native geolocation API via the Cordova Geolocation plug-in. This allows for usage of high precision GPS service. A GPS logging functionality was also introduced as a part of the mobile geolocation component. It employs a WebSQL database to store location information (available values are longitude, latitude, altitude, horizontal accuracy, current velocity and heading). Another storage options are also available, including local file storage or other database types. This logging functionality can be extended to display various statistics about created GPS tracks or the tracks themselves as features on the map. Displaying velocity and altitude as well as centring on the user's current position functions as intended.

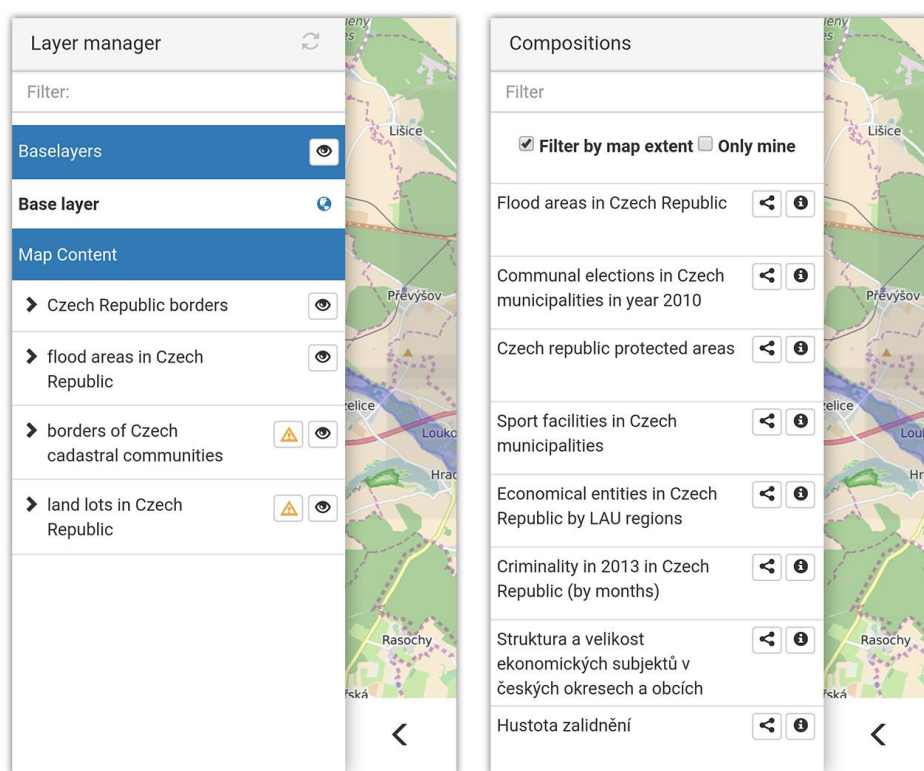


Figure 1 - Mobile Thematic Map

Mobile specific layout was dealt with mainly at the Dresden 2015 hackathon. A CSS stylesheet is used to redefine properties of relevant classes. The mobile toolbar is proposed to be located on the bottom of the screen and to contain only four buttons, namely Layer manager, Search, Geolocation and one specific to each example. This results in a simpler layout that is easier to use on a mobile device than the desktop layout.

This is first solution available on Google Play, at the following URL:

<https://play.google.com/store/apps/details?id=com.sdi4apps.thematicmaps>

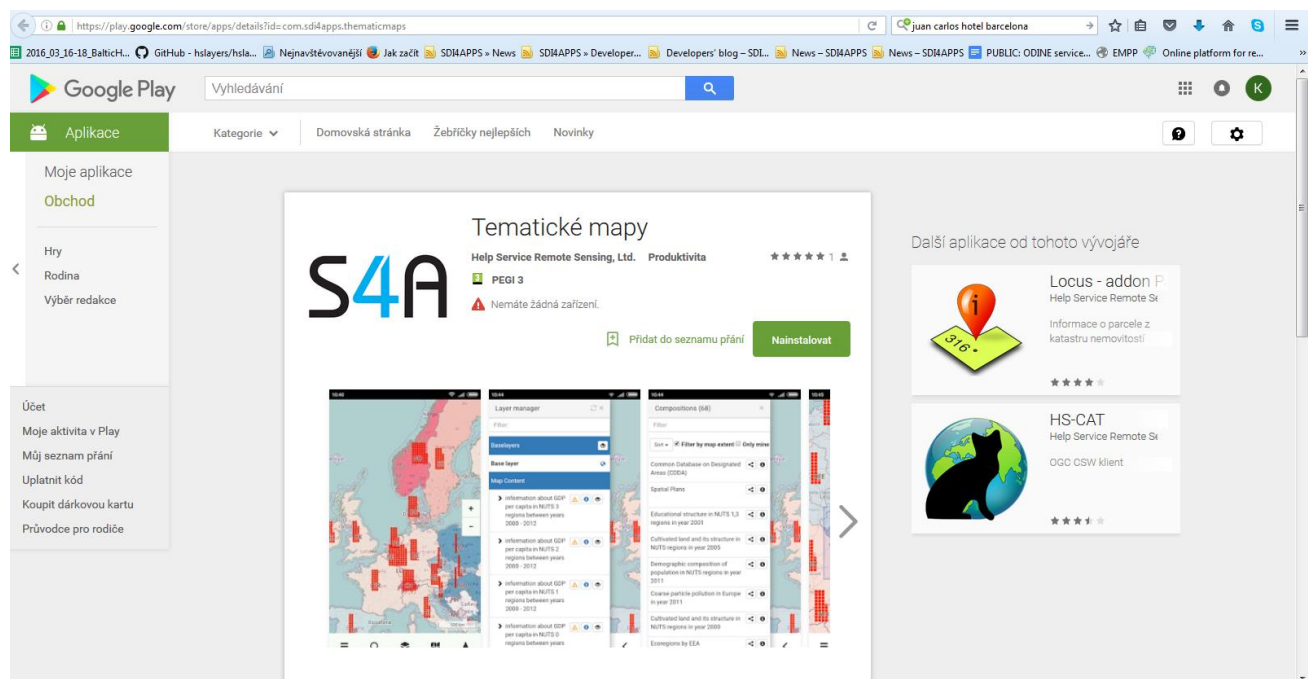


Figure 2 - Google Play Web Page for Mobile Thematic Maps

4.2 SensLog

SensLog is a software component for sensor data collecting, storing and publishing in the web. SensLog consists of database model and server-side application. Relational database model is implemented in PostgreSQL database system with PostGIS extension. Database model contains procedures and functions to process and manipulate with data in transactional form. Especially several triggers prevents data integrity. Database model was developed as universal model both for mobile and static sensors. It can be used for storing data from mobile devices (e.g. in scenario Human-as-Sensor).

Server-side part of the SensLog contains most of the application logic. Diagram on figure 4.1 shows simple overview schema of the SensLog. Sensor data are pushed to SensLog from sensor networks (in fields or smart devices) through RESTful API. Receiver module contains check mechanism to prevent insertion of data in incorrect form. Data consumers can receive data by one of the interfaces after authentication. There are several services to provide not only measured data or status data of the network but also structure of sensor network itself.

Users can use Web GUI that allows visualization of measured data in form of simple charts and maps. Main access to sensor data is provided by RESTful or SOS interface and it is expected further processing in other clients or applications.

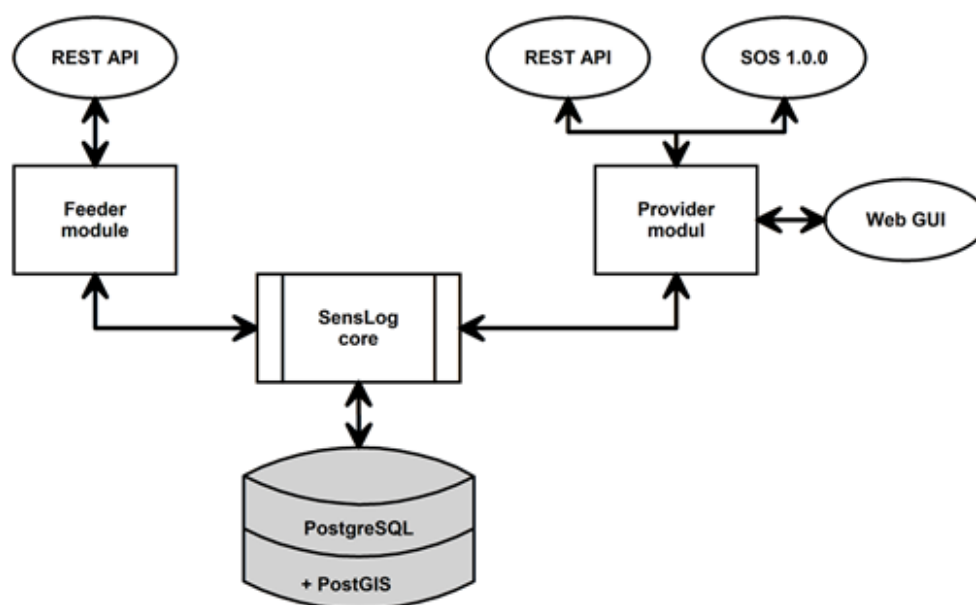


Figure 3 - Overview diagram of SensLog structure

SensLog contains simple GUI for visualization of measured data on the Web. Web GUI contains visualization of unit positions and visualization of current values in sensor network and history of values for selected time span for specified sensor.

Teplota vzduchu	Vlhkost vzduchu	Rosný bod	Tlak vzduchu	Rychlost větru (min)	Rychlost větru (a)	Rychlost větru (max)	Směr větru (min)	Směr větru (a)	Směr větru (max)	Děšť akumulace	Děšť intenzita	Děšť trvání	Kroupy akumulace	Kroupy intenzita	Kroupy trvání
2.3 °C	65.400002 %	-3.132891 °C	971.900024 Pa	0.4 km/h	1.5 km/h	1.8 km/h	248 °	303 °	277 °	0 mm	0 mm/h	0 s	0 zás/cm ²	0 zás/cm ² /h	0 s
11.02.2016 21:30:02	11.02.2016 18:50:02	11.02.2016 20:45:02	11.02.2016 20:00:02	11.02.2016 19:50:02	11.02.2016 21:45:02	11.02.2016 18:20:02	11.02.2016 21:25:02	11.02.2016 19:15:01	11.02.2016 19:30:02	11.02.2016 21:25:02	11.02.2016 20:45:02	11.02.2016 21:40:03	11.02.2016 21:00:04	11.02.2016 21:40:03	11.02.2016 19:25:02

Počet dnů historie: 7 30 90 365

Figure 4 - Example of current observed values of selected unit

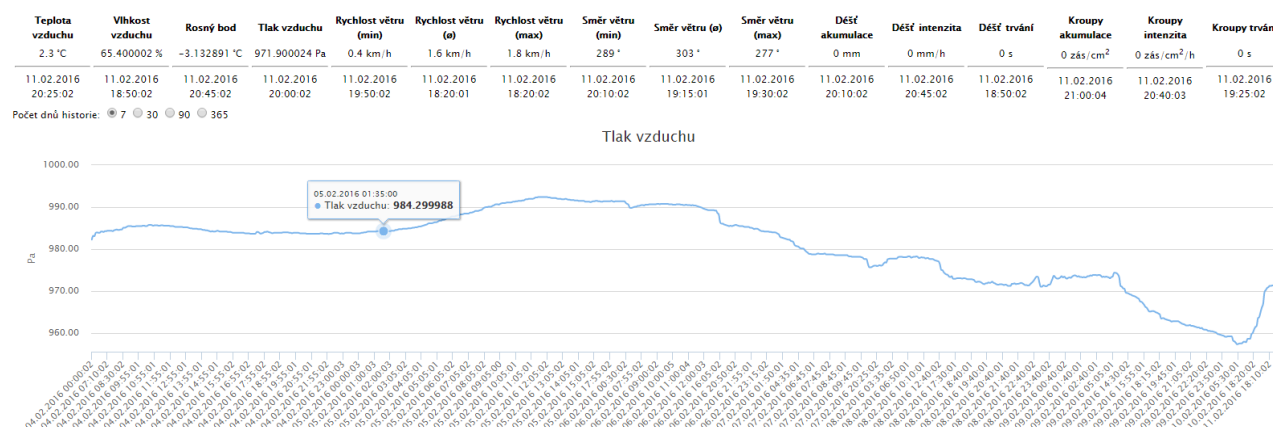


Figure 5 - Example of 7-day history of air pressure sensor of selected unit

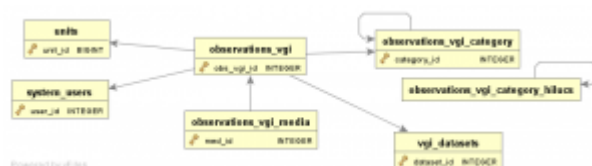
Important features of current version of SensLog are:

- Collecting sensor data from both mobile and static sensors
- Collecting observations in numerical data type, positions of sensors in WGS-84 coordinates, alert events occurred during measurement
- Storing data in relational database model, prepared for long time series by partitioning of most used tables observations and positions
- Publishing stored data in raw or preprocessed form, running predefined analyses on observations
- Publishing data in JSON format, by OGC SOS 1.0.0 service, or in form of chart visualization

4.2.1 SensLog and VGI

A new module for storing and publishing VGI was implemented to SensLog. Common type of VGI are Points of Interest (POI). The concept of POI is very similar to general observation concept. Implemented scenario suppose that collected POI is spatially and temporally localized observation with several attributes and connected multimedia files. POI is classified into one of predefined categories and can be assigned to some dataset.

The SensLog data model was extended. New tables are shown on Fig. 1 below.



Added tables are following:

- observations_vgi - table storing POI with all attributes
- observations_vgi_media - table storing media file connected to POI
- observations_vgi_category - table storing categories of POI, master table of categories
- observations_vgi_category_hilucs - inherited table with Land Use categories
- vgi_datasets - table storing metadata of dataset

Each POI has information about the device and the user that had created it. Categories of POIs are stored in tables inherited from the master table observations_vgi_category. This way allows storing of different kinds of hierarchical categories. Expandable schema of POI attributes is used. Only mandatory attributes are stored in separate columns, like timestamp, geometry, category, dataset etc. Optional attributes are stored in JSON format in separate column. Geometry of POIs and device metadata are stored in core part of SensLog data model.

Implemented service are described below. More services and functions will be added during next steps.

VGI Service

VGI service allows to insert new POI and update stored POIs, to select POIs by several filters and to select categories and datasets.

InsertVgiObservation

URL

rest/vgi/insobs		
Method	Functionality	Format
POST	Provides insertion or update of POI into database. Returns id of	plain-text

	inserted or updated POI.	
Parameter	Format	Role
timestamp	Textual value	Timestamp when was POI observed (ISO 8601 compatible), mandatory
category	Numerical value	Identifier of category, mandatory
description	Textual value	Free text description of POI, optional
attributes	Textual value	JSON formatted additional attributes, optional
dataset	Numerical value	Identifier of dataset, optional
unitId	Numerical value	Identifier of device producing POI, mandatory
lon	Numerical value	Longitude of POI, mandatory
lat	Numerical value	Latitude of POI, mandatory
media	File	Connected media file, optional
obs_vgi_id	Numerical value	ID of POI, if present method updates existing POI, optional

ObservationService

Select observations

URL		
rest/vgi/observations/select		
Method	Functionality	Format
GET	Provides stored POIs in JSON or GeoJSON format by selected filters.	JSON, GeoJSON
Parameter	Format	Role
user_name	Textual value	Name of user that collects POIs, only own user_name or user_name from same group allowed, mandatory
format	Textual value	Name of output format (geojson, json allowed), optional
fromTime	Textual value	Begin timestamp of frame, optional
toTime	Textual value	End timestamp of frame, optional

Media service

Select media

URL		
rest/vgi/media/select		
Method	Functionality	Format
GET	Provides stored POI media	depends on stored media format
Parameter	Format	Role
obs_id	Numerical value	ID of POI

Category service

Select category

URL		
rest/vgi/category/select		
Method	Functionality	Format
GET	Provides stored categories of POIS	JSON

Dataset service

Select dataset

URL		
rest/vgi/dataset/select		
Method	Functionality	Format
GET	Provides stored datasets of POIs	JSON

4.3 IoT Discovery

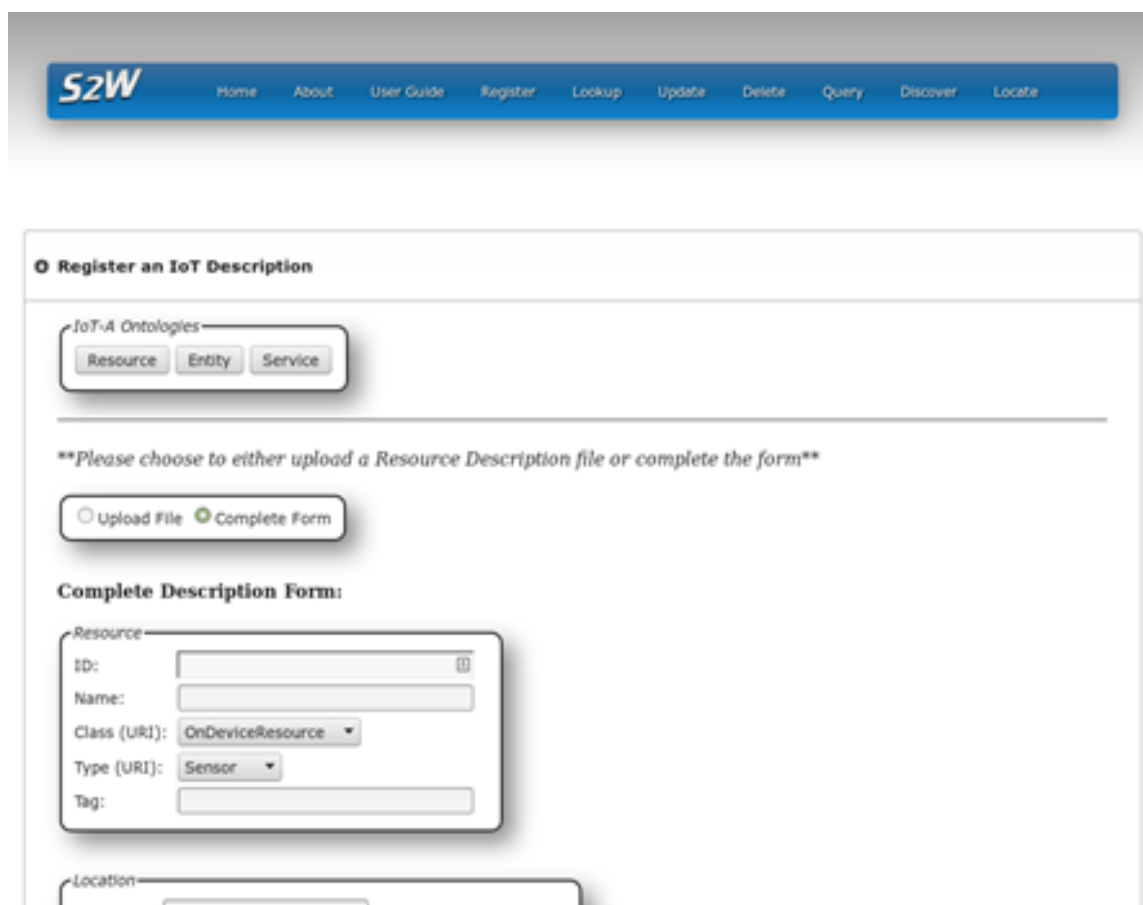
IoT Discovery is Generic Enabler from FIWARE group. The IoT Discovery allows discovery of IoT objects, by providing a repository to register Things, Resources, and Devices, using semantically-annotated Descriptions based on the Internet of Things Architecture ontology models. The IoT Discovery uses Sense2Web IoT Linked Data Platform, which provides a repository for the CRUD (Create, Read, Update and Delete) management of semantic IoT descriptions, that complies with the IoT-A ontology models. Sense2Web can also associate different IoT object ontologies to domain data and other resources on the Web using Linked Open Data.

The IoT Discovery provides a set of interfaces a user can interact with. The first is a Web User Interface whereby a user can perform CRUD operations on the IoT Descriptions, and also query the IoT Descriptions as well. When registering or updating, a user can either upload an IoT Description or complete a form which is then sent to the server to be converted to RDF, and storing it in the RDF database.

The second interface is a RESTful CRUD and SPARQL interface. This interface mainly supports M2M interactions. An application can also perform CRUD operations on the IoT descriptions in the repository, and query for a particular piece of information from the descriptions using SPARQL.

Register

The current ontologies that are supported are the IoT-A ontologies that define a Resource, Entity and Service. Registering can be done either by uploading a description file or by completing a form. The form will not be accepted unless an ID, Name and Latitude/Longitude coordinates are at least entered. Once the form is submitted the page will return links for viewing the RDF result of the submission in various formats i.e. RDF/XML, RDF/XML-ABBREV, RDF/JSON, N3, N-TRIPLE and TURTLE.

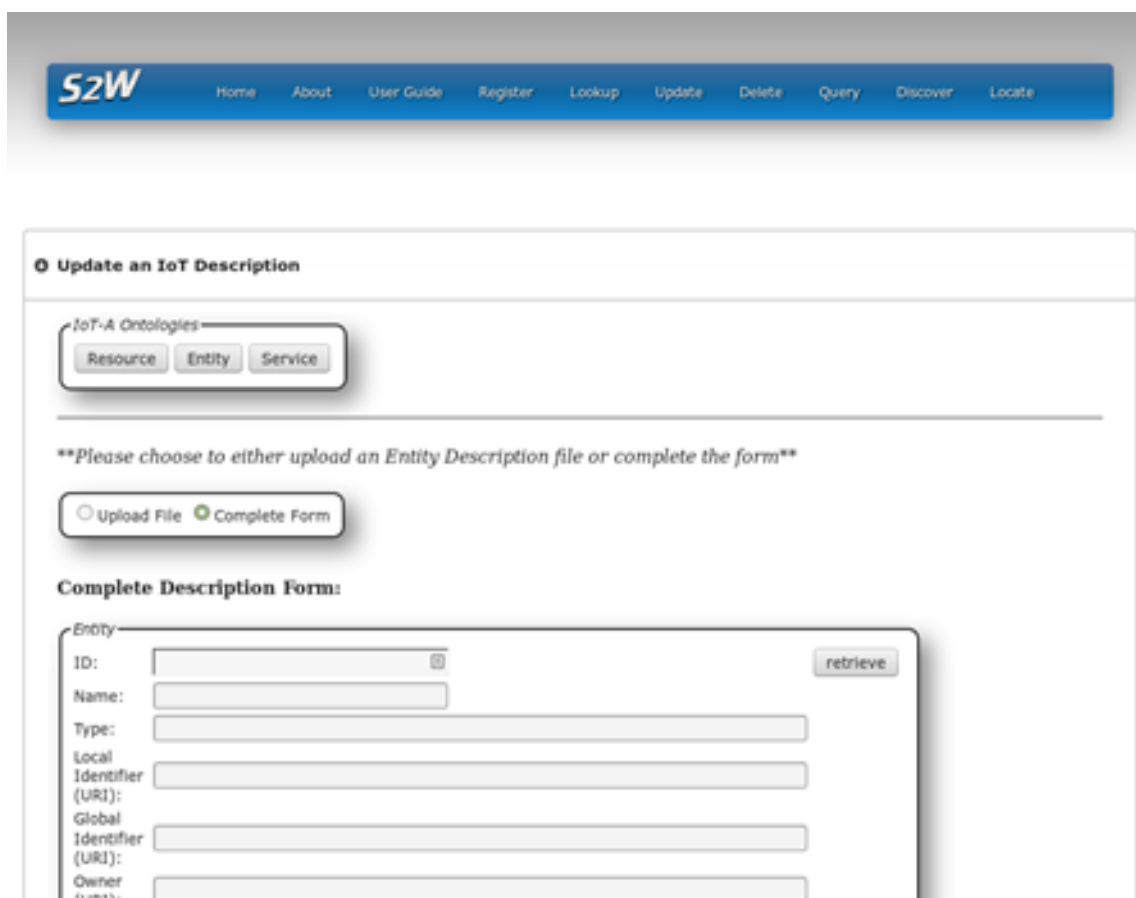


The screenshot shows a web application interface for S2W. At the top is a blue navigation bar with the S2W logo and links: Home, About, User Guide, Register, Lookup, Update, Delete, Query, Discover, and Locate. Below this is a section titled "Register an IoT Description". Inside this section, there is a sub-section "IoT-A Ontologies" with three buttons: Resource, Entity, and Service. Below these buttons is a message: "**Please choose to either upload a Resource Description file or complete the form**". Under this message are two radio buttons: "Upload File" (unselected) and "Complete Form" (selected). Below the radio buttons is a section titled "Complete Description Form:". This section contains a "Resource" sub-section with the following fields: "ID:" (text input), "Name:" (text input), "Class (URI):" (dropdown menu with "OnDeviceResource" selected), "Type (URI):" (dropdown menu with "Sensor" selected), and "Tag:" (text input). Below the "Resource" section is a "Location" sub-section with a text input field.

Figure 6 - Register form example

Update

The Update operation is similar to the Register with the exception that the current values for a description can be retrieved and populated into the fields by entering the ID of the description in question, and clicking on the "retrieve" button.

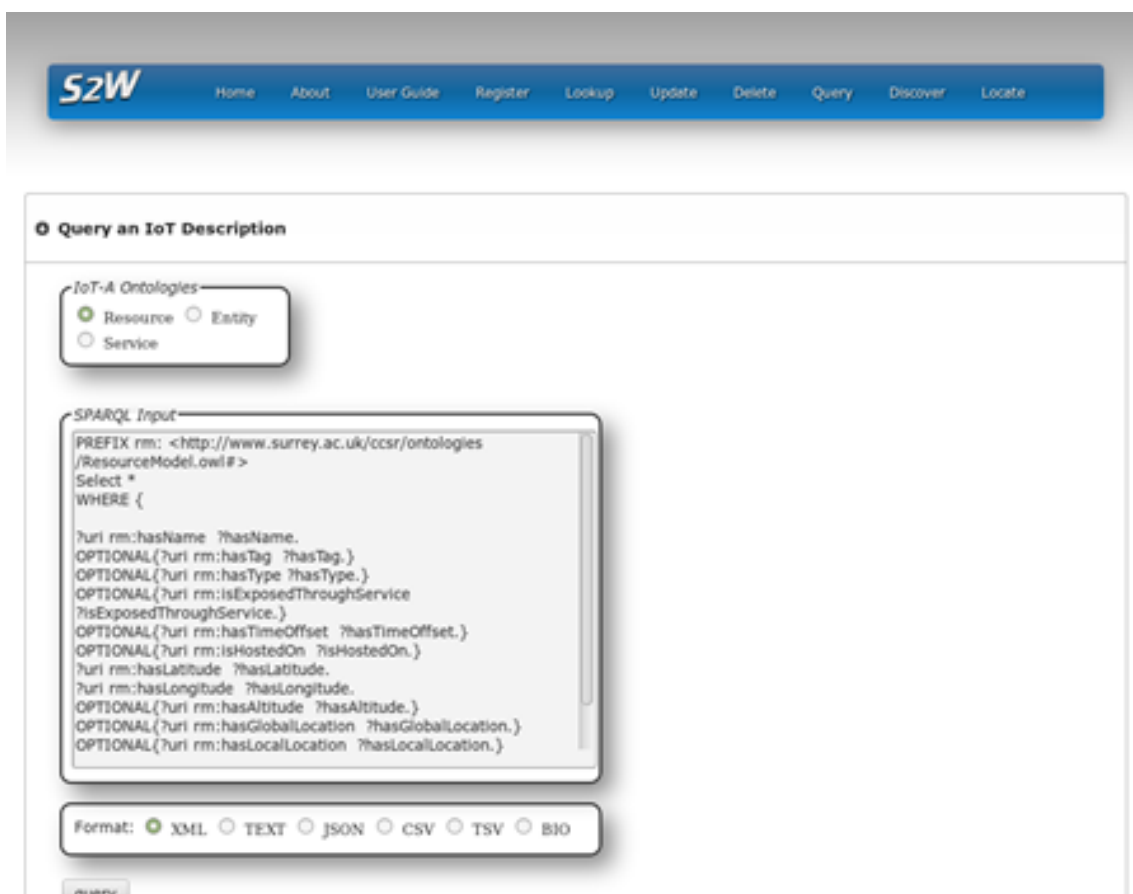


The screenshot shows the 'Update an IoT Description' interface. At the top is a blue navigation bar with the 'S2W' logo and links: Home, About, User Guide, Register, Lookup, Update, Delete, Query, Discover, and Locate. Below the navigation bar, the main content area is titled 'Update an IoT Description'. It features a section for 'IoT-A Ontologies' with three buttons: 'Resource', 'Entity', and 'Service'. A message states: '**Please choose to either upload an Entity Description file or complete the form**'. Below this, there are two radio buttons: 'Upload File' (unselected) and 'Complete Form' (selected). The 'Complete Description Form' section is titled 'Entity' and contains several input fields: 'ID:' (with a small icon), 'Name:', 'Type:', 'Local Identifier (URI):', 'Global Identifier (URI):', and 'Owner (URI):'. A 'retrieve' button is located to the right of the 'ID:' field.

Figure 7 - Update form example

Query

The platform supports SPARQL for querying of descriptions. When choosing a particular type of description, the respective SPARQL template is provided for a user to use and edit.



The screenshot shows the S2W web application interface. At the top is a navigation bar with links: Home, About, User Guide, Register, Lookup, Update, Delete, Query, Discover, and Locate. Below this is a section titled "Query an IoT Description". Inside this section, there is a "IoT-A Ontologies" group with three radio buttons: "Resource" (selected), "Entity", and "Service". Below the ontologies is a "SPARQL Input" text area containing the following query:

```
PREFIX rm: <http://www.surrey.ac.uk/ccsr/ontologies/ResourceModel.owl#>
Select *
WHERE {
  ?uri rm:hasName ?hasName.
  OPTIONAL{?uri rm:hasTag ?hasTag.}
  OPTIONAL{?uri rm:hasType ?hasType.}
  OPTIONAL{?uri rm:isExposedThroughService ?isExposedThroughService.}
  OPTIONAL{?uri rm:hasTimeOffset ?hasTimeOffset.}
  OPTIONAL{?uri rm:isHostedOn ?isHostedOn.}
  ?uri rm:hasLatitude ?hasLatitude.
  ?uri rm:hasLongitude ?hasLongitude.
  OPTIONAL{?uri rm:hasAltitude ?hasAltitude.}
  OPTIONAL{?uri rm:hasGlobalLocation ?hasGlobalLocation.}
  OPTIONAL{?uri rm:hasLocalLocation ?hasLocalLocation.}
}
```

Below the SPARQL input is a "Format:" group with radio buttons for XML (selected), TEXT, JSON, CSV, TSV, and BIO. At the bottom left of the form is a "Go" button.

Figure 8 - Query form example

Main features of the IoT Discovery enabler at this version are:

- Registering metadata of sensor data producers in RDF format or by form
- Querying and searching for registered sensors in the catalogue by GUI operations or by SPARQL queries

4.4 VGI module

VGI client use the some Cordoba based HSlayers NG technologies lime Mobile Thematic Maps, but there is added number of functions supporting geolocation, inserting photo, inserting text. It communicate with server trough SensLog VGI API.

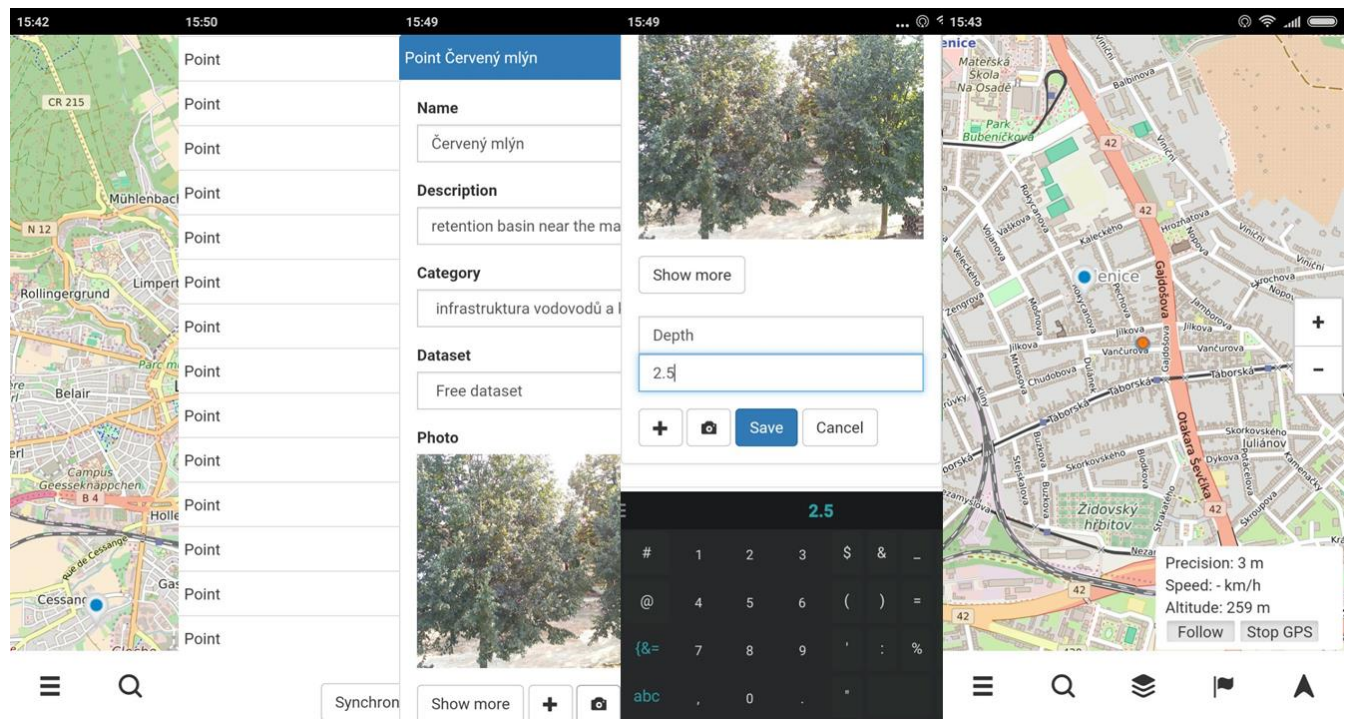


Figure 9 - VGI App

5 S4A.JS

SDI4Apps includes a lightweight API library for rapid application development in HTML and JavaScript. This library is called s4a.js.

5.1 Deployment and installation

s4a.js is deployed as part of the platform setup script. Any virtual server generated using that script will include the s4a.js script at a pre-defined path on the on-board web server, Apache.

s4a.js consists partly of functionality that relies on a running instance of the SDI4Apps platform and the OpenAPI, partly of stand-alone functionality. Thus, the library has also been deployed as a Bower package that can be installed in any JavaScript project using the simple command

```
bower install s4a.js
```

5.2 Dependencies

s4a.js relies on server side web services as well as parallel client-side JavaScript libraries that provide infrastructure functions.

The dependencies may be summarized as follows:

- Functions that rely on web services (mainly the OpenAPI and SensLog) include routing, storage, feature synchronization, tile download, search and custom storage services
- Most functions rely on jQuery
- Map related functions rely on OpenLayers3
- Graphing and diagramming functions rely on d3.js, crossfilter.js and dc.js
- Mobile related functions rely on the Apache Cordova framework for cross-platform app development.

All dependencies are Open Source libraries with permissive licenses.

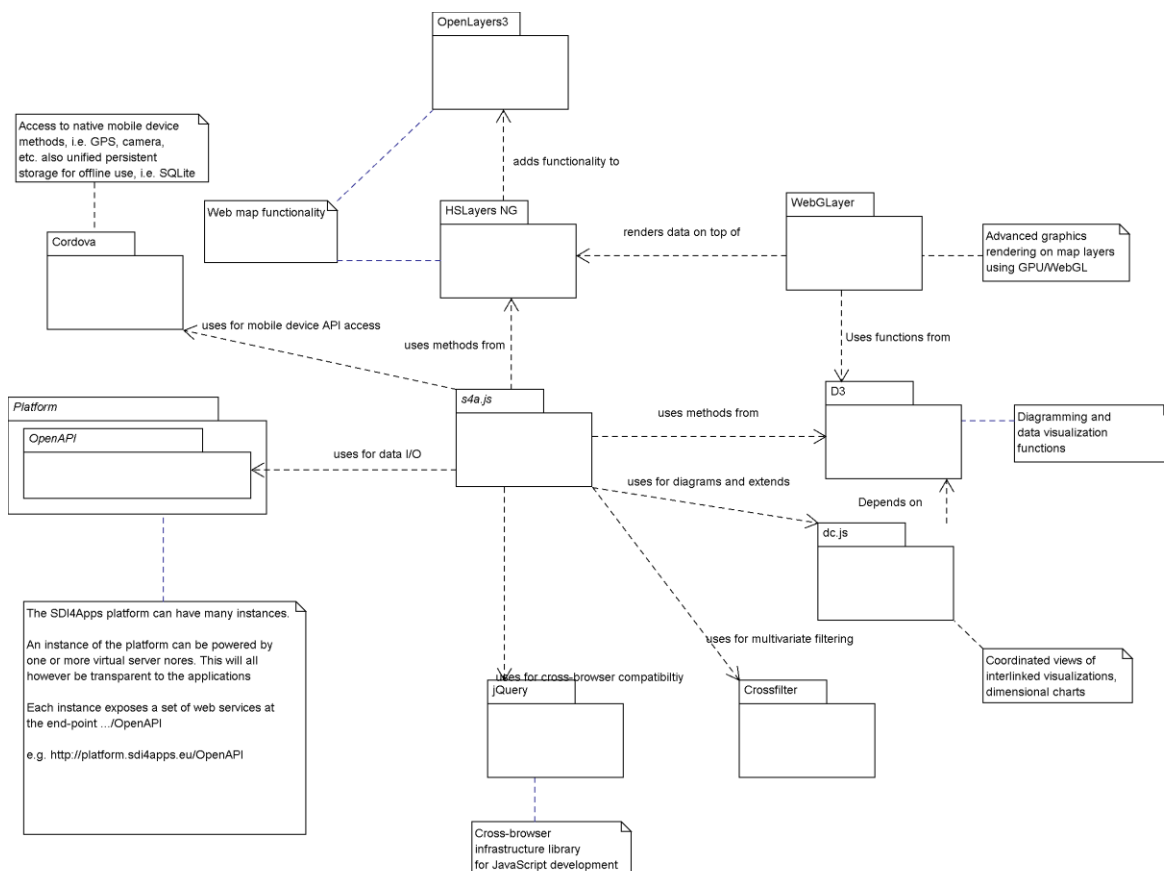


Figure 10 - the above diagram shows and explains the dependencies between s4a.js and other components

5.3 Architecture

The architecture of s4a.js is as a component in the presentation layer of SDI4Apps, calling and utilizing functionality that sits in the application layer -- and providing stand-alone functionality of its own. The latter mainly in the form of visualization functions.

The library makes use of the AJAX functions in jQuery for sending requests and receiving responses from the underlying web services (i.e. in the OpenAPI). The transport is however not XML but more lightweight and transport overhead friendly JSON.

Since the library may be used in third party web applications that are hosted on different servers, another architectural constraint is that the OpenAPI must permit CORS (cross origin resource sharing). By default, a script loaded from host <http://a.com> is only able to call other resources on that same domain. By enabling CORS on the server, applications may sit *anywhere* while the central functionality remains on a third party server, i.e. a centralized cloud instance of the SDI4Apps platform.

5.4 Main features

The detailed design of the library is provided in deliverable D4.2 - advanced tools API design. Thus, the high-level view UML package centric perspective is described here.

- A Map component provides productivity methods for quickly building OpenLayers based map interfaces
- An Information Retrieval component provides methods for searching and indexing data using the GeoDoc web services of the OpenAPI
- An Advanced Visualization component provides methods for visualization of numerical data as different types of charts and statistical maps with an emphasis on *coordinated views*.
- A Mobile component that provides access to functions for caching tiles offline as well as offline editing of vector data. This component extends Cordova.
- An Analytics and Modelling component that provides access to methods to invoke and produce outputs based on the ‘area suitability model’ conceived during the SDI4Apps MedHack event in Patras and later further developed by SDI4Apps partners.

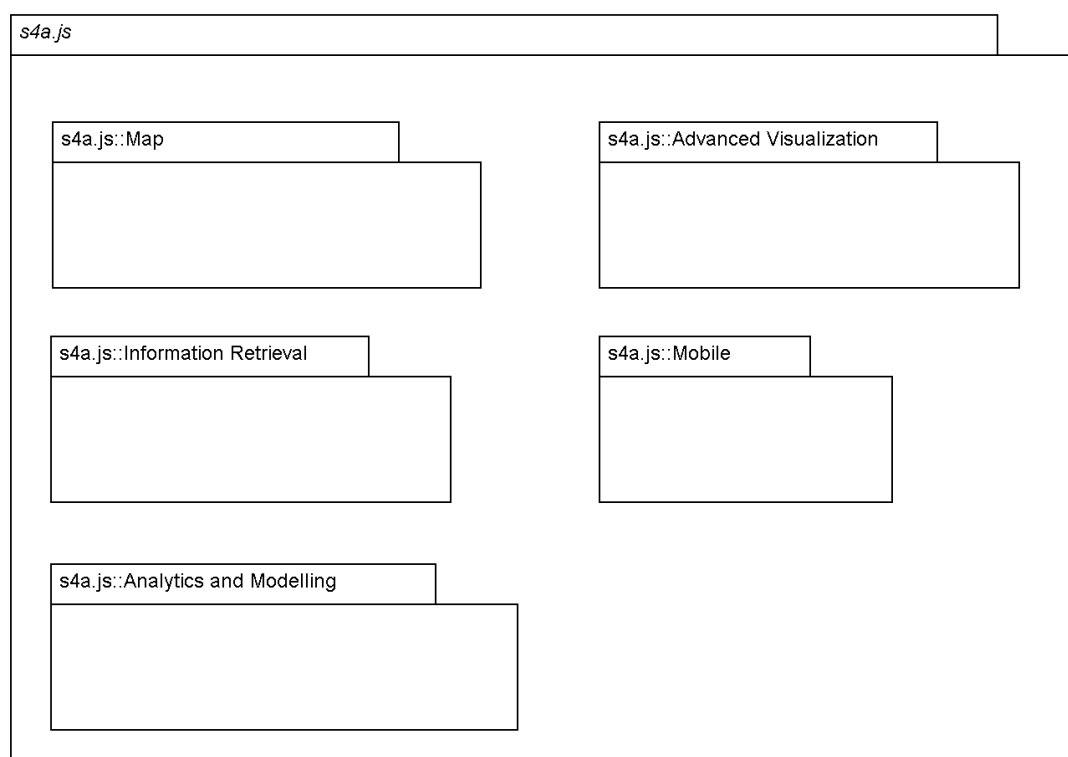


Figure 11 - The above diagram shows the principal packages of the s4a.js library

6 REFERENCES TO PLATFORM

A running instance of the platform, along with example client developed on platform API, services, enablers and data, is available at:

<http://portal.sdi4apps.eu/>

Example screenshots:

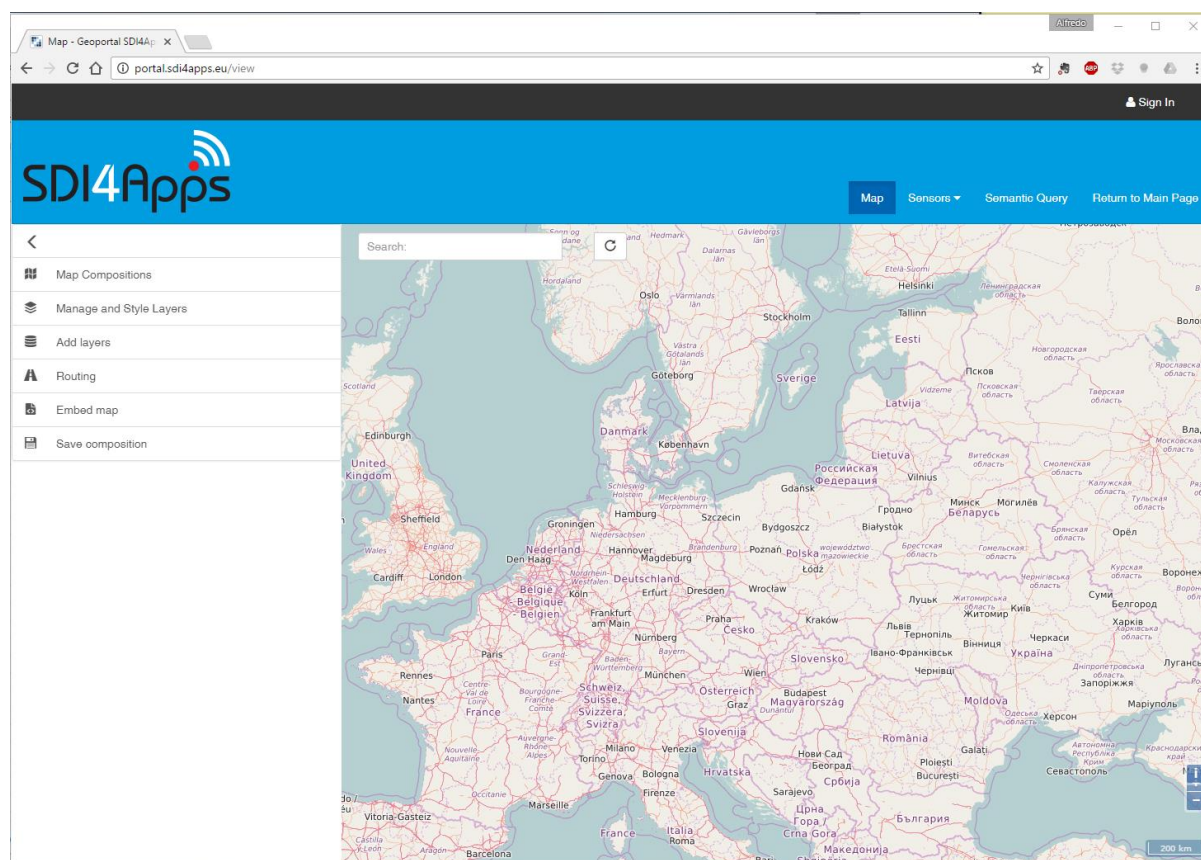


Figure 12 - Map viewer

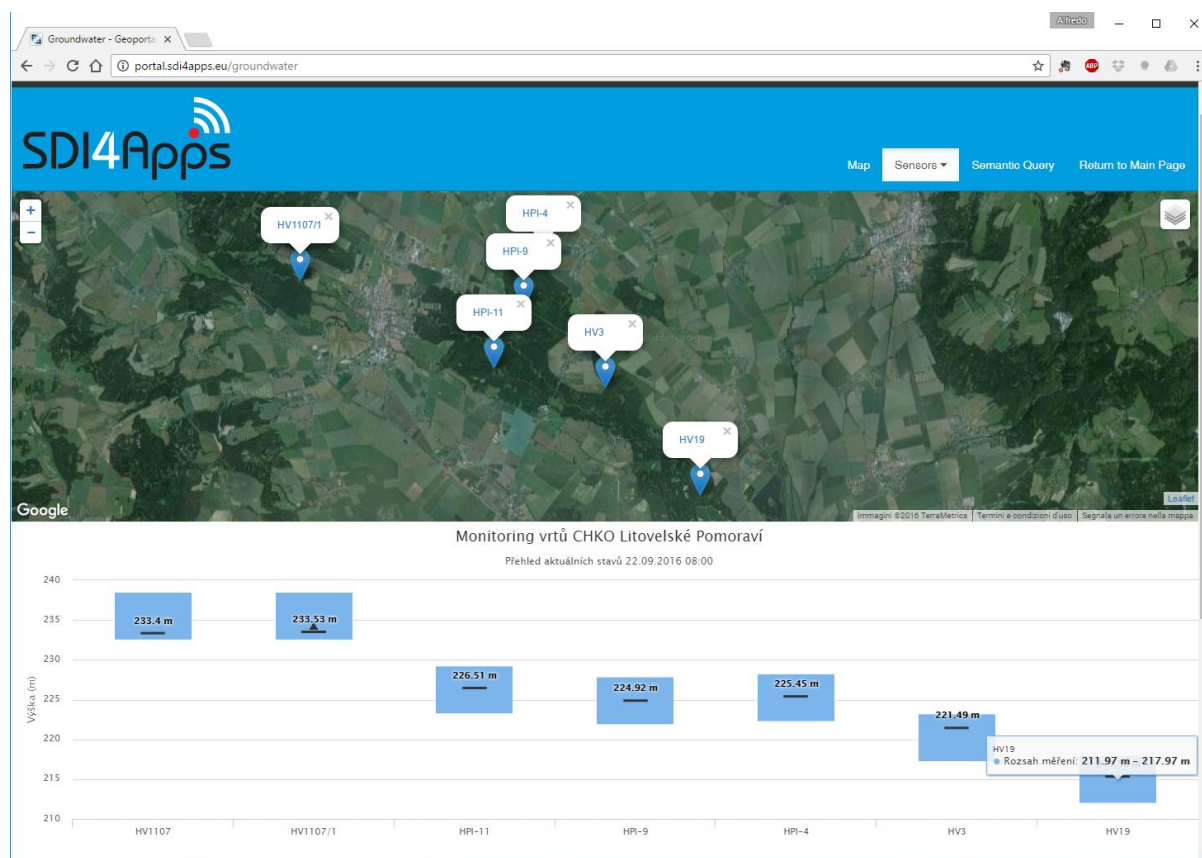
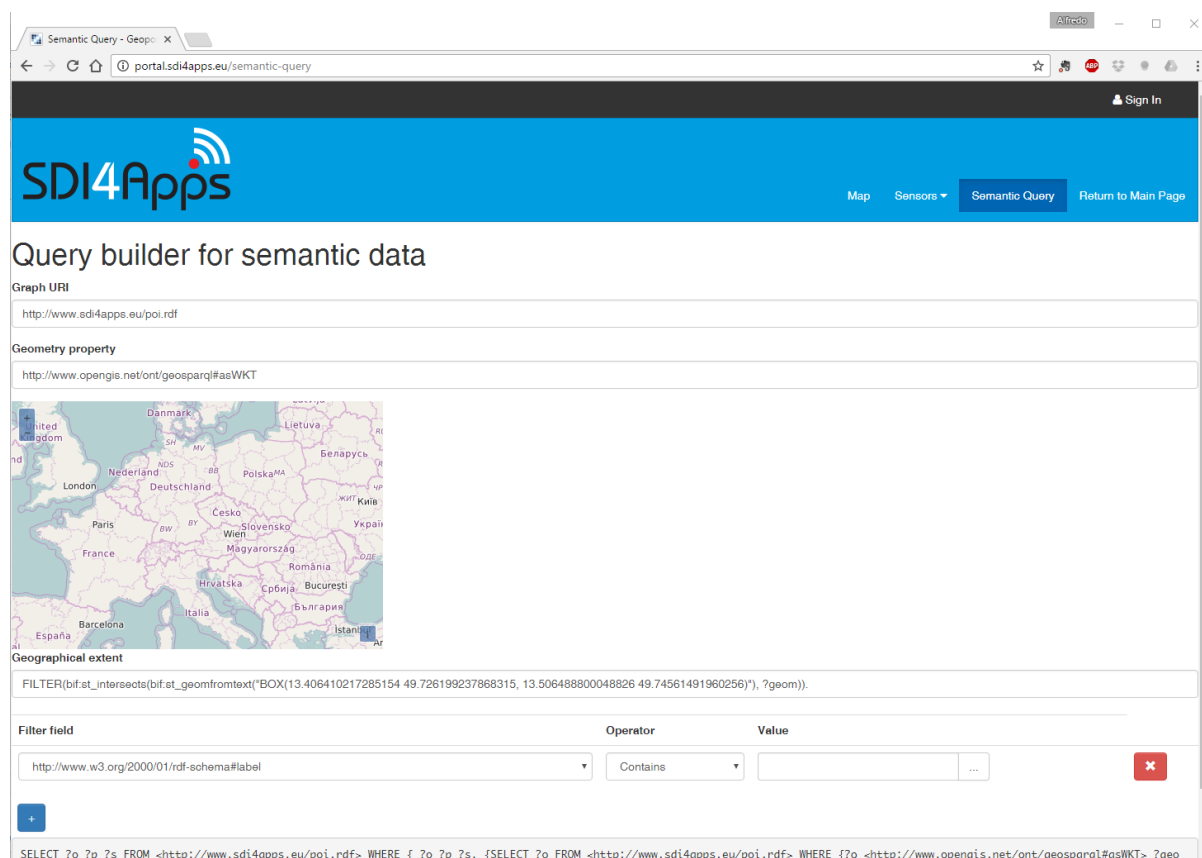


Figure 13 - Sensors viewer



Graph URI

http://www.sdi4apps.eu/poi.rdf

Geometry property

http://www.opengis.net/ont/geosparql#asWKT

Geographical extent

FILTER(bif:st_intersects(bif:st_geomfromtext("BOX(13.406410217285154 49.726199237868315, 13.506488800048826 49.74561491960256)"), ?geom)).

Filter field	Operator	Value
http://www.w3.org/2000/01/rdf-schema#label	Contains	

SELECT ?o ?p ?s FROM <http://www.sdi4apps.eu/poi.rdf> WHERE { ?o ?p ?s. {SELECT ?o FROM <http://www.sdi4apps.eu/poi.rdf> WHERE {?o <http://www.opengis.net/ont/geosparql#asWKT> ?geo

Figure 14 - Spatial and semantic query builder

7 GITHUB SCRIPT

As the **Deliverable D3.4.1 “Updated Platform Release - Y2”** describes in detail in its section 2 “Basic Cloud Enablers Update”, the SDI4Apps platform has the ability to be deployed into multiple Infrastructure-as-a-Service clouds in a uniform way. This ability is realized using a technique called **contextualization** - by modifying a base operating system during its first start in a virtual machine according to a provided list of needed modifications.

The SDI4Apps platform is defined in a configuration file **user-data.yaml** published in the GitHub repository <https://github.com/SDI4Apps/cloud-platform>. This configuration file can be used in any IaaS cloud which provides a disk image with an operating system with preinstalled software package named **cloud-init**. The configuration file contains instructions for modifying the base system to a system with all needed SDI specific enablers installed and running.

The configuration file is specific for a given version of an operating system. Originally we have chosen the **Ubuntu 14.04 LTS (Trusty Tahr)** version of Linux, because in that time it was the latest stable version of the most widely used Linux distribution, with long time support (hence LTS in the name) which means provided security updates for 5 years since its release in April 2014 (hence 14.04 in its name). Since then, a new version of Ubuntu Linux named **Ubuntu 16.04 LTS (Xenial Xerus)** was released in April 2016, with long time support provided up to April 2021. This new version of Ubuntu is already available as a disk image in all the main commercial clouds - Amazon AWS, Google CE, Microsoft Azure - as of September 2016, thus it can be used for deployment.

We have created a new version of the configuration file intended for the new version of Ubuntu, the file is named **user-data-xenial.yaml**. The new version installs the same set of SDI specific enablers as the previous version, just into a newer version of operating system, so for example the included version of PHP is 7.0 instead of 5.5.

Example screenshot:

[illegible]

Figure 15 - Example of script output

8 CONCLUSION

The final platform is now available as virtual server. It is now ready for utilization by pilots and external users. The first tests started at the Riga Hackathon, where separate instantiations were deployed for the pilots. Several problems were identified and solved.

The availability of the platform as Virtual servers is important for future business modelling. It can also be considered an enabler to a potential Open Source product, where support to many potential users can entail future incomes. This is a necessary preliminary step, to reach the level of available commercial services.