Thesis in geographical information technics nr 17

Development of a Web GI System for Disaster Management

Anton Lundkvist

Program in surveying and land management

Faculty of Engineering

Department of Physical Geography and Ecosystem Science

Lund University

Development of a Web GI System for Disaster Management

Author: Anton Lundkvist

Master Thesis, 30 credits EXTM05

Supervisor: Ali Mansourian, Departement of Physical Geography and Ecosystem Science Lund University

Examinor: Petter Pilesjö, Departement of Physical Geography and Ecosystem Science Lund University

Opponent: Frida Christiansson

Abstract

Spatial data is an important and necessary ingredient in the disaster management cycle. The preparedness phase requires simulation of disastrous events, identification of disaster management hotspots and creation of hazard- and risk maps. A rapid response in the event of a disaster requires good situational awareness and fast sharing of that knowledge across the community. Further, in the recovery phase, conclusions drawn from evaluating impacts from past events through analysing spatial data can be used to prevent reconstruction of risk prone infrastructure and also prevent repopulation of hazardous areas.

In the 90's, a new awareness of global climate change and hazards linked to this phenomenon, gave GIS a new role in disaster management and risk assessment. Web based systems gained quickly in popularity due to the simplicity and accessibility of the services. This rapid evolvement of GI web services and web GIS in the context of disaster management, has offered a plethora of possibilities for system development for this specific use.

A robust and dynamic GI system for disaster management and risk assessment requires dynamic data distribution, a meaningful visual representation of that data to the end user and tools for spatial processing to be integrated into the system. The functionality of the system should also be adjusted to the specific needs of the stakeholders and decision makers involved in the disaster management activities for which the system is intended.

The aim of this thesis is to develop a web GIS based on a set of requirements expressed by WHO Regional Office for Europe. The system aims at being highly interactive, interoperable and accessible through implementing OGC compliant web services and using modern open source techniques for web mapping. The system uses a combination of client-side and server-side spatial processes to create interactivity for end users and offers enhanced accessibility of metadata through a REST API built on top of GeoServer, as a compliment to OGC services.

The system development is a part of the Web GIS for Risk Assessment (WGRAS) project at the Department of Physical Geography and Ecosystem Science at Lund University. A first version of the system was presented for the member countries of WHO Europe in Yerevan in Armenia, at a workshop for GIS for risk assessment and disaster management, the 2nd of December 2015.

Preface

This thesis was written by Anton Lundkvist and supervised by Ali Mansourian at the Department of Physical Geography and Ecosystem Science at Lund University. The thesis was a part of the Web GIS for Risk Assessment (WGRAS) project as a collaboration between the Department of Physical Geography and Ecosystem Science and WHO Regional Office for Europe in Copenhagen.

Contents

1 Introduction

1.1 Background

A disaster is an occurrence disrupting the normal conditions of existence and causing a level of suffering that exceeds the capacity of adjustment of the affected community. Disaster management includes preparedness, response, rehabilitation, reconstruction and prevention for such events. The aim of a reliable and stable infrastructure for disaster management is primarily to avoid the occurrence of the disasters but also to reduce the losses and achieve rapid and durable recovery after the event.

In recent years, the general use of GIS and accompanying technologies has seen a strong raise in popularity. This can partly be explained by the emergence of internet in the early nineties, which has taken GIS from being a field for experts-only and introduced it to a much wider user group. The introduction of GIS to wider audience has led to a rapid development of new easily accessible and user-friendly tools, both online and on the local network.

At the same time the frequency and severity of natural and manmade disasters have increased worldwide due to increase in population and increasing population concentration in hazard prone environments. The need for stable and reliable infrastructures for disaster management has thus risen and GIS and especially Web GIS and GI Web Services has become a natural component of such infrastructures. The role of GIS in these systems is to provide tools for visualization, spatial analysis and sharing of data between decision makers.

The system presented in this thesis was developed in the context of the Web GIS for Risk Assessment System (WGRAS) project at the Department of Physical Geography and Ecosystem Science at Lund University. The project and the resulting application is a collaboration between the GIS-Centre and the WHO regional office for Europe in Copenhagen. It was presented for the member countries of WHO Europe in Yerevan in Armenia, at a workshop for GIS for risk assessment and disaster management, the 2nd of December 2015.

1.2 Problem statement

When designing a web GIS for disaster management a number of decisions has to be made based on the requirements of the system. These requirements can look different based on the involved organizations and decision makers using the system but also the geographical area that the system is targeted for. However, there are some general key features associated with most systems.

Due to the dynamic nature of disaster response, the data has to be updated and readily available in the system at any time. This implies a good structure for storing and accessing the data over the network. There are a number of ways to store geographical data but the most common are either a file-based system or in a geographical database, both having their pros and cons.

Disaster management also requires a meaningful visual representation of the data to the end user. In a web GIS the client is usually a web browser which puts some constrains on data formats. Even though the new standard of HTML5 has made browsers much more capable than before, they still have large limitations. Map data often comes in specialized file formats and therefor has to be converted into readable formats for the browser to be able to interact with it.

A common task in disaster management is prevention and preparation for feature events. This implies simulation and risk analysis of possible events and scenarios. It is therefor desirable for the system to implement this functionality and to be able to present and persist the results of such a simulation for further analysis. This functionality is common in desktop environments where data is stored in proximity to the user e.g. on the same machine and the process can run without network communication. A migration of this functionality into a web GIS implies a separation between data and client, which requires well defined communication protocols between the client and the server.

It is also desirable that the system provides good accessibility meaning that it should be easy for the users to discover data suiting their particular needs. This ability is paired with the systems interoperable capacities, meaning that the system can integrate and create links to external data sources that can be used within the application context.

1.3 Aim

A web GIS for emergency and disaster management will be developed according to a set of requirements defined by WHO in Copenhagen for the first version of WGRAS. The aim can be split into the following parts:

1. Develop a prototype system architecture for a web GIS for disaster management based on the requirements expressed by WHO in Copenhagen.
2. Based on the requirements and the proposed system architecture, develop a working application for disaster and emergency management with focus on usability, accessibility and interoperability.

1.4 Method

The method in this study includes three main parts. In the first step, a short study on available frameworks and policies for disaster management was conducted. This was done to get a basic but important understanding of definitions, terminology and mechanics embraced by the global community of disaster and risk management.

The second step was to study the available techniques used to create interoperable and accessible web GIS for disaster management. This was achieved by going through research in the area of web GIS in general and web GIS for disaster management in particular, and also by searching the web for live web GIS implementations. Since web GIS is undergoing a rapid development only publications from the last ten years were considered.

In the third step, a prototype system architecture was developed. Based on requirements regarding functionality and with the common structure of disaster management frameworks in mind, a final implementation was developed using techniques discovered in the second step. This implementation was also based on lessons learned from the WHO workshop in Yerevan. When no existing techniques were found to fully cover the desired functionality, custom implementations were added.

1.5 Requirement Specification

The requirements expressed by WHO for the primary version of WGRAS are presented in the following list.

General requirements:

1. The application should be implemented using open source techniques.
2. The application should not use a database
3. The application should support user authentication.

The application should also be able to:

4. Visualize data derived from a risk analysis process.
5. Visualize raster and vector data.
6. Perform some spatial processing.
7. Integrate external data sources.

1.6 Disposition

In section 2, definitions of Disaster and Risk are laid out together with existing frameworks for Disaster Management. Section 3 explains the role of GIS, web GIS in Disaster Management, and brings up some examples of online systems for visualization of risk maps together with common technologies used for such systems. Section 4 presents the development of the WGRAS system; the technologies used and implementations. In section 5, the development and techniques used in the system are discussed based on the initial requirements and section 6 presents the conclusions drawn from this work.

2 Disaster management

This section gives a brief overview of some of the existing frameworks adopted and defined within the global community of disaster management and risk reduction. It should not be seen as an exhausting record over existing definitions or frameworks but rather as an account of the most influential frameworks adopted in the global community in recent years. It is important to know that the term disaster management may have different definitions depending on the context in which is used. Both cultural and political differences may therefor greatly influence definitions and implementations of such a framework.

2.1 Definitions of Disaster and Risk

Although there is no official definition of a disaster, all major organizations are defining it in a similar way. The International Federation of Red Cross and Red Crescent Societies (IFRC) refers to a disaster as "*a sudden, calamitous event that seriously disrupts the functioning of a community or society and causes human, material, and economic or environmental losses that exceed the community's or society's ability to cope using its own resources*" (IFRC 2015a).

Disasters can be caused by nature and originate from events such as flooding, landslide, cyclones, tsunamis etc. The occurrence of such an event itself does not imply a disaster but when it affects areas where the community is not prepared to withstand the impact of the event, a disaster often follows. A disaster can also originate from human activities, such as terrorist attacks and wars or from technical failures such as nuclear meltdowns or accidents (Mansourian et al. 2006).

The immediate effects of a disaster are loss of lives, destruction of essential infrastructure and creation of victims in the affected area. To cope with and reduce the effects of the event the community has to make available large resources within a short period of time. When a community is lacking these resources or lacking the ability to coordinate them in order to reduce the impact, the negative effects of the event aggregates rapidly and could evolve into a full-scale disaster. (Mansourian et al. 2006)

2.2 Frameworks for Disaster Management

The concept of disaster management can refer to the framework defining the policies and guidelines an organization sets up for how to act when a disaster occurs. It can also refer to the actual implementations of such a framework and the activities put into practice for preventing and responding to disasters. In this section disaster management refers to the former, meaning that practical implementations and practices will not be discussed.

The Sendai Framework for Disaster Risk Reduction was established in march 2015 by the United Nations (UN) and is a non-binding agreement for UN member nations. It is meant to provide guidelines at local, national, regional and international levels for understanding and reducing risk. The agreement also points at the importance for all levels of society to invest in and enhance frameworks for disaster management. The overall goal being a "*substantial reduction of disaster risk and losses in lives, livelihoods and health and in the economic, physical, social, cultural and environmental assets of persons, businesses, communities and countries*." (UNISDR 2016).

The United Nations Platform for Space-based Information for Disaster Management and Emergency Response (UN-SPIDER) is the primary UN organization designated for promoting space based information in disaster management. The organization is directly bound to the Sendai Framework and is responsible for providing support to countries based on the priorities expressed by the framework. It structures the efforts in disaster management in three main categories; response, rehabilitation and recovery. These three phases describe efforts being put into action once a disaster has struck and is part of a much wider concept called disaster-risk reduction where also preparedness is included (UN-SPIDER 2016).

The World Health Organization (WHO) is another major partner in the global community for disaster management. WHO is mainly active in the health sector and its role is to direct and coordinate international health within the UN system. The organization works on all levels of society and collaborates with a number of partners to maximize its coverage across the health field. (WHO 2016 a)

Just as the UN, WHO also works under the Sendai Framework and hence the structures for their Disaster and Emergency Management programs are similar to the one defined by UN-SPIDER, although more health related. In recent years the WHO has adopted a more proactive approach in their strategies for disaster management, meaning that more focus on is put on preparedness and mitigation (WHO 2016 b).

In the most recent plan for disaster management set up by the IFRC, it recognizes that for disaster management to be successful, it has to incorporate preparedness, response, and recovery phases.

Preparedness often refers to all the measures taken *before* the event. It might be actions taken on a national level, by a community or an organization to prepare for a disastrous event or to predict such events with the goal of minimizing its effects. Response and recovery on the other hand, are actions taken *after* the disaster has occurred (IFRC 2016 b).

Disaster response aims at stabilizing the physical and emotional condition of survivors and to rescue people from immediate danger within the affected area. It can also aim at restoring essential facilities to prevent spreading of diseases and further loss of lives. Characteristic for this phase is a need for good situational awareness and fast sharing of that knowledge across the local community affected by the hazardous event (Peggion et al. 2008).

Recovery is referred to as all the long term actions taken to rebuild the community and strengthen its capabilities. The response and recovery phases are usually parallel processes and can be seen as complementary to each other (IFRC 2016 c).

Although there exist differences in implementations and policies, the general building blocks for disaster management are the same. Preparedness, response and recovery phases are recognized as crucial for a community to build up an effective infrastructure to cope with disasters. This infrastructure can be greatly improved by the use of GIS technology, both for visualization of risk and as a tool for risk assessment in hazard prone areas (Fabbri et al. 2007).

Spatial data can therefore be considered as an important and necessary ingredient in the disaster management cycle. The preparedness phase requires simulation of disastrous events, identification of disaster management hotspots and creation of hazard- and risk maps. A rapid response in the event of a disaster requires good situational awareness and fast sharing of that knowledge across the community. Further, in the recovery phase, conclusions drawn from evaluating impacts from past events through analysing spatial data, can be used to prevent reconstruction of risk prone infrastructure and also prevent repopulation of hazardous areas (Peggion et al. 2008).

3 Web GIS in Disaster Management and Risk Assessment

In the 90's, a new awareness of global climate change and hazards linked to this phenomenon, gave GIS a new role in disaster management and risk assessment. Web based systems gained quickly in popularity due to the simplicity and accessibility of the services. Professionals in disaster management no longer had to be experts in either computer technology or GIS to integrate spatial data components in their analysis. In addition, they no longer had to download and install software on the local system, as the services were compatible with common browser technology (Peggion et al. 2008).

In 2005, the emergence of public Application Programming Interfaces (API) offered by Google and Microsoft enabled users to create customized web map applications. This also enabled a public participation in the disaster management process as damage information could be communicated directly through maps published on the web (Kawasaki et al. 2013).

Even though this was revolutionizing from a historical point of view, the available web GIS were not complete. They still lacked many of the features available in traditional desktop GIS, such as tools for spatial analysis and interoperability with other systems. As such, the systems were used mainly for visualization of spatial data from a spatial analysis process performed on a local system. Most implementations were also implemented for a specific hazard or risk area (Muller et al. 2006).

A common approach in disaster management has been the establishment of a centralized Emergency Operation Centre (EOC). The EOC is a facility for collection and dissemination of damage information. This information is used for creating hazard and risk maps for use in the decision making process (Kawasaki and Sadohara 2005). In EOC based disaster management structures, all or most damage and risk information passes through the EOC pipeline, which is controlled by a relatively small and inaugurated team of specialists, including GIS professionals. However, in recent years, the introduction of web GIS into the disaster management process has made the structure less centralized and more dynamic. The possibilities of rapid data exchange over the web has enabled collaboration between the public and the disaster management experts in ways that were not possible only decades ago. It is proposed that this technical evolution therefor represents a paradigm shift in the disaster management community (Kawasaki et al. 2013).

3.1 Participative web GIS

Participative web GIS is a system that makes use of stakeholder's knowledge of the local territory to aid in the selection of risk reducing strategies in the disaster management cycle. The idea is that if this knowledge can be collected, integrated and evaluated in the disaster management process, the outcome of the process will be more accurate and better adopted to the local environment (Aye et al. 2015).

Resent research has proposed a two-stage workflow for participative web GIS system aimed at facilitating work in the preparedness phase in disaster management process. In stage one, low-level stakeholders such as technical experts from different areas can make initial risk assessments and propose strategies for mitigation based on the available risk information.

The system is able to present risk maps that are either stored in system itself or uploaded by the user. By creating new features and assign attributes to those features, risk-reducing measures can be formulated and stored in the system. This can be new constructions such as dams to prevent a flooding etc. The proposed alternatives for preparation and mitigation is then passed on to high-level decision makers that can make use of Multi-Criteria Evaluation (MCE) to make a selection of measures that should be considered. This functionality is provided by the system in a decision-making module (Aye et al. 2015).

It is pointed out that an important feature in such a system is the ability to support users with different roles and hence different administrative rights. In the system, the *admin* role is assigned to low-level stakeholders and a public role is assigned to high-level stakeholders such as decision-makers. This differentiation and separation of roles and access rights within the system has the advantage of reducing the risk for incorrect use of resources. It also lessens the potential confusion a user might feel over being presented with tools he/she might not know how to use. A role based access system is therefor considered as a fundamental functionality for a participative web GIS system where users from different organizations and with different competences are collaborating (Aye et al. 2015).

3.2 Public platforms

Public platforms are systems that do not restrict data access to a set of user but aims at spreading the data in the open domain. The role of such a system in a disaster management context is to create awareness by spreading knowledge about risk and improve resilience by making citizens aware of their own role in disaster management. Resilience can also be improved by visualizing how a potential disaster might affect the community or local environment of the citizen. Public platforms can therefor be seen as an important tool for communication between experts, decision makers and the community (Albano et al. 2015).

Public platforms can be especially effective in areas where the government's own ability to implement disaster management frameworks is weak due to lack of resources. The platform can then be used for spreading information from other sources such as research projects in other countries targeting that area. This has been investigated in the Malawi Spatial Data Platform (MASDAP), which is an online tool where users investigate existing risk maps covering Malawi and surrounding areas. The tool also provides uploading and creating custom risk maps that can be shared with others (Balbo et al. 2013).

3.3 Examples of web GIS implementations

In this section, a few examples of online applications for visualizing risk data are presented. To structure the examination of the applications, the categories *Description* and *Useful Features* have been used and they are defined as follows:

- **Description**: Describes the application and its main features, such as appearance and main functionality.
- **Useful features:** Aims at describing interesting and useful features in the context of Disaster Management.

3.3.1 The WHO e-atlas of disaster risk for the European Region

3.3.1.1 Description

Available at: http://data.euro.who.int/e-atlas/europe/

This tool was produced by the WHO Regional office for Europe with a goal to encourage the ministries of health in member countries and other stakeholders to improve their disaster management capacities. The application is a pure visualization tool where a user can choose to view maps by country, region (Europe) or by hazard. The covered hazards are Seismic, Flood, Landslide, Wind speed or Heat wave, where Wind speed and Heat wave have subdivisions based on return periods.

The application has to main user scenarios. The first is to view documentation such as how to use the application and read about the methodology used to produce the maps. The second is to view the maps by choosing region or hazard to visualize.

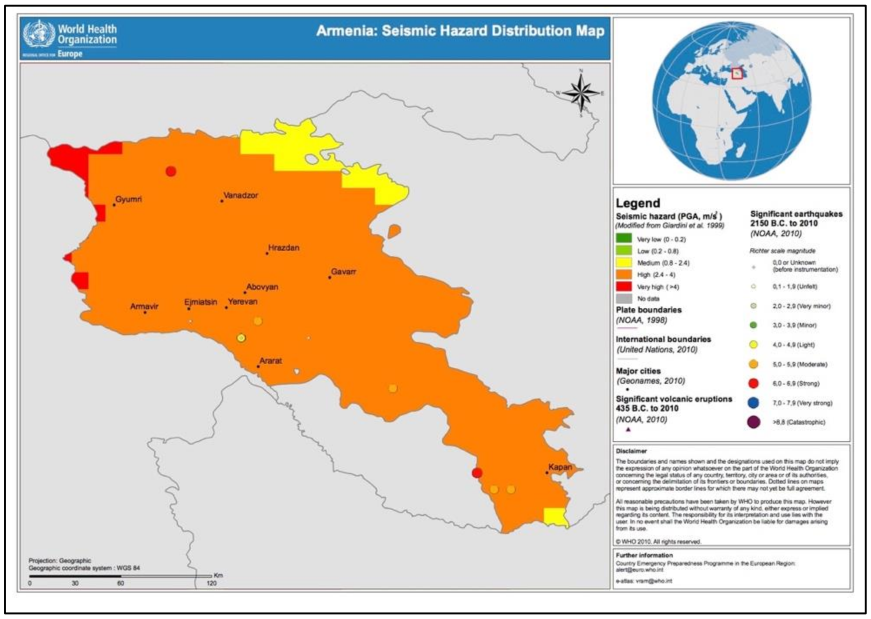
To view the documentation, the user is presented with a list of links from where the user can download different pdf documents describing the models used in each step of the production of the data used in the application.

To view the maps, the user can choose between region-based or hazard index-based visualization. Each path leads the user into a series of new choices to narrow down the information finally presented on the map.

Once a map is presented to the user, it offers few possibilities for interaction. The map has zoom and pan buttons but panning is restricted to the region that the map was generated for and zooming can be done in three steps.

3.3.1.2 *Useful features*

A more important feature is the ability to download the generated map as a pdf, see figure 1. The generated pdf contains the map, a legend and some further information. The map is also generated with the WHO logo and a title. This is nice feature as it makes it possible to use the map offline for different purposes and it is certainly one of the core functionalities.



Disclaimer

The boundaries and names shown and the designations used on this map do not imply the expression of any opinion whatsoever on the part of the World Health Organization concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries. Dotted lines on maps represent approximate border lines for which there may not yet be full agreement.

All reasonable precautions have been taken by WHO to produce this map. However this map is being distributed without warranty of any kind, either express or implied regarding its content. The responsibility for its interpretation and use lies with the user. In no event shall the World Health Organization be liable for damages arising from its use.

© WHO 2010. All rights reserved.

Further information

Country Emergency Preparedness Programme in the European Region: alert@euro.who.int

e-atlas: vram@who.int

Figure 1, PDF map, The WHO e-atlas for Disaster Risk for the European Regio (WHO 2016c)

The documentation for the WHO e-atlas is rich and this is one of the main advantages with this application. The user can get precise information about the models used to produce the hazard index data. The documentation is provided as links to pdf documents available for download.

3.3.2 SCALGO LIVE GLOBAL

3.3.2.1 *Description*

Available at: http://scalgo.com/live/

SCALGO is a Danish company specializing in implementing terrain data processing algorithms. Their algorithms build on research from the Centre for Massive Data Algorithms at Aarhus University in Denmark. The SCALGO LIVE GLOBAL application is a map showing digital elevation models with global coverage. The elevation data is used as input to models for water flow, sea-level rise and watersheds.

The water flow model calculates how surface water is transported to the ocean and the user can change the water network detail by choosing the minimal upstream area for a river.

The sea -level rise module shows how the water level rises in the landscape based on a threshold level given by the user, see figure 2. By clicking on the map, the user can also find out how much the water level has to rise to reach that specific point.

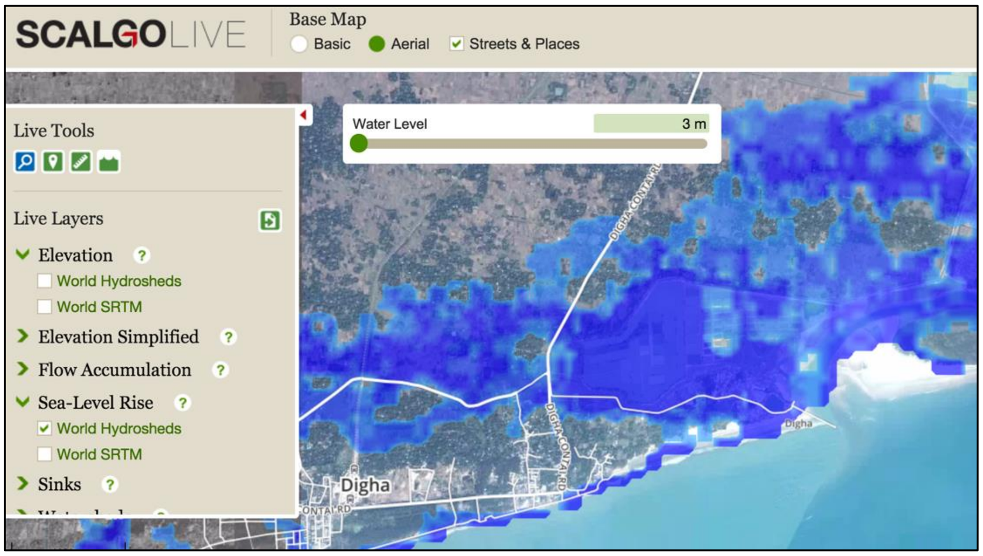


Figure 2, Sea-level rise module, SCALGO LIVE GLOBAL (SCALGO 2016)

The water-shed module is used to calculate the water shed area for a specific point on the map. When a point is selected, the upstream area that drains through that point is calculated and shown on the map together with the "nearest" path for the water to reach ocean level.

The application is easy to use and visually appealing. It is implemented as a Single Page Application (SPA), which means that the user does not have to navigate to different URLs to access new functionality. Instead, every feature is presented in a popup or window much like a desktop application. Instructions on how to use the tools are easy to access by clicking on icons in the menu.

3.3.2.2 Useful features

The most interesting feature of this application from a disaster management or risk assessment perspective is probably the water level module. The user can get a visual presentation of the water level by increasing the threshold value with a slider in the menu. The scale of the slider implies that the precision is 1 mm, which is quite incredible. No documentation was found to confirm this.

A tool like this could be useful for flooding analysis when hazardous areas have to be identified, either in a simulation to increase preparedness or when a flood has already occurred and the water level has to be continuously monitored. The main advantage is that the tool is giving instant feedback on user input.

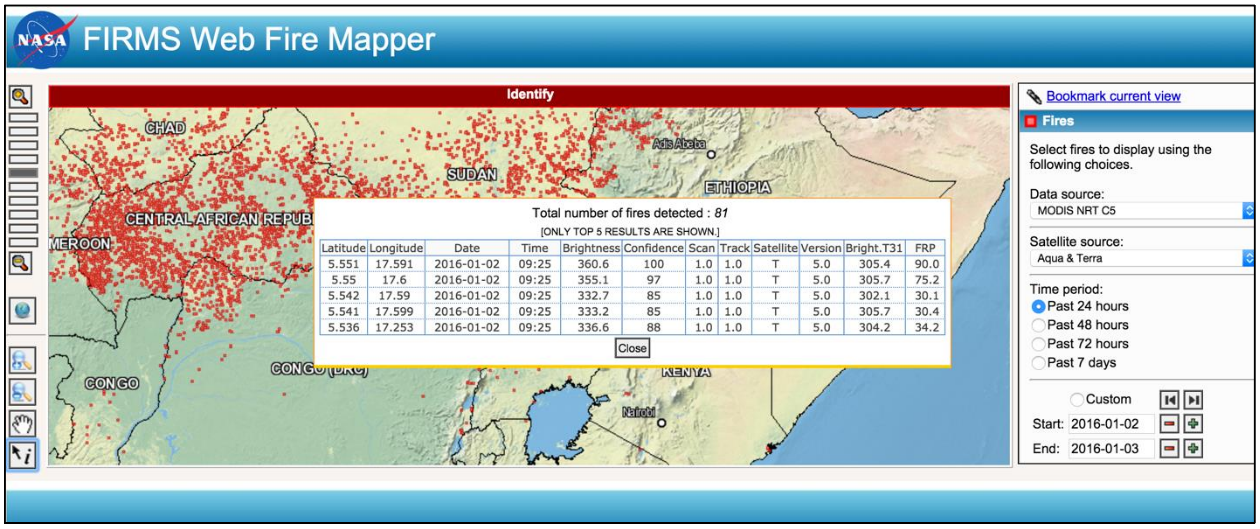
3.3.3 FIRMS Web Fire Mapper

3.3.3.1 Description

Available at: https://firms.modaps.eosdis.nasa.gov/firemap/

The FIRMS Web Fire Mapper shows near real-time fire data from the past 24 hours, 48 hours, 72 hours and 7 days back in time. It is produced by NASA by analysing data from the Modis sensor on board the Aqua EOS and Terra satellites and uses the Fire and Thermal Anomalies Product to find active fires within a 1x1 km pixel. If one or more fire is found within the area, the pixel is marked as active. Active fires are displayed as red squares on the map.

The user can choose the desired time interval by checking checkboxes in the menu. There is also a possibility to customize the time interval for which to display fires. The data and satellite source can be altered in the same menu. An identify tool lets the user get information about active fires in a particular location. The information is presented as a table with the coordinates, date, confidence and some other categories describing the hotspot, see figure 3.



Total number of fires detected : *81*

[ONLY TOP 5 RESULTS ARE SHOWN.]

Latitude	Longitude	Date	Time	Brightness	Confidence	Scan	Track	Satellite	Version	Bright.T31	FRP
5.551	17.591	2016-01-02	09:25	360.6	100	1.0	1.0	T	5.0	305.4	90.0
5.55	17.6	2016-01-02	09:25	355.1	97	1.0	1.0	T	5.0	305.7	75.2
5.542	17.59	2016-01-02	09:25	332.7	85	1.0	1.0	T	5.0	302.1	30.1
5.541	17.599	2016-01-02	09:25	333.2	85	1.0	1.0	T	5.0	305.7	30.4
5.536	17.253	2016-01-02	09:25	336.6	88	1.0	1.0	T	5.0	304.2	34.2

Select fires to display using the following choices.

Data source: MODIS NRT C5

Satellite source: Aqua & Terra

Time period:
[✓ Past 24 hours]
Past 48 hours
Past 72 hours
Past 7 days

Custom
Start: 2016-01-02
End: 2016-01-03

Figure 3, Detected active fires for a specific location, FIRMS Web Fire Map (FIRMS 2016)

3.3.3.2 Useful features

The ability to get near real-time fire data visualized on a map is very useful for disaster management. It could for example give useful information to decision makers trying to coordinate resources in an ongoing disaster.

FIRMS Web Fire Mapper also exposes a Web Map Service, which makes it possible to integrate available data into a desktop GIS or web GIS application. The data is also available for download in the most common spatial data formats.

3.4 System architecture and techniques for web GIS

The high-level architecture of any web application usually consists of a server and a client. Further, three main parts are required for the application to be fully functional; storage, distribution and visualization. (Evans and Sabel. 2012)

To store data on the server, a database of some sort is used. The database can be implemented as a folder system storing information in files, or a Database Management System that lets other parts of the system interact with a database storing data in tables and supports a query language to access the information. (Harrie and Svensson 2012)

The distribution of data is handled by a web server. The web server is responsible for serving static content such as HTML, JavaScript, images etc. and to fetch data from the database and serve this to the client.

The client, usually a browser is responsible for visualizing the data received from the server. Based on the HTML and JavaScript received from the web server, it builds up a Document Object Model (DOM) in which the user can interact and give new input to the system. Upon interaction, the browser sends a new request to the server and based on the response, it can rebuild the DOM to present the new information.

This general architecture is also valid for web GIS applications but the use of spatial information requires support for other data formats. This support has to be implemented at all three levels of the architecture. (Evans and Sabel. 2012)

On the storage level, spatial data support is achieved by extending the database with spatial capabilities, meaning both support for spatial data types and an extended query language to perform spatial queries. The extended DBMS can convert spatial data and serve it in a variety of formats known to the web server. A file system could also be used, which implies that the server must be able to read the file formats, as no initial conversion will be done.

The distribution of spatial data requires that the server is able to parse spatial data acquired from the storage into formats useful for the client. The common way is to add an additional server to the system, e.g. a map server. The map server is specialized in serving spatial data and in most cases it is also supports spatial queries and topological functions.

For visualizing spatial data in the browser, a common protocol for requesting the data from the server is needed. The protocol can be seen as the language that the components use to communicate; if they do not speak the same language the communication fails. To continue this analogy, the language also needs to implement expressions for describing the content in the communication. For spatial data, this could for instance be expressions for describing location, extent and topology.

3.4.1 Data storage

3.4.1.1 File system

In a web application, data storage and retrieval of that data in a fast and secure manner is crucial. The simplest form of data storage for a web GIS is to simply store files on the web server in a location that can be reached by the server program. The file format has to be known to the server program so that the relevant content can be extracted. This technique is used in applications where only a small amount of data is used. (Harrie and Svensson 2012)

The advantage of using a file system is that it is easy to maintain, as no database server has to be configured. However, to access information in a file the whole file has to be loaded and read by the server. The server then has to scan the content to find the right information before forwarding it to the client. To change the content of a file, the application first has to check that no one else is working with the same file. This can be a problem if the application has many concurrent users. (Harrie and Svensson 2012)

3.4.1.2 Databases

Databases typically organizes data in models meant to reflect objects in the real world. The term database refers to the collection of data and the structures used to represent that data. It can be tables, views, relations between tables etc. The database is often part of a Database Management System (DBMS), which is used to encapsulate the database and expose its capabilities to a network of other databases or applications. The DBMS also defines a query language, which is used by external components to extract and alter the data in the database (Elmasri 1994).

Today the most popular database model for general use is the relational model that uses table structures for organizing data. Each row of the table is depicting an object and each column defines the attributes. A combination of the relational database model and the object oriented database model is commonly used in GIS and is called Object- Relational database. This model structures data in tables and provides all the functionality of a relational database but adds support for user-defined types, which can be used to add support for geometry (Harrie and Svensson 2012).

3.4.2 Data distribution

The central role of a web server is to process requests from a client, usually a browser. The web server interprets the requests and responds with serving content from the data storage configured for that particular server. The communication between the client and the server needs to be performed in a standardized way since a misinterpretation on either end could result in bad user experience, loss of data or compromising data integrity. This section describes the standardized HTTP communication protocol and OGC web Services, which are standardized HTTP interfaces for exchange of spatial data between client and server.

3.4.2.1 Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a standardized communication protocol for the World Wide Web (WWW). It uses Uniform Resource Identifiers (URI) to identify the path to the resource. The protocol also uses a request-response model and defines a set of request methods commonly known as HTTP-verbs (W3 2016a).

The currently used standard, HTTP/1.1 defines eight different methods where the most commonly used are GET, PUT, POST and DELETE (W3 2016b).

A HTTP request always uses one of these methods and specifies the path to the requested resource in the URI. An example of a HTTP GET request could be that someone uses a browser to access a website.

The request is sent to the URI specified in the address bar of the browser. If the URI is valid, e.g. if a web server is configured to respond to that specific URI, a response is sent back containing the static files required by the browser to build up the DOM (W3 2016b).

The HHTP/1.1 standard defines GET, PUT, POST and DELETE methods for the following purposes:

The GET method should be used to *retrieve* information from a resource. The request can use parameters in the URI to define e.g. the range of data to be returned in the response.

The PUT method should be used to *update* an already existing resource in the storage. If the resource does not exist, the request can also invoke the creation of that entity.

The POST method should be used for *creating* new resources. The resource to be created should be submitted in the request body.

The DELETE method should be used for *deleting* a resource on the server.

An implementation of HTTP endpoints on the server does not necessarily have to follow these rules. A web application could be configured to use only POST requests throughout its client-server communication and still be highly functional. However, it is highly recommended to follow the standard to avoid confusion if the server exposes its endpoints to more than one client (W3 2016b).

3.4.2.2 OGC Web Services

In a web GIS implementation, the standard HTTP protocol is usually not enough for distributing spatial data. For instance, there is no definition in HTTP/1.1 on how to request or send a georeferenced image over a network. For such a request to be meaningful, it might have to define the format of the image, the extent or the bounding box of the image and in what coordinate system the extent is defined.

The Open Geospatial Consortium (OGC) is a consortium with over 520 members that works with developing standards for a more GIS enabled web. OGC has worked extensively with defining new standards that extends HTTP to overcome the problems mentioned above (OGC 2016a).

OGC Web Services (OWS) refers to a set of standards defined for publishing raster and vector data over the web. All OWS can be seen as HTTP interfaces since they build on HTTP request methods and adds supports for additional parameters (OGC 2016b).

The specification for OWS is quite extensive and the following description does therefor not provide a complete list of parameters used in each request. The most commonly used operations are described as follows, together with some example requests and responses.

3.4.2.2.1 Web Map Service

Web Map Service (WMS) is used for requesting raster data in formats that the browser is able to visualize. The two most commonly used versions are WMS 1.1.1 and WMS 1.3.0. WMS requires a set of parameters to be set in the URL so that the right operation is performed on the server. The most common operations used are *GetCapabilities*, *GetMap* and *GetFeatureInfo* (OGC 2016b).

The GetCapabilities operation should generate an informative response listing all the resources available on the server together with some metadata and available operations. The specification specifies which parameters are mandatory and which are optional. For instance, the *request* parameter has to be set to *GetCapabilities* and the *service* parameter has to be set to *WMS*. The response format should be valid XML according to a Document Type Definition (DTD) defined in the specification (Michaels and Ames 2008).

A GetMap request instructs the server to generate a map from the requested resource. The operation requires ten mandatory parameters to be set in the URL. Examples of required parameters are *layers* indicating the data resource the map should be generated from, *SRS* indicating the spatial reference system to be used and *bbox* for the map extent. One optional parameter is *transparent* which indicates if the background of the map should be transparent or not. The generated map can be delivered in a list of different formats, depending on the implementation (Michaels and Ames 2008).

The GetFeatureInfo request lets the client query attribute data of a feature at a specific location. Just as the other operations the request needs specific parameters to be set in the URL, such as what data resource and location the query refers to (OGC 2016b).

A typical workflow for requesting a WMS with the main steps 1-3 is shown in figure 4. Communication with the database is omitted.

1. The client first performs a HTTP GET request to the *GetCapabilities* URL to find out what resources are available.
2. The client parses the request and finds the necessary details to build up the URI for a *GetMap* request for a specific resource.
3. With the information found in the XML-document, a *GetMap* request can be performed. The response from the server contains an image in the requested format.

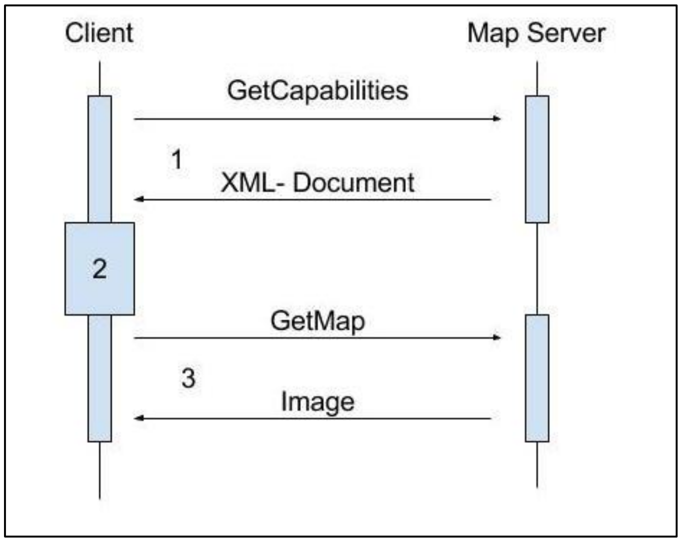


Figure 4, WMS request sequence (OGC 2016b)

3.4.2.2.2 Web Feature Service

The Web Feature Service (WFS) is a standard for exchanging vector data. OGC has specified several versions of WFS but the most widely used are 1.0.0, 1.1.0 and 2.0.0. The standard is extensive and supports several operations depending on the version that is used. The most common operations are *GetCapabilities*, *DescribeFeatureType* and *GetFeature* (OGC 2016c).

The *GetCapabilities* operation is similar to the same operation for a WMS. It produces a metadata document in XML syntax containing all the resources available on the server together with the supported WFS operations. The parameters for each operation are also listed and the document should be valid according to a specified DTD (Michaels and Ames 2008).

A *DescribeFeatureType* operation produces a description for a specific feature type or data source. The description includes a list of features for that feature type and the attributes describing it. The client can request the information in a number of output formats. The supported formats depend on the server's capabilities (OGC 2016c).

The *GetFeature* operation is used to request a collection of features from a specific data source on the server. Since this request could potentially produce a large amount of data, the client is able to limit the response in a number of ways. If only a single feature is needed, the *featureID* parameter can be specified in the URL and only the feature with a matching id will be returned. In many cases the *client does not know the featureID* and in that case the *count* parameter can be used. By setting *count* to a number n, a maximum of n features will be returned in the response. Features can also be requested and sorted by attribute name (OGC 2016c).

The *GetFeature* operation also supports returning feature types in different formats. The most common formats used in web GIS implementations are GML and JSON.

A typical workflow for requesting WFS with the main steps 1-5 is shown in figure 5. Communication with the database is omitted.

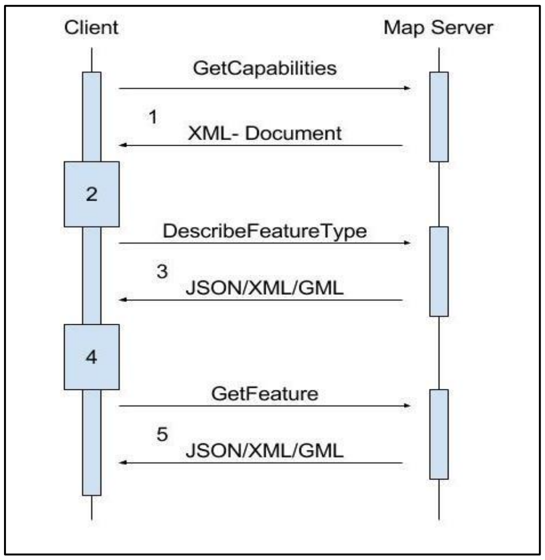


Figure 5, WFS request sequence (OGC 2016c)

1. The client performs a HTTP GET request to the *GetCapabilities* URL to find out what resources are available.
2. The client parses the request and finds the necessary details to build up the URI for a *DescribeFeatureType* request.
3. The client initiates a *DescribeFeatureType* request. The response from the server contains detailed information about specific feature types, such as attribute names.
4. The client parses the response and based on this information it “decides” what feature type it wants to request. It then builds up the URI for the *GetFeature* request.
5. The *GetFeature* request is sent from client and the server responds with a textual representation of the vector data in a data format supported by the server. This could for instance be JSON, GML and XML etc.

The information in the capabilities XML document is actually enough for building up the URI for step 5. Hence, in the sequence described in figure 5, step 3 and 4 are only necessary if the client needs detailed information about feature types before requesting the actual feature. (Michaels and Ames 2008)

3.4.2.2.3 Web Coverage Service

The Web Coverage Service (WCS) is an OGC standard for retrieving the raw imagery of raster data. WMS and WCS requests can be performed on the same data source but while WMS will transform and return that data in a format readable by browsers, WCS can return other formats as well. Examples of such formats is GeoTiff, ArcGrid or GTopo30. WCS can be used for complex spatial analysis when raw information available in the source data is needed. Due to incompatible formats, that analysis can not be done in the browser but needs a desktop GIS or a web server capable of reading and parsing that information.

The WCS standard defines the operations *GetCapabilities*, *DescribeCoverage* and *GetCoverage*. Just as the other OGC services, the desired operation has to be specified in the URL together with additional required and optional parameters specific for that operation. (OGC 2016d)

The *GetCapabilities* operation is similar to the WMS *GetCapabilities* operation. It returns a listing for the WCS resources and operations available at the server at the moment. The *DescribeCoverage* operation returns information about the coordinate system, the metadata and the available formats for that resource. *GetCoverage* returns the source data for the specific resource. The client can choose to request a portion of the data if needed (OGC 2016d).

3.4.2.2.4 Web Processing Service

The Web Processing Service (WPS) provides a standard for geospatial processes implemented as web services. The standard defines how a server should publish such processes and how HTTP requests and responses should be implemented for both the client and the server (OGC 2016e).

In contrast to WMS, WFS and WCS, the WPS standard does not specify that the input data for the process should be available on the server responsible for the process. Instead, it also allows data to be included in the request itself. This makes it possible for a client to submit data stored or created at any other location, get it processed and retrieve the result for further use. The standard does not specify which spatial operations should be available or how they should be performed (OGC 2016e).

The WPS standard defines *GetCapabilities*, *DescribeProcess* and an *Excecute* operations. The *GetCapabilities* operation returns a list over the available processes in a valid XML according to a DTD defined by the standard. *DescribeProcess* returns a description for a particular process and the *Excecute* operation is used for starting the actual process at the server.

The general request- response sequence for a WPS is shown in figure 6.

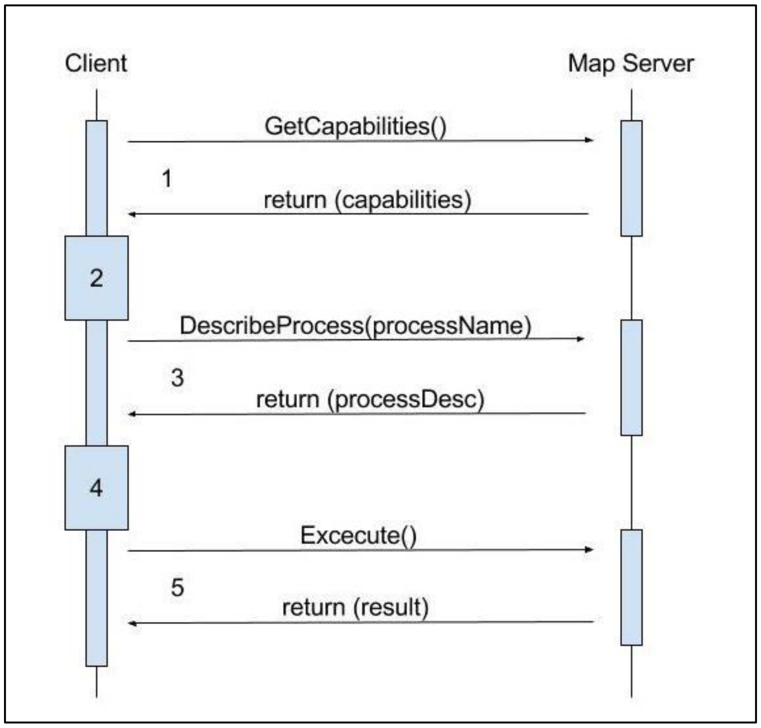


Figure 6, WPS request sequence (OGC 2016e)

1. The client performs a HTTP GET request to the *GetCapabilities* URL to find out which processes the map server exposes.
2. The client parses the request and finds the necessary details to build up the URI for a *DescribeProcess* request.
3. The client initiates a *DescribeProcess* request. The response from the server contains detailed information about a specific process, such as required and optional inputs and output parameters and formats for both input and output.
4. The client parses the response and based on this information it “decides” which process it wants to execute. It then builds up the URI for the Execute request.
5. The *Execute* request is sent from client and the server responds in a format agreed upon by the client and server. This could for instance be JSON, GML, XML for vector data or PNG or JPEG for raster data.

According to the WPS interface standard, the *Excecute* operation should return a processing job identifier that the client can use to check the status of an ongoing process. This can be useful in the case a large amount of data is being processed. In that case the client may want to perform other tasks while waiting for the response to come back. The client can then "ping" the map server under an ongoing process by providing the process id in a HTTP GET request. If the process is not finished, the map server will respond with a status update. The status can indicate that the process is currently ongoing or finished (OGC 2016e).

Another important feature of WPS is its dynamic input and output capabilities. The client can send input data either as value or by reference and the process can respond with either a value or a reference. Input and output by value means that the actual data is included in the request and response. Input and output by reference means that instead of including the data in the request or response, a URI pointing to the resource is provided (OGC 2016e).

If the client wants to send input as value, it has to attach the actual input data in the HTTP method body sent to the *Execute* operation. This scenario is shown in figure 7.

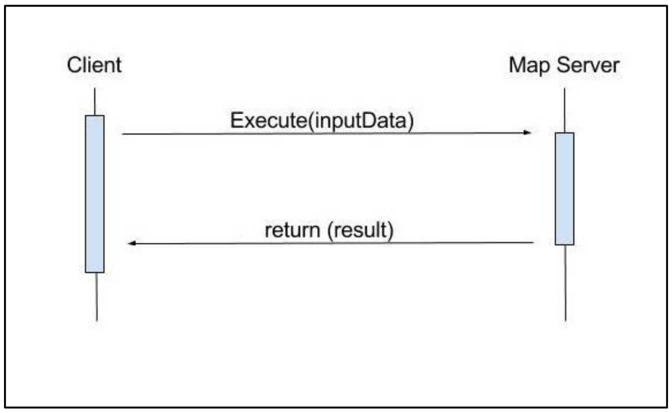


Figure 7 WPS input as value (OGC 2016e)

To provide input by reference, the client has to submit a reference such as a URI in the request. The URI should then point to a data source available to the map server at the time the process is executed. This scenario is shown in figure 8.

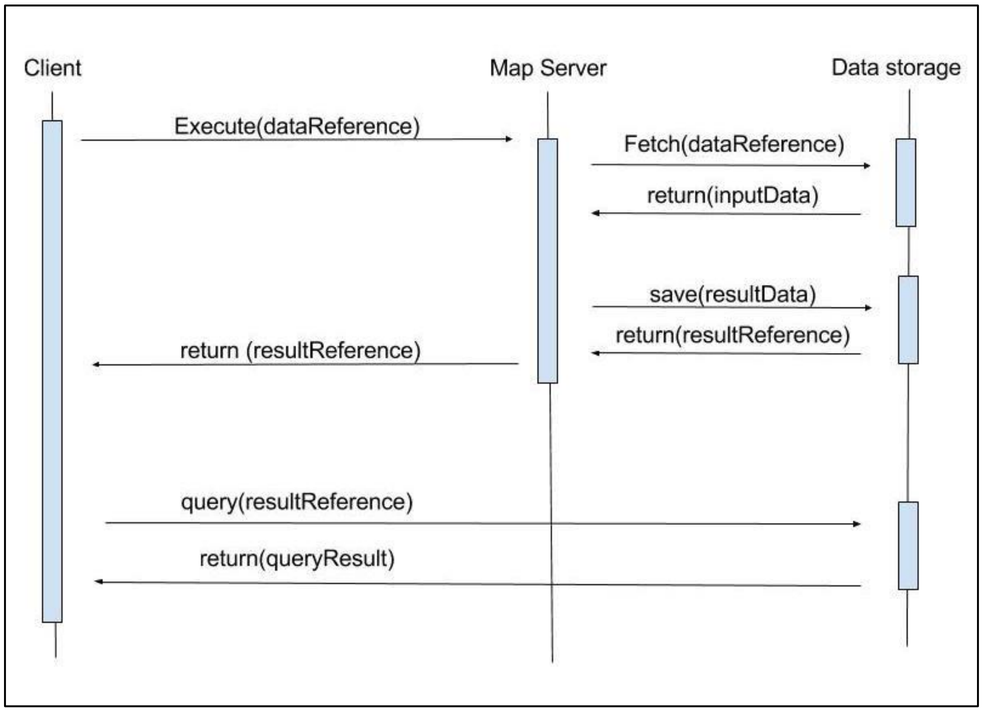


Figure 8, WPS input by reference (OGC 2016e)

3.4.3 Visualization of spatial data

The previous sections have showed how spatial data can be stored in either a file system or a database, how the server can process that data and finally distribute the data to other components in a web GIS. The end user of such a system is usually a human interacting with a computer, so the data also needs to be visualized.

This section will focus on the client part of the system and present some techniques and practises used to achieve an understandable and human friendly representation of spatial data on the web. The focus will be on browser technology and not dive deeper into hybrid implementations such as combined desktop-web systems.

3.4.3.1 Client types

Based on their architecture, client implementations can be split into the two categories thin and thick client types. A purely thin client is used for requesting and displaying data only; it delegates most or all the data processing to the server. The reason for using a thin client architecture is that the server usually has more computing power than the client machine. It can also be advantageous from a development perspective to separate the visualization from the processing functionality. This is a well known programming pattern called Model-View-Controller and aims at separating components based on their functionality (Martin 2003).

A thick client means that most or even all data processing is performed on the client machine. If the client is a browser, native browser technology is usually not enough and third tier programs can be used as plugins. This is necessary since processing data means handling spatial data formats with no support in modern browser technology. An alternative is to distribute the client as a desktop installation and take advantage of the native technologies available for that environment. (Hung et al. 2014)

3.4.3.2 Interchange formats

The browser puts constraints on the formats used for interchanging geospatial information. This section will give a brief presentation of the most widely used interchange formats for modern browsers.

3.4.3.2.1 Extensible Markup Language XML

XML is a mark-up language closely related to HTML. XML is used for data storage and distribution while HTML is used for visualization. XML lets the user define custom elements which makes the format flexible and suitable for building up an internal data structures in an application (W3 2016c).

The main advantages with XML as a language for interchanging information over the web is the strict mark-up rules, its readability and its wide support in modern browsers. The strict mark-up rules make it easy to validate XML documents. There exists a wide variety of validation tools, both online, as plugins and as desktop installations (W3 2016c).

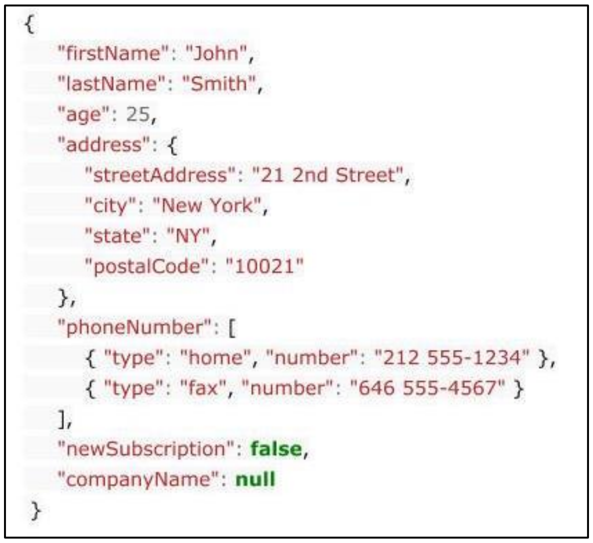
3.4.3.2.2 Geography Markup Language

Since the XML standard does not put any constraints on which elements should be used to build up the document, the format has been used to develop new standards. One such standard adopted by OGC is the Geography Markup Language (GML). The format is used to describe geographical features containing points, lines and polygons. The rules set up for GML encoding are defined in a GML schema. The schema describes the document and the rules applied to that document. Users can extend GML by develop their own schemas to create a better representation of data specific to their environment. The use of custom schemas requires that all the systems interchanging data in that format also uses the same schema for validation and encoding (OGC 2016f).

3.4.3.2.3 JavaScript Object Notation JSON

JSON is a human readable open standard format derived from the JavaScript language. It is widely used in JavaScript applications for asynchronous communication between server and client. The syntax is similar to the JavaScript syntax and thus supported by all browsers on the market (IETF 2016).

JSON can represent four primitive types: string, number, boolean, and null. It can also represent the two structured types array and objects. An example of a JSON representation can be seen in figure 9.



```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ],
    "newSubscription": false,
    "companyName": null
}
```

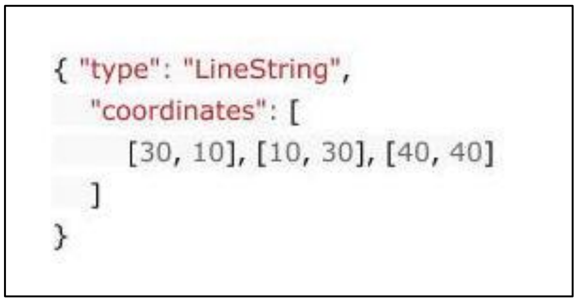
Figure 9, JSON notation (IETF 2016)

3.4.3.2.4 GeoJSON

GeoJSON is an extended version of JSON. It provides support for the geometry types Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon. The format is not adopted as a standard by OGC but is developed by an internet working group called Internet Engineering Task Force (IETF) (GeoJSON 2016).

The JSON like syntax of GeoJSON has made it popular for use in JavaScript driven map clients and it is today supported by most JavaScript mapping libraries.

An example of a LineString representation in GeoJSON syntax is seen in figure 10.



```
{ "type": "LineString",
    "coordinates": [
        [30, 10], [10, 30], [40, 40]
    ]
}
```

Figure 10, GeoJSON notation (GeoJSON 2016)

3.4.3.3 Multi- and Single Page Applications

Modern client side architecture is a broad topic and there exists an abundance of frameworks and libraries to aid in development. Multipage applications (MPA) and single page applications (SPA) can be seen as high level client side architecture concepts and since these implementations do not depend directly on specific frameworks or coding libraries, theses will not be presented here.

Multipage application (MPA) refers to an application where the user is redirected to different URLs upon interaction with the application. It has been the standard for web applications since the web was first invented and the technique is still used today.

In MPA a page is often linked to a certain action and when that action is performed by the user, the application reloads and fetches a new page from the server. An example of such an action could be a scenario where a user wants to comment an article found online. The user inputs a text into a textbox and press the submit button. The text is then posted to the server for validation and insertion into the database. The server then responds with a new HTML file where the content of the comment list is updated with the submitted text. The browser then builds a new DOM tree from the received HTML file, which results in a reload of the page (Ashmore, D. C. 2014).

The MPA technique was developed since browsers in the early days of internet did not have the ability to asynchronously update its content. The only way to create interactive content was to let the web server update and serve HTML to the client. The downsides with MPA in interactive web applications are that the user experience is lowered by the constant reloading and that it requires a lot of client side coding on the server. Client side coding on the server means mixing coding languages with each other, which can create unreadable and unmaintainable code (Marting 2003).

In a Single Page Application (SPA) all the content updates are handled by JavaScript in the browser. In the example above, the text will still be posted to the server to be validated and stored, but with the difference that the response from the server does not contain any HTML document. Instead, the server responds with a status code indicating whether the request was successfully handled or not. The browser then acts depending on that status code and can update only the part of the HTML affected by the user interaction. In the example above, this would mean that the browser could create a new HTML tag for the comment and inject it in the comment list without reloading. This functionality was made available with new JavaScript techniques such as asynchronous loading (AJAX) and new JavaScript frameworks (Mesbah and van Deursen 2007).

The concept of SPA does not mean that the application has to have only one page. Instead, the meaning of *single page* refers to that there is only one HTML document sent from the server when the application loads and that page is accessed by requesting one main HTTP endpoint. SPA can still deliver the notion of navigating through multiple “pages” but these pages are generated by JavaScript on the client instead of the server (Mazzetti et al. 2008).

The advantages with SPA is faster loading time, better user experience and a clean separation of client side and server side code. Faster loading is achieved by the fact that the payload of requests and responses sent to the server during the applications lifetime only contains raw data and is not embedded in HTML documents. A faster loading time means better user experience since the user does not have to wait for page loading upon making requests (Mazzetti et al. 2008).

3.5 Interoperability and accessibility

Just as the internal architecture of a web GIS can impact the maintainability of the system for the developers, it can also impact the interoperability and accessibility for users. Interoperability refers to the application’s ability to interact and integrate other data sources and accessibility refers to how users can discover and browse the data exposed by the application.

3.5.1 General issues

The World Wide Web (WWW) offers unlimited discovery of data in all possible formats. These data entities all have their own semantics and underlying structure defining how the data can be used and integrated into different environments. This structure is often hidden from the consumer since the data normally goes through a number of processing steps before being presented as text or images in the browser.

The heterogeneity of data and data formats makes it a challenging task to integrate data from different data sources into one application. This problem is present also in web GIS, since spatial data and topological relations can be defined in different ways depending on different local standards in GIS architectures. There might be schematic differences in the database in which the data sets are stored. There might be semantic differences in the data sets and there might also be syntactic differences meaning that the systems use different formats for storage and distribution of data. (Ding et al. 2007)

OWS can be used to overcome the syntactic differences. The use of standardized web services means that data can be retrieved by using standard communication protocols. However, even if OWS is used to bridge the syntactic differences, the schematic and semantic differences will remain. It could very well happen that data queries performed on OWS, such as WFS filtering, produce unexpected results due to semantic differences (Ding et al. 2007).

The term Semantic Web (SW) or Web 3.0, as it is sometimes referred to, was coined by Tim Berners Lee in 2001. The idea is that data discovery should not be restricted to the scope of individual applications but that the web itself should be a giant interconnected graph linking between data based on semantic interpretations. The difference between the current web and the SW is that the current web is linking documents containing data, the SW should be able to link the data itself; regardless of what document or application scope it belongs to (W3 2016d).

3.5.2 Ontology driven architecture

Ontology driven web GIS systems has been proposed to overcome these problems and introduce semantic abilities to web GIS. The ontology describes data in a language comprehensible to both machine and humans and is a central part of SW. There exists many languages or constructs for ontologies, such as Resource Description Framework (RDF) and the Web Ontology Language (OWL) which are both standardized by the World Wide Web Consortium (W3C) (Ding et al. 2007).

Ontologies offer the advantage of creating a semantically uniform description of data and are used to link different data sets together. It can thus facilitate data discovery and make data integration and linkage between systems easier (Vaccari et al. 2012).

The use of ontology in a web GIS does not automatically mean that interoperability and accessibility issues are solved. The ontology could provide a linkage between two data sets where no semantic differences could exist, but the schema for that ontology would have to be broad enough to embrace the features and topological relations existing in both data sets (Bogdanovic et al. 2015a).

Creating common or global ontologies have been proved to be difficult and expensive. The alternative would be to adopt different ontologies but then the integration of the two data sets would imply merging, mapping or translating the ontologies to use common semantic rules (Ding et al. 2007).

3.5.3 Context driven architecure

The ability to link geospatial data based on semantics to improve accessibility within and outside the application context is a research area in itself. One novel approach is to create a Context Driven Architecture (CDA) where the user's behaviour and preferences are used to determine what data is relevant to the user (Bogdanovic et al. 2015b).

The architecture builds on extending OGC Web Services to create a supplementary service layer where user preferences can be determined and stored. This information can then be used to display the relevant geospatial data for that particular user and also create attribute based styling according to the user's interests. This requires map styling capabilities on the client, since style configurations are not stored explicitly but should be dynamically created based on the changing preferences of users (Bogdanovic et al. 2015b).

4 WGRAS application development

The application presented in this section was developed in the context of the WGRAS project at the Department of Physical Geography and Ecosystem Science in Lund. The project and the resulting application is a collaboration between the GIS-Centre and the WHO regional office for Europe in Copenhagen. It was presented for the member countries of WHO Europe in Yerevan Armenia, at a workshop for GIS for risk assessment and disaster management in December 2015.

4.1 Evaluation of requirements

In this section, the requirements listed in section 1.5 are evaluated and their impact on the development procedure for the system is assessed.

1. Open source.

The Free and Open Source Software (FOSS) community is today a vibrant and rapidly evolving community. Specialized FOSS for geospatial implementations are monitored and advocated by the Open Source Geospatial Foundation (OSGeo 2016). The decision to build the WGRAS application with FOSS is therefore opens up a good opportunity to explore and take part in this community and to take advantage of all the online resources available.

As FOSS does not automatically mean that the software is free to use without restrictions but depends on which open source license is applied, special caution has to taken and the licenses has to be checked before the software is used. For instance, the GNU General Public License (GNU GPL) state that the inclusion of a licensed resource into the own implementation means that the derived work has to use the same license as the original. This is commonly known as the copyleft principle (GNU 2016).

The MIT license is less strict and lets the user publish and sell the software under a new stricter license. The MIT license also states that the user of an MIT licensed software cannot hold the originator liable for any malfunctions in the software (MIT 2016).

For WGRAS, the inclusion of open source licensed software and technologies should therefore be done with care and awareness regarding these implications.

2. No database.

This requirement means that a file-based or cloud based storage must be used. The data must be available for the server to reach the files at any time. It also means that user credentials must be stored in a format easily parsed by the server.

3. User authentication.

User authentication could be achieved with ordinary web technologies and could be implemented in a way so that user sessions can be stored in the browser for a certain time. This allows for a better user experience since the user do not have to log every time they visit the application.

As seen in section 3.1, a role based authentication system can be beneficiary for a participative web GIS and such an implementation will therefor be considered.

4. Visualizing risk data.

This requirement can be seen as a basic requirement for the whole application. Displaying risk data could use attribute based styling of data based on for example risk index. This requirement also links to 5, as risk related spatial data can come in both vector and raster formats.

5. Visualizing raster and vector data.

This requirement means that the server needs to expose at least two different services, one serving vector data and one for raster data. Both services must be able to transform spatial data formats into formats suitable for the browser and provide a standardized communication protocol between server and client. As seen in section 3.4.2.2, OWS can be used to create a common data distribution interface for spatial data and is therefore a good candidate. Another option would be to implement a custom communication protocol based on HTTP and evaluate the performance against OWS.

6. Spatial processing.

This requirement is loosely defined but regardless, it must be evaluated if the processing should be performed on the client or on the server. In section 3.4.3.1, the differences between thick and thin clients were explained and these differences should be considered. The advantages and new possibilities of modern browser technology and client side processing possibilities should also be considered.

7. integrating external data sources

This requirement makes the techniques laid out in section 3.5 relevant. To implement a fully functional ontology based system is out of scope here, simply because of the time limit of the project. Techniques for accessing metadata and the let the system be aware of user context are interesting and does not have to be very complicated in terms of implementations.

4.2 Development Considerations

4.2.1 Map Interface

In section 3.4.3.3, the differences between Single Page Application (SPA) and Multi Page Application interfaces were explained. Using SPA techniques has the advantage of providing a good user experience since the HTML document can be partially updated and the whole page does not have to reload on user interactions.

A map application typically provides a map section, a list of some kind showing the active layers and a set of tools to interact with the map. In a SPA all of this has to be fitted into one window, just as with a desktop GIS. All the HTML also has to be rendered once the application loads. New technologies such as HTML 5 and modern JavaScript libraries makes this possible. An example of a SPA is seen in the SCALGO Live application in section 3.3.2.

If a MPA approach is used, the result would be similar to the online example of WHO e-atlas seen in 3.3.1. This approach is suitable for applications containing a lot of data and a clear separation of data categories is needed.

For WGRAS the SPA technique is probably more relevant since the required functionality is much like a desktop GIS and code separation is valuable from a development perspective.

4.2.2 Data Storage

Since no local relational or object relational database system was considered for WGRAS during development, see requirement 2 in section 1.5, the remaining alternatives were to use a cloud based storage or a local file system.

Cloud based storages has been successfully tested in GIS and web GIS implementations in various forms. The main benefits of cloud-based storage for spatial data are scalability, ready to use solutions and accessibility (Ahmed 2013). The distributed structure of cloud services makes them suitable for scalable applications, meaning that storage capabilities can be extended as the system grows. Since the service is ready-to-use out of the box, implementation and maintenance time is reduced compared to a traditional local storage alternative (Han and Feng 2015).

Cloud storages for spatial data are implemented as both traditional object relational databases and document or file based systems. However, it is still considered as relatively young and untested technique for large-scale web GIS projects. Besides, using a cloud based storage means involving a third part for managing the system and that part has to be trusted by the system administrators (Han and Feng 2015).

File based storage systems have been used in GIS for long time and are well tested. As seen in section 3.4.1, file based systems are easy to maintain and are suitable for small scale projects. The developer or systems administrator has direct access to these files and can manage them with ordinary tools available on the local system, which can be an advantage. A file based storage system can be as simple as a folder stored somewhere on the server. Different file formats can be stored together in the folder without special configurations. Due to the diversity of spatial data formats, this is an obvious advantage (Harrie and Svensson 2012).

The main disadvantage of a simple folder based storage system is that concurrent manipulation of files must be avoided. This means that multiple users cannot write changes to the files at the same time unless the server uses some kind of threading system.

4.2.3 Data Distribution

An important part of a system for distribution of spatial data is a well working and reliable map server. There exists both standardized and non-standardized solutions that can be used for serving spatial data. Non-standardized services are offered by for example Google and their web mapping API (Google 2016). These services are very popular for integrating maps into both web applications and mobile solutions.

Compared to OGC compliant services the non-standardized services are often closed meaning that the developer cannot tweak and adjust data handling on the server side. Further, non-standardized services may have the ability to consume data from standardized services, such as WMS and WFS but cannot expose such services themselves. A non-standardized service is not restricted to use standardized data formats but can implement their own, as was the case with Google and the Keyhole Markup Language (KML). This format had such qualities and became so popular that it was later standardized by OGC (Brinkhoff 2007).

Mixing both OGC compliant services and non-standardized services has become popular in web GIS. These systems are referred to as Mashups and can enhance the web application with functionality that is not available or is to advanced to implement in the local system. In web GIS for disaster management the use of mashup technology can provide flexibility and data integration from different sources. This can in turn improve the decision making process by providing integration of real time data from sources such as mobile devices (Karnatak et al. 2012).

Restrictions in modern browsers makes it impossible to request data from another domain unless some kind of proxy solution is implemented. Modern technologies make it possible to allow Cross- Origin Resource Sharing (CORS) on the server hosting the data (W3C 2016). In situations where CORS is not implemented on the external server, the requesting server can use a proxy script or JSONP techniques to achieve the same functionality (Karnatak et al 2012).

To improve interoperability and accessibility, techniques such as ontology and context driven architectures can be used. The use of ontology has proven to be successful in web GIS systems for integrating and linking to external data sources (Chung. 2015). Integrating user context into the system can provide better accessibility by analyzing user behavior (Bogdanovic et al 2015b).

OWS provides a uniform and standardized communication protocol for a web GIS. However, the rich functionality of OWS can also make it very complex and create a barrier between the data and users with little or no knowledge about the standard (Tamayo et al. 2012). To ease the use of OWS and create better accessibility, the concept of Representational State Transfer (REST) can provide a service layer where some of the complexity is sorted out. REST can also be used to create linkage between data sets that are normally isolated from each other (Maso et al. 2014).

REST principles was introduced by Roy Fielding in 2000 and can be summarized by the following: client-server interaction, stateless interaction, cache support, chaining of services, code on demand and uniform interface. The major difference with traditional service interfaces is that REST usually only make use the HTTP standard for requesting resources and those resources should be referenced by a Uniform Resource Identifier (URI) that describes the resource in an understandable way (Fielding 2000).

For WGRAS all of the techniques mentioned above are interesting. Considering the requirements listed in section 1.5, the following can be concluded:

- OWS could be used to visualize data derived from a risk analysis process, requirement 4.
- OWS could be used to visualize raster and vector data, requirement 5.
- OWS could be used to perform spatial processing through WPS, requirement 6.
- OWS paired with some kind of CORS or proxy technique can be used to integrate external data sources, requirement 7.
- Accessibility could be improved by implementing a REST service to lower the complexity of OWS services. This was not a requirement but can be seen as a general improvement of the system.

4.2.4 Spatial Processing

Spatial processing is traditionally associated to server side implementations since such processes often includes large amount of data and quite heavy calculations. However, the evolving browser technology, the use of powerful client machines and a vibrant open source community has provided new tools for spatial processing on the client side. Examples of such tools are new JavaScript libraries such as *turf* (turf 2016) and OpenLayers 3 (OpenLayers 2016) that provide lightweight spatial processing functionality for vector data.

In a web GIS, there is a clear separation of spatial data stored in various formats on the server and the visualization of that data in the browser. Spatial data formats such as GeoTIFF and Shapefile cannot be displayed directly in the browser but has to be converted into browser friendly formats. This conversion often means that the displayed data do not carry all the information available in the original file. This is especially true for the GeoTIFF format, which has to be converted to e.g. PNG or JPEG before being sent to the client for rendering. However, HTML 5 does offer access to the pixel values of the resulting image in the browser and there exists techniques for manipulating these values with JavaScript. This can be used for client style pixel classification of raster data in the browser environment (Pliutau and Prasad. 2013).

The GeoTIFF might contain altitude information mapped to each pixel but the resulting PNG image is built up with pixels with four channels, red, green, blue and alpha. These channels are used to visualize the altitude value of the original file but cannot be used to extract the altitude value. This makes it necessary for the client to request the server if the accurate altitude value is needed (Pliutau and Prasad. 2013).

Spatial processing on vector data in the client is somewhat easier as the geometry, attributes and attribute values of each feature can be encoded with the feature itself in formats such as GML. All the information about each feature is then known to the client and can be used as input to the process (OGC 2016f).

All processing made on the client uses the resources available on that specific machine, such Graphics-Processing unit (GPU) and Central Processing Unit (CPU). If client side processing is to be implemented it is therefor important to evaluate if the expected user group can be thought to provide enough computing power for the processes to run smoothly.

For development of WGRAS the following can be concluded regarding available techniques and the requirements in section 1.5:

- Techniques are available for both server and client side geospatial processing; this can be used to fulfill requirement 6.
- Accurate geospatial processing of raster data has to be done on the server.
- New HTML5 and JavaScript techniques makes it possible to alter pixel values of images retrieved from a WMS in the browser.

4.3 Architectural overview

WGRAS is implemented following the client server model laid out in section 3.4. In figure 11, the main parts of the application can be seen.

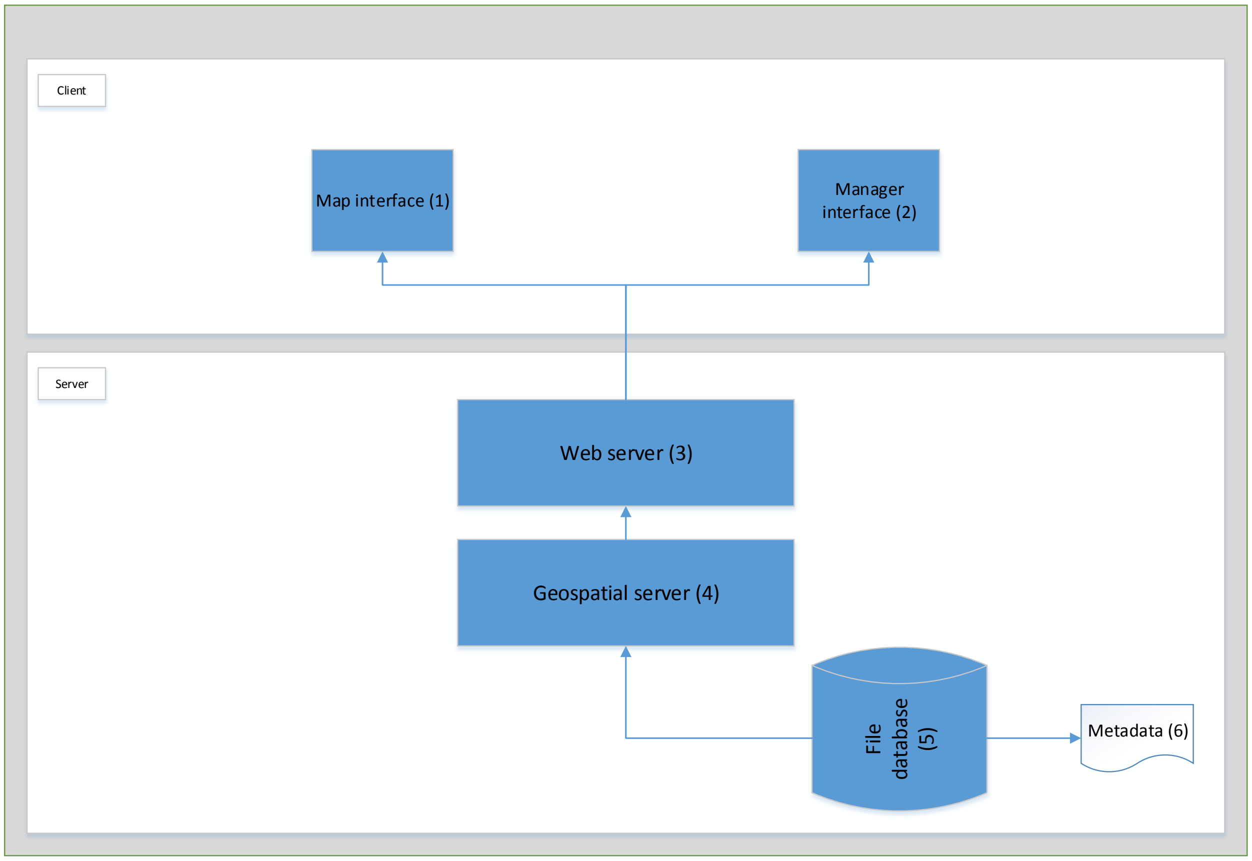


Figure 11, WGRAS Application architecture

(1) **Map:** Interface able to present map data from the internal geospatial service (4) and external services.

(2) **Manager:** Interface for admin tasks such as adding/deleting users and uploading shapefiles to the file database (5).

(3) **Web server**: Acts as an entry point to the network. Serves static files (JavaScript, HTML, CSS, images) to build up the client applications (1-2) and handles user authentication.

(4) **Geospatial server:** Serves map data from the file system (5) and performs spatial processing on both internal data sets and user provided data from the map interface.

(5) **File database:** File system to store spatial data and user credentials.

(6) **Metadata:** Metadata for spatial data entities, such as projection, description, creation date, URI etc.

4.4 Used Technologies

The following list presents the technologies used for development:

- Map server: GeoServer 2.8.0.
- Web Server: Tomcat 8 servlet container.
- Client development: Ext JS version 6.
- Server development: Java version 1.8.0_65.
- Client side mapping API: OpenLayers version 3.8.1.
- Client side spatial processing: Turf JavaScript library and JavaScript in general.
- Version handling: GIT

4.5 Map Interface

For the map interface the SPA approach was used. The whole interface is developed in JavaScript using the Ext JS library (ExtJs 2016). The interface was inspired by the SCALGO LIVE GLOBAL application presented in section 3.3.2 and the goal was to create an interface resembling common desktop GIS applications. The interface can be seen in figure 12.

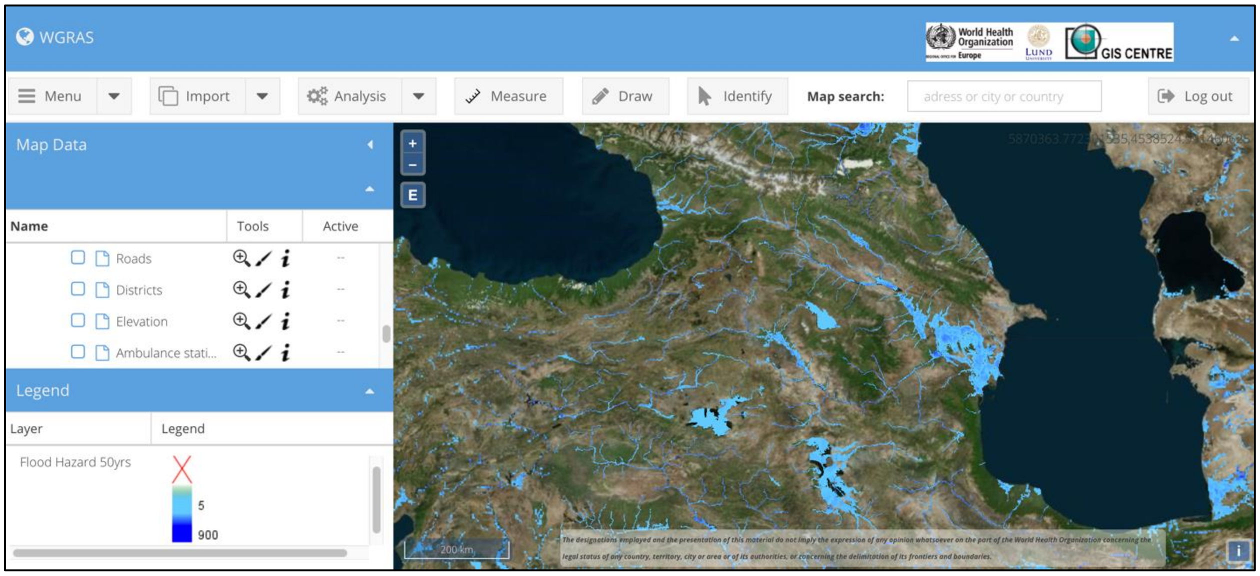


Figure 12, WGRAS Application interface

The interface contains a list of currently imported map data to the left. In this list, there are tools for visibility, zooming to data extent, styling, toggle data set information and activating/deactivating the layer. Below this list there is legend panel showing legends for all the visible map layers. The top bar contains buttons for activating different tools and the main map panel is in the middle.

4.6 User authentication

The system provides two roles for users. The Admin role can be assigned to users that should have possibility to use the manager interface. This interface is not presented in detail in this thesis but provides functionality for adding/deleting users and uploading map data to the system. The user role provides access to the map interface only. User credentials are stored in a file on the server and all passwords are hashed with a hashing algorithm that is non-reversible. This provides security in the case credentials were stolen since hashed passwords cannot be used to login.

4.7 Data Storage

For WGRAS, a file-based storage was chosen for storing both spatial data and user credentials. This file system resides in a location accessible only by the server. Access rules are set to strict, meaning that there exists no public URL pointing to the folder; instead, the data is only reachable by passing through the server pipeline.

Vector data is stored in shapefiles. This format is in itself a type of hybrid storage structure with three main files .shp, .shx and .dbf., where the .dbf file is structured as a table in a relational database. Each entry in the .dbf file is linked to a feature stored in the .shp file (ESRI. 1988).

Raster data is stored in Tagged Image File Format (TIFF) files. The extension of TIFF called GeoTIFF is used to store additional metadata such as projection, reference system and other information needed to create a meaningful representation of the data.

4.8 Data distribution

The architecture for data distribution in a web GIS application affects accessibility and interoperability and also data integrity and overall security. This requires a standardized communication protocol to be used between the server and client.

4.8.1 GeoServer

For WGRAS an OGC compliant map server implementation was chosen. During the development phase other map server systems where considered and a comparison can be seen in table 1. The table should not be seen as a record over all the functionality offered by the server systems; instead, the comparison was done using the needed functionality according to the requirements.

Table 1, Comparison of open source web map server software

	Functionality						
Server	**OGC map serving (WMS, WFS, WCS)**	**OGC map processing (WPS, WFS-T)**	**Cartographic Styling**	**Reprojection**	**Spatial processing**	**Printing**	**REST**
Mapfish	Some				Some	Yes	Yes
Geomajas	Some				Some	Yes	Yes
Qgis server	Yes		Yes				
GeoServer	Yes	Yes	Yes	Yes			Yes
MapServer	Yes		Yes	Yes			

As seen in table 1, GeoServer is fulfilling most of the needed functionality. GeoServer is a well known and widely used server implementation that is able to serve map data in a variety of formats and is fully OGC compliant (GeoServer 2016).

GeoServer was configured to work as a pure map server in the system. This means that this service layer is not responsible for user authentication but only acts as an endpoint for map requests. User authentication is instead handled by the web server, which provides authentication for the map server. The typical pattern for requesting map resources as seen in figure 4 in section 3.4.2.2 is used throughout the request chain.

4.8.2 GeoServer REST API Extension

To simplify extraction of metadata from resources distributed in WGRAS, the REST interface of GeoServer was extended with additional functionality. This section explains the GeoServer REST interface and the extensions created for WGRAS.

The REST interface provided by GeoServer can be used to extract metadata for available resources and can be seen as a complement to the OWS *GetCapabilities* request. The interface uses the following HTTP methods:

- GET to extract resources
- PUT to update existing resources
- POST for creating resources

- DELETE for deleting resources

GeoServer structures its data in *workspaces* for grouping similar resources and each resource is bound to a specific *data store*. The REST interface maps URIs and resources following this structure and the URI for extracting metadata for a layer looks as follows:

Table 2, URI for built in REST API in GeoServer

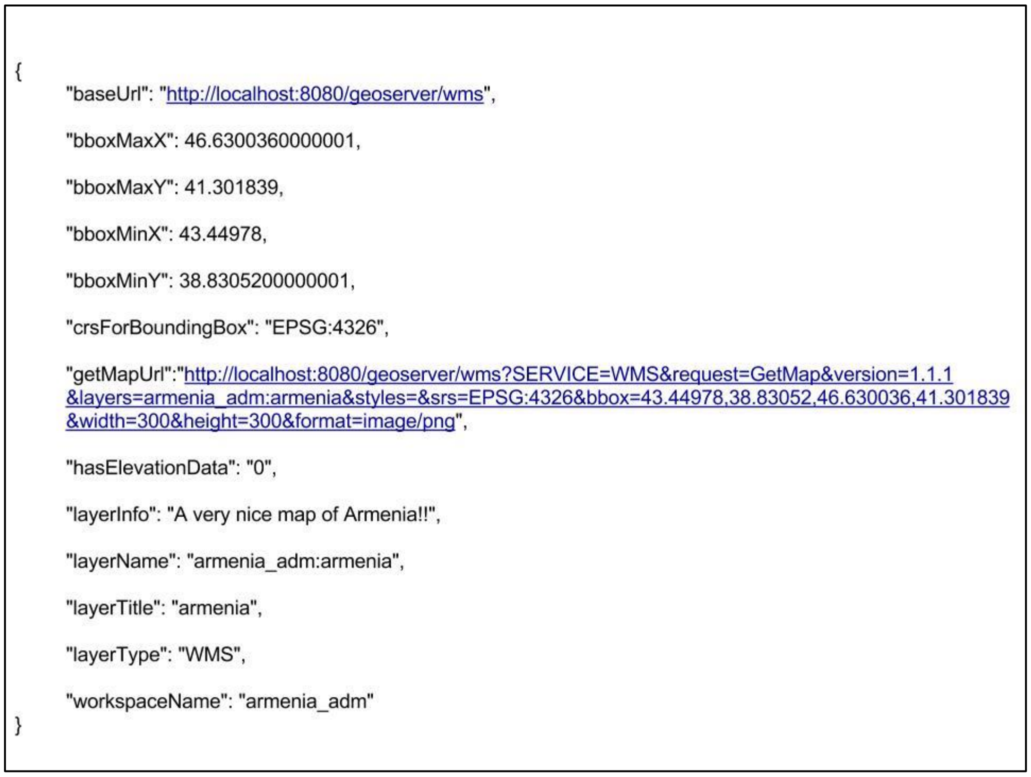
http://localhost:8080/geoserver/rest/workspaces/armenia_adm/datastores/armenia_adm/featuretypes/armenia.xml	Retrieves metadata for a specific resource

A GET request to this URI will produce an XML document with information about the spatial reference system, attributes, bounding box etc. To extract this information, the user needs to know the workspace name, the name of the data store and the name of the feature type. This information is not necessarily known to the user trying to extract the data and therefore a few simplified REST endpoints were created and integrated in the system. An excerpt of these endpoints can be seen in table 2.

Table 3, URI for REST API extensions in WGRAS

http://wgras.gis.lu.se/rest/layers	Lists all available metadata for all resources
http://wgras.gis.lu.se/rest/layers/wms	Lists all available metadata for WMS resources
http://wgras.gis.lu.se/rest/layers/wfs	Lists all available metadata for WFS resources
http://wgras.gis.lu.se/rest/layers/wms/<layerName>	Retrieves metadata for the specified WMS resource
http://wgras.gis.lu.se/rest/layers/wfs/<layerName>	Retrieves metadata for the specified WFS resource

A GET request to describe a specific resource produces the representation seen in figure 13.



```
{
        "baseUrl": "http://localhost:8080/geoserver/wms",

        "bboxMaxX": 46.6300360000001,

        "bboxMaxY": 41.301839,

        "bboxMinX": 43.44978,

        "bboxMinY": 38.8305200000001,

        "crsForBoundingBox": "EPSG:4326",

        "getMapUrl":"http://localhost:8080/geoserver/wms?SERVICE=WMS&request=GetMap&version=1.1.1
        &layers=armenia_adm:armenia&styles=&srs=EPSG:4326&bbox=43.44978,38.83052,46.630036,41.301839
        &width=300&height=300&format=image/png",

        "hasElevationData": "0",

        "layerInfo": "A very nice map of Armenia!!",

        "layerName": "armenia_adm:armenia",

        "layerTitle": "armenia",

        "layerType": "WMS",

        "workspaceName": "armenia_adm"
}
```

Figure 13, Response from the REST API upon requesting a layer description

The most important here is the *getMapUrl* entry, which is dynamically generated for each resource and can be used to request the map image. The *layerType* denotes if the resource is a WMS or WFS map. Metadata can be generated in both XML and JSON, which makes it readable by any browser client. The intent with this implementation is to create a streamlined request flow for clients that are not aware of the internal data structure of WGRAS.

The main differences between using the REST extension compared to the built-in GeoServer REST API are listed below. This comparison assumes the requesting part has no knowledge of the internal data structure of WGRAS and therefor have to explore the structure through HTTP requests.

- By using the extension, it is not necessary to know the data store name or the workspace name of the resource. If using the GeoServer REST API, the workspace name, store name and layer name has to be known.
- The representation contains a GetMap URI that can be used directly to request the map image or vector feature in the case of WFS. This is not provided by GeoServer REST API.
- To extract the information shown in figure 13, the client needs to perform one HTTP GET request. To extract the same information with the built in GeoServer REST API, four consecutive HTTP GET requests are needed.

The OWS *GetCapabilities* operation can also be used to extract the information shown in figure 13. The difference is that the client must be familiar with the OGC standard. Besides, the *GetCapabilities* operation can only return XML while the extension can return both XML and JSON formatted metadata.

4.8.3 Data Import

Two different implementations were done for data import. One tool imports data from the internal data storage. This tool was included in the first release of the system in December 2015. The second tool is for creating connections to external OGC compliant services and is still under evaluation.

The import tool designed for the internal storage is used for initial loading of map data and is also available under the import menu if the user needs to load additional data. As the application loads, the information extracted from the capabilities document is stored in local storage in the browser (W3Schools 2016). This information is used throughout the user session and can be retrieved at any time. If the information for some reason is lost, a backup system is alerted which immediately retrieves new information from the server. In this way, the client knows what layers are available and can build up *GetMap* requests as well as show available metadata without having to make unnecessary requests to the server during the user session.

Imported data is organized by category in the layer list. A category is mapped to a workspace in GeoServer and thus each category can contain many layers. This is shown in figure 14.

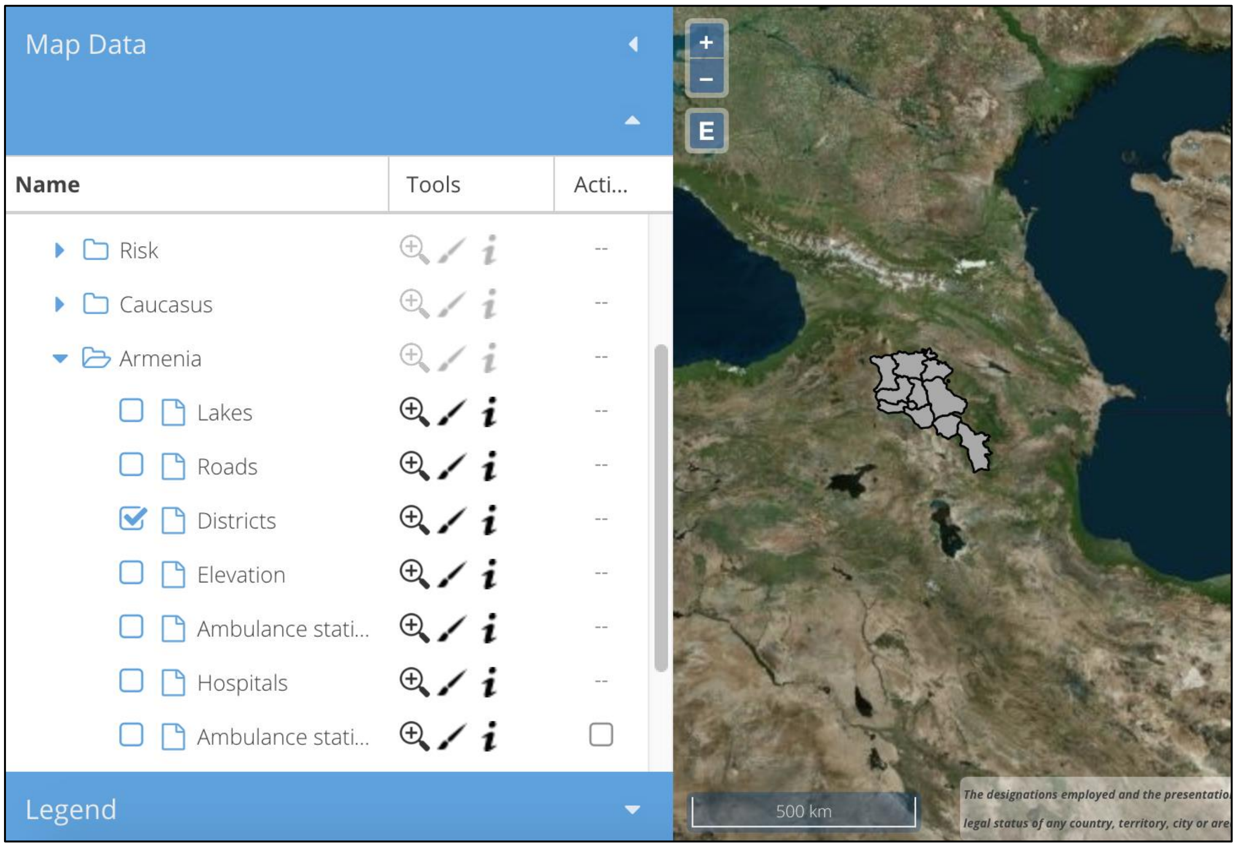


Figure 14, Layer categorization in WGRAS

To create a connection to an external service, the user can activate the *Create Connection* tool located under the import menu. As mentioned above this tool is still under evaluation and was not included in the first release of the system. The process is described below and can also bee seen in figure 15.

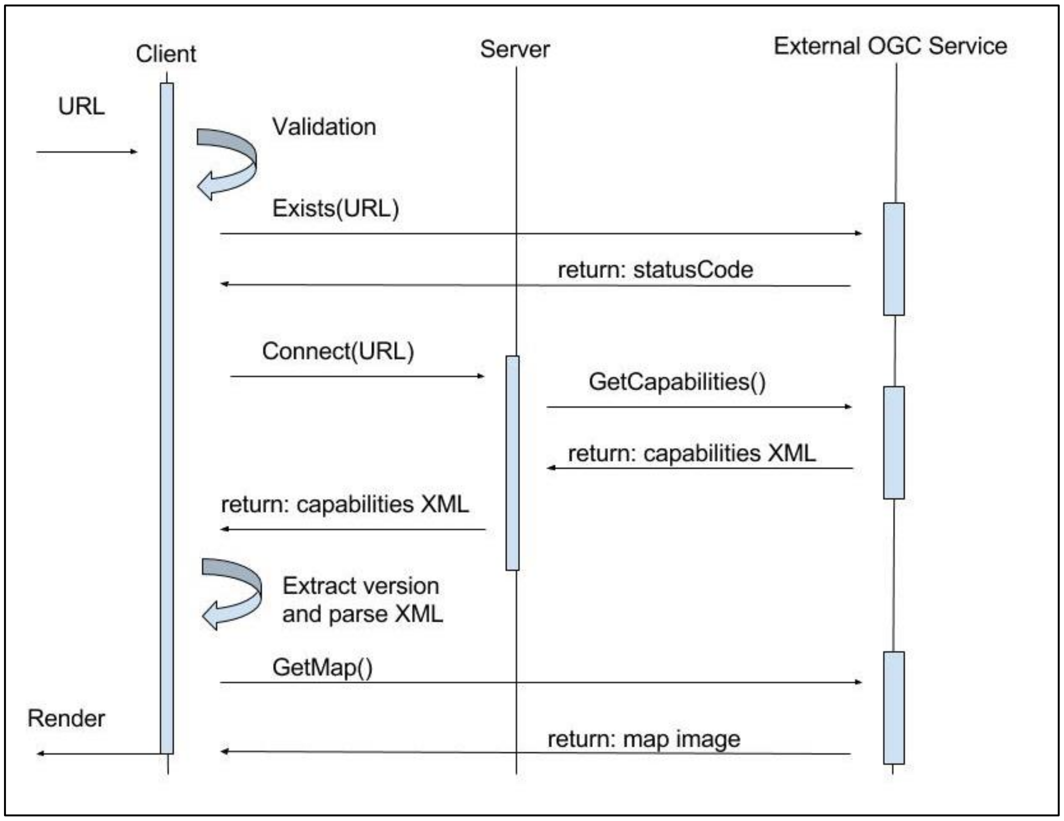


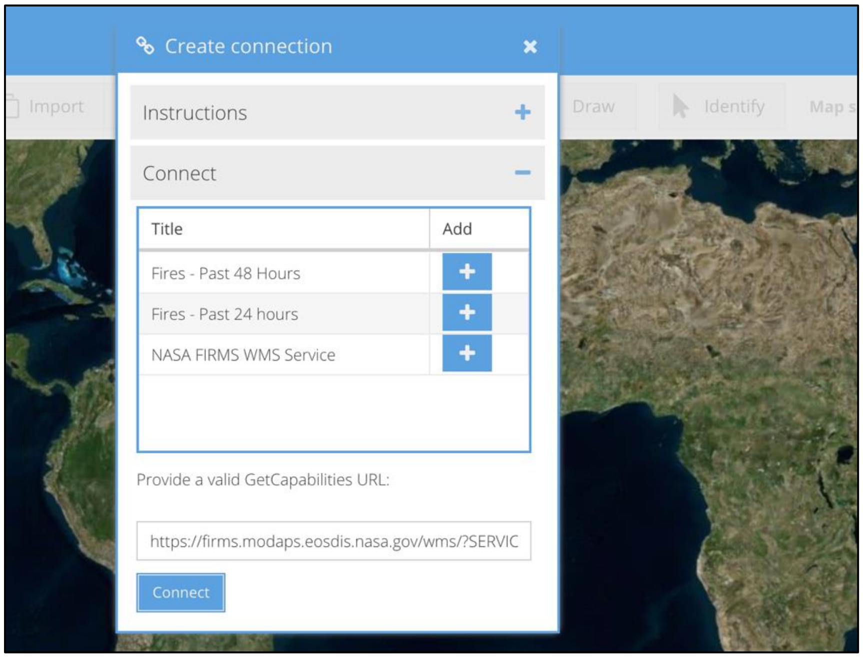
Figure 15, Request sequence for connecting to an external OGC compliant map service

The process of connecting to an external service requires a valid *GetCapabilities* URL for the OGC compliant WMS service. The URL is provided by the user and is then validated by the client in two steps.

First, the URL is compared to a regular expression to make sure no harmful input is provided. The second validation is done by sending a HTTP request to the URL and see if the server is responding. Because of restrictions in cross-domain resource sharing between browsers, this request is sure to fail. The intent is not to get a successful response but to check that there is a server responding at the given URL. If the response contains a status code, it is indicating that the server exists and the process can continue (W3 2016e).

In the next step, the URL is sent to the web server, which performs the request to the external OGC service and sends back the capabilities document to the client. Here the web server is acting as a proxy to overcome the problems of cross origin resource sharing in the browser.

The client parses the capabilities document from the external service by first determining the service version. This is important since the capabilities document looks different depending on version and the client needs to adopt the right reading module to parse the XML document. The available layers are then listed and presented with an 'add' button that can be used to add the layer to the map, see figure 16.


Create connection

Instructions

Connect

Title	Add
Fires - Past 48 Hours	+
Fires - Past 24 hours	+
NASA FIRMS WMS Service	+

Provide a valid GetCapabilities URL:

https://firms.modaps.eosdis.nasa.gov/wms/?SERVIC

Figure 16, Result from requesting an external OGC compliant service

4.9 Spatial Processing

The requirements expressed by WHO do not specify any specific spatial processing to be implemented in the system, however during development the need for specific geospatial processing tools occurred. In this section the implementation of such processes in WGRAS are explained.

4.9.1 Flooding analysis

The WPS interface is used for providing standardized geospatial processing. The standard does not mandate which processes should be exposed by the service but rather specifies the way a process should be described and also executed.

In WGRAS, WPS processes are used to perform geospatial processing upon user interaction in the map interface. GeoServer exposes a great number of geospatial processes by default, such as buffer, intersection, polygon extraction from raster data etc. These processes are all OGC compliant meaning that every process supports the *GetCapabilities*, *DescribeProcess* and *Excecute* operations. The existing WPS interface in GeoServer can therefor be used to create other useful OGC compliant processes using the existing structure.

Two such extensions were created for a flooding tool which can be used to visualize the water level in the landscape based on user input. This can be seen in figure 17.

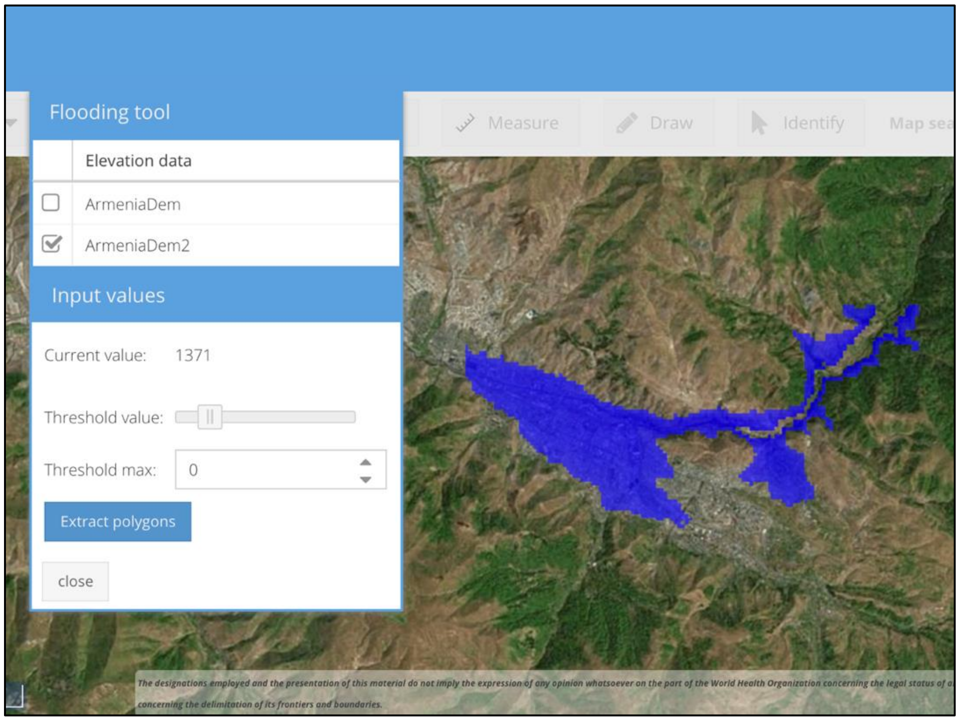


Figure 17, Flooding tool, shows water level rise in the landscape based on user input.

In figure 17, a threshold value for the raise in water level is provided by sliding the slider or altering the number input box. The next step is shown in in figure 18, where the polygon for the area is extracted and saved as a new layer on the map.

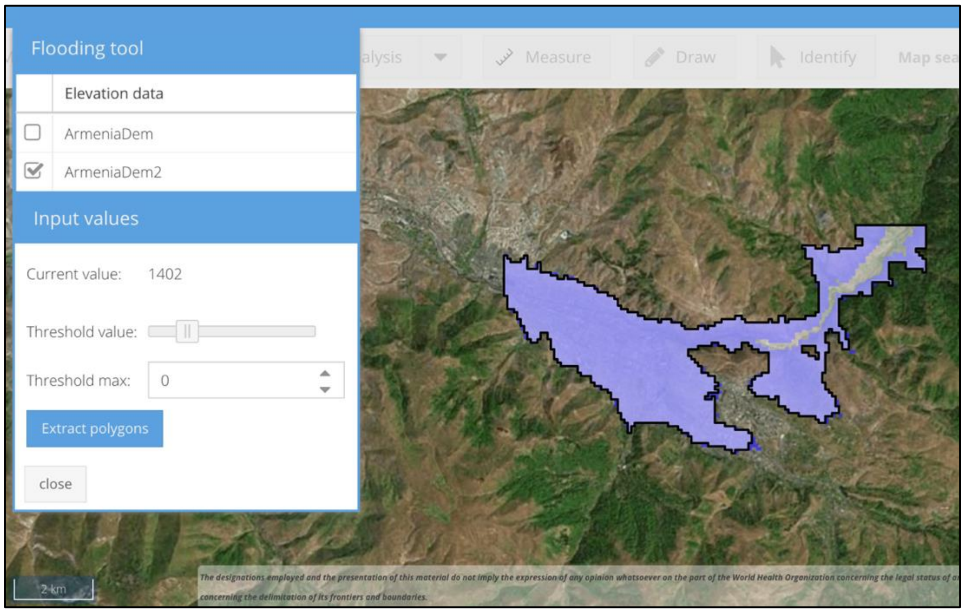


Figure 18, Extracted polygon based on user input to the flooding tool.

This tool uses a combination of client side and server side processing. The tool is implemented to handle classification of pixels based on a threshold in a global search algorithm, and a local search based on a threshold value and a location. The global search method simply looks for pixels with elevation values less than the threshold, while the local search algorithm searches for adjacent pixels (8-nearest neighbors) with a value less than the threshold value.

The main steps for this tool in global search mode are listed below:

1. Client requests a PNG image of a Digital Elevation Model (DEM) from GeoServer. The DEM is color coded in grey scale with each channel (Red, Green and Blue) ranging from 0 to 255.
2. Client requests the min and max elevation values of that DEM from the server and sets input limits accordingly.
3. The client interpolates the user input to a value between 0-255 and changes the pixel value to a blue color if the current pixel value is less than the input value.
4. When the user requests the polygon, the client makes a request to the server to perform the extraction based on the threshold value.
5. The server process performs a global search on the GeoTIFF and finds the pixels with an elevation value less than the threshold. The pixels with a value less than the threshold are included in the polygon.
6. The polygon is returned as vector data to the client.

The main steps for this tool in local search mode are listed below:

1. Client requests a PNG image of a Digital Elevation Model (DEM) from GeoServer. The DEM is color coded in grey scale with each channel (Red, Green and Blue) ranging from 0 to 255.
2. Client requests the min and max elevation values of that DEM from the server.
3. Input consists of: a) a threshold value for increase in water level and b) a location on the map from where the water level should increase.
4. The client interpolates the user input to a value between 0 and 255 and performs a nearest neighbor search in all eight directions. If the pixel value is less than the threshold it is given a blue color.
5. When the user requests the polygon, the client makes a request to the server to extract the polygon from the GeoTIFF.
6. The server process performs a search on the GeoTIFF with the same algorithm as in 4 and finds the pixels with an elevation value less than the threshold. The found pixels are included in the polygon.
7. The polygon is returned as vector data to the client.

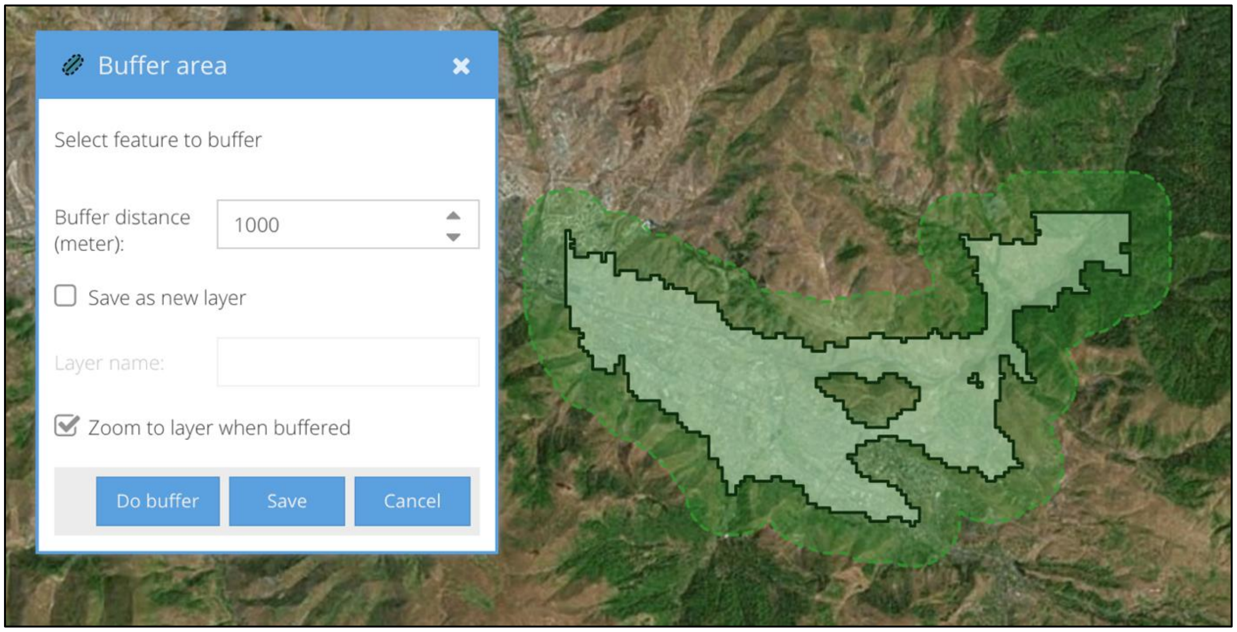
The algorithm used in step 4 and step 6 is called flood fill algorithm and searches all eight neighboring pixels (Nosal 2008). The pseudo code is seen in figure 19.

```
Flood-fill (node, currentValue, thresholdValue):
if currentValue is equal to thresholdValue, return
Set Q to the empty queue
Add node to the end of Q
While Q is not empty:
    Set n equal to the first element of Q
    Remove first element from Q
    If the value of n is equal or less than thresholdValue:
        Let n belong to polygon and mark n as processed
        Add all eight neighboring pixels to end of Q, if they have not been processed yet
Return
```

Figure 19, Pseudo code for Flood fill algorithm used in the flooding tool.

4.9.2 Buffer

The *Buffer* tool is using the WPS process *Buffer* available in GeoServer. The process requires two parameters, the geometry to perform the buffer on and the buffer distance. Figure 20 shows the interface for the tool and the resulting polygon from a buffer operation performed on polygon extracted with the *Flooding* tool.



Buffer area

Select feature to buffer

Buffer distance (meter): 1000

Save as new layer

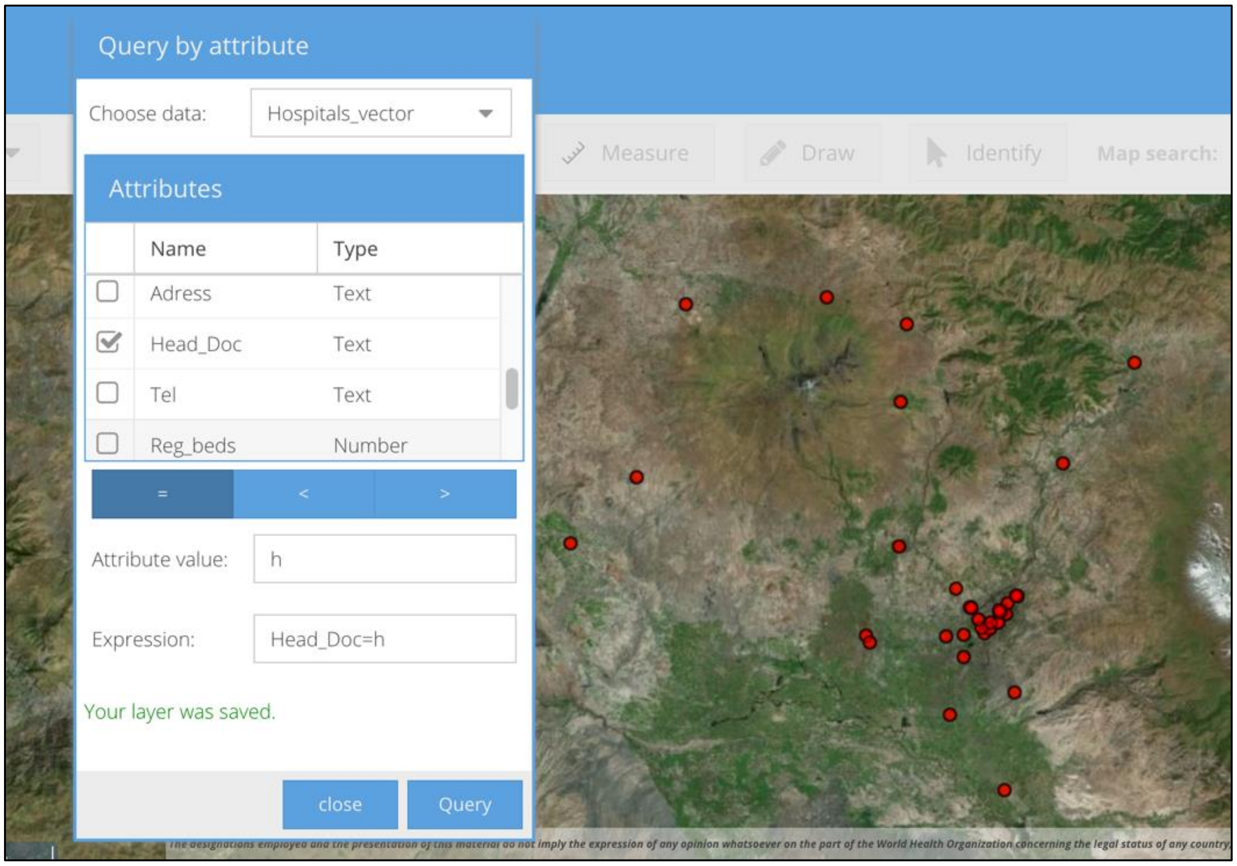
Layer name:

Zoom to layer when buffered

Figure 20, Buffer operation performed on a polygon.

4.9.3 Query by attribute

The *Query by Attribute* tool is implemented as a client side process. The process supports querying attributes of string and number data types. The process is performed by reading the attributes of the layer and the data type for each attribute is determined. If the attribute is of type string (text) the less than and greater than operators are disabled, since a string only can be searched for equality. Each feature is then iterated over and attribute values for the chosen attribute are compared to the user input. Found features are then stored in an array and added to a new layer. The tool is tested for layers with up to 700 features and response time was still very short, less than 150 Ms. Figure 21 shows the interface for this tool.


Query by attribute

Choose data: Hospitals_vector

Attributes

	Name	Type
☐	Adress	Text
✓	Head_Doc	Text
☐	Tel	Text
☐	Reg_beds	Number

Attribute value: h

Expression: Head_Doc=h

Your layer was saved.

The designations employed and the presentation of this material do not imply the expression of any opinion whatsoever on the part of the World Health Organization concerning the legal status of any country

Figure 21, Query by attribute tool

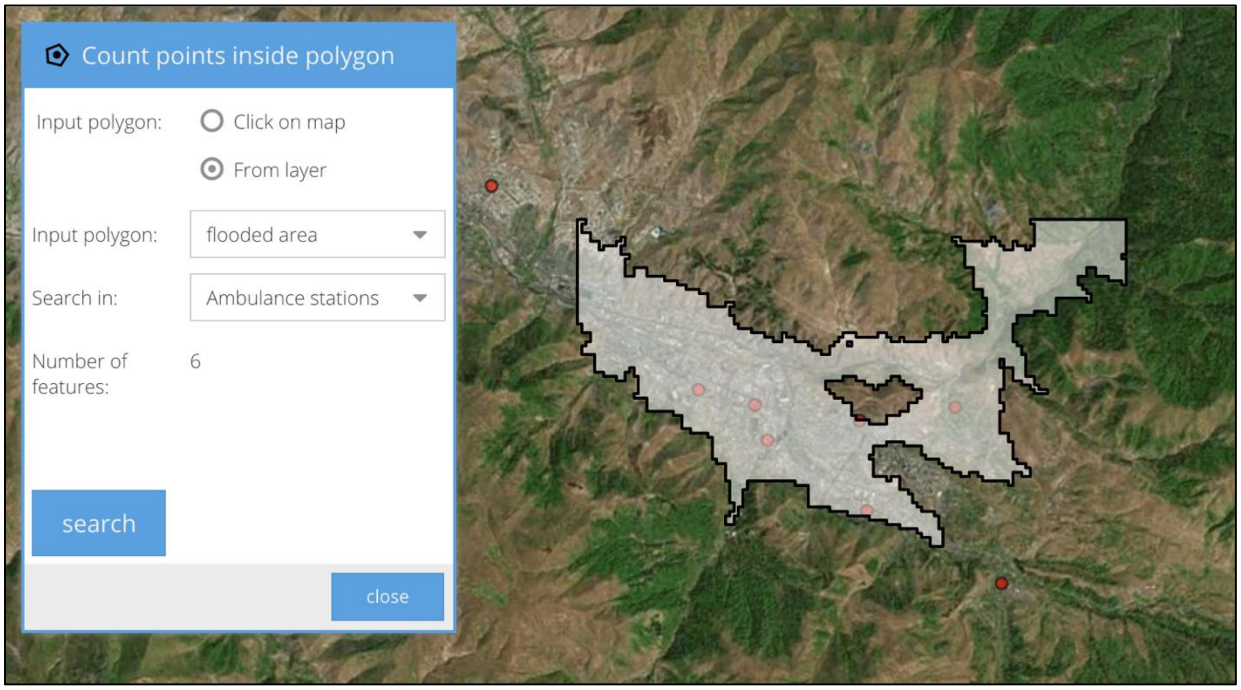
4.9.4 Point in polygon

The *Point in Polygon* tool is implemented as a client side process. For this tool the JavaScript library *turf* was used (Turf 2016a). The function used to check if a point is inside a polygon is called *inside* and is defined as follows:

turf.inside(Feature.<Point>, Feature.<Polygon|MultiPolygon>) - Returns Boolean true if the Point is inside the Polygon; false if the Point is not inside the Polygon

Feature.<Point> is a GeoJSON encoded point feature and Feature.<Polygon|MultiPolygon>) is a GeoJSON encoded polygon or multipolygon. (Turf 2016b)

Figure 22 shows the interface for this tool.


Count points inside polygon

Input polygon: ◯ Click on map [✓ From layer]

Input polygon: flooded area

Search in: Ambulance stations

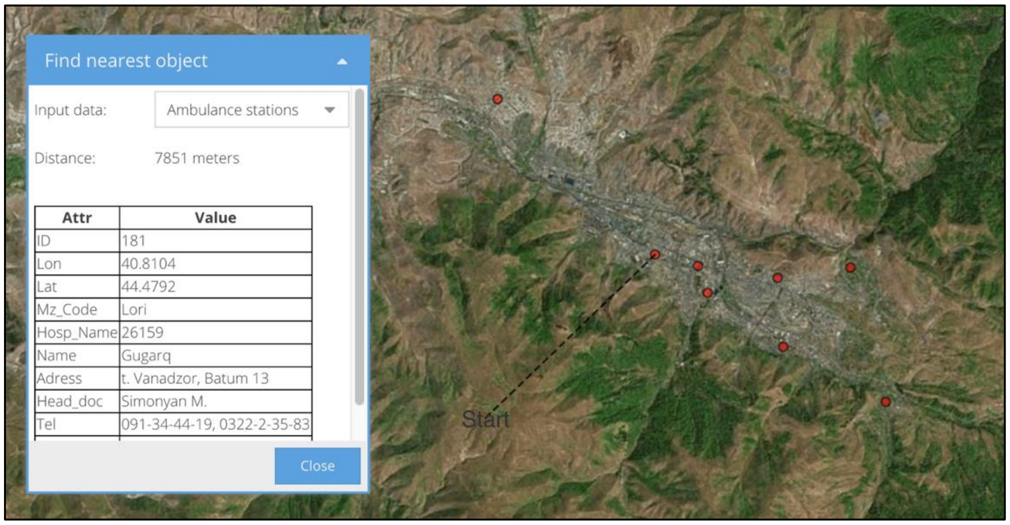
Number of features: 6

Figure 22, Point in polygon tool

4.9.5 Other tools

Tools for searching nearest object, measure distance and area, map search and styling were also implemented.

The *Nearest Object* tool was implemented as a client side process. It takes two inputs; a layer to search in and a location to search from. The process iterates over the features in the layer and calculates the distance from each feature to the given location. If stores the feature if its distance is less than the smallest found so far in the iteration. The tool then displays the distance and the attributes for the nearest object. This is shown in figure 23.


Find nearest object

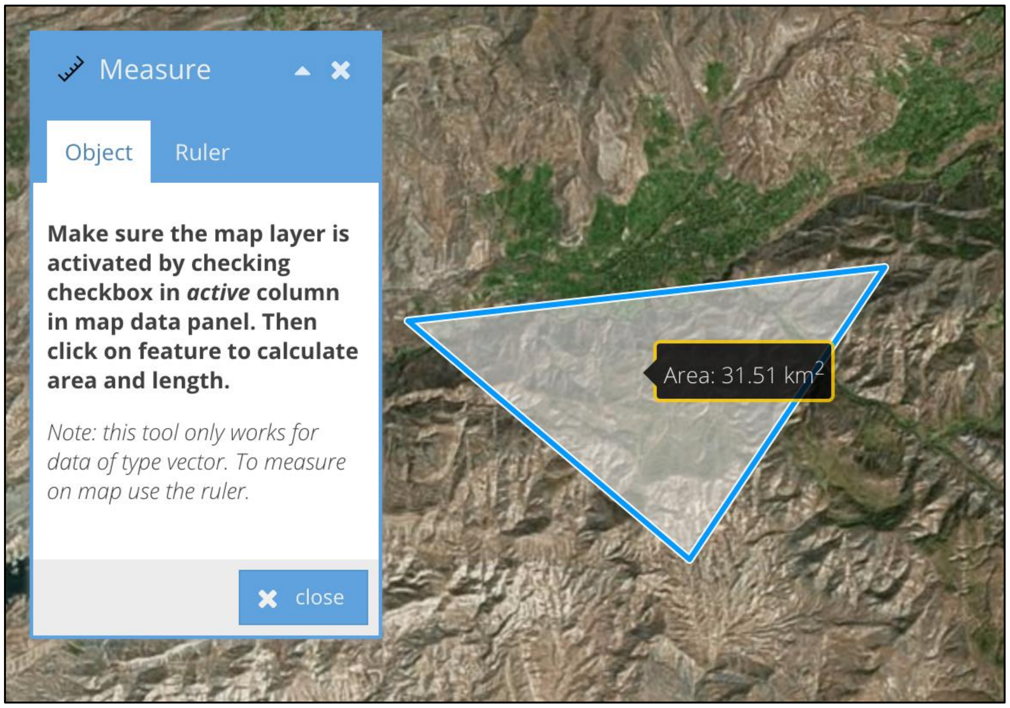
Input data: Ambulance stations

Distance: 7851 meters

Attr	Value
ID	181
Lon	40.8104
Lat	44.4792
Mz_Code	Lori
Hosp_Name	26159
Name	Gugarq
Adress	t. Vanadzor, Batum 13
Head_doc	Simonyan M.
Tel	091-34-44-19, 0322-2-35-83

Figure 23, Nearest Object tool

The *Measure* tool is also implemented as client side processes. The tool can measure either the length or area of an existing object on the map or by drawing a line or polygon on the map. The basic spatial processing functionality is provided by the client-side mapping library OpenLayers 3 (OpenLayers 2016). Additional functionality was added to provide a nice presentation of the result on the map. This can be seen in figure 24.


Measure

Object | Ruler

Make sure the map layer is activated by checking checkbox in *active* column in map data panel. Then click on feature to calculate area and length.

Note: this tool only works for data of type vector. To measure on map use the ruler.

Figure 24, Tool for measuring distance/length and area.

The *Map Search* tool is provided as a search box in the interface and uses the Nominatim API that performs a search on OpenStreetMap data. The API accepts a HTTP GET request with the search string provided as a parameter in the URL (Nominatim 2016). A successful response contains a bounding box for the found location, a name of the object and a number of other parameters. The tool parses this response, pans the map to the bounding box, and places a marker on the map displaying the name. This can be seen in figure 25.

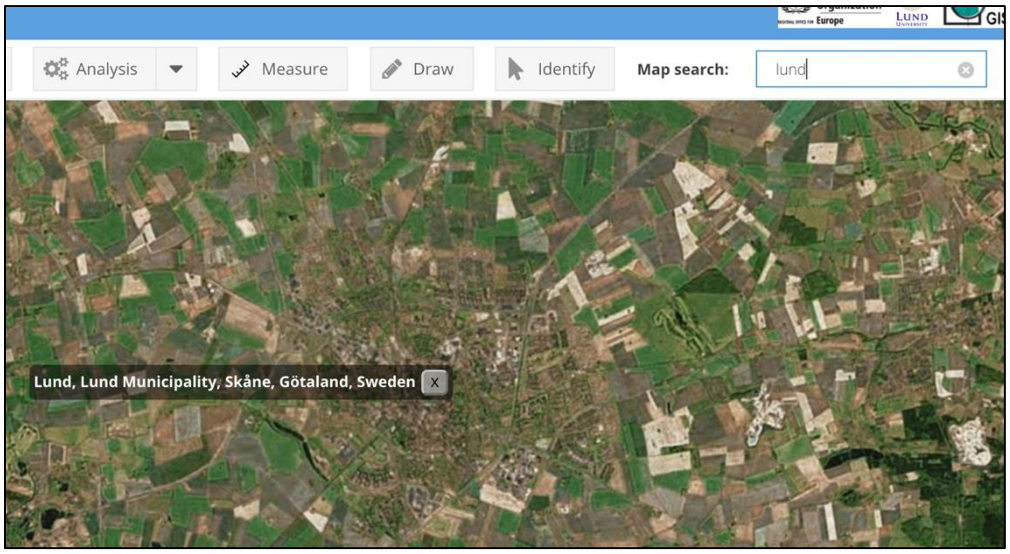
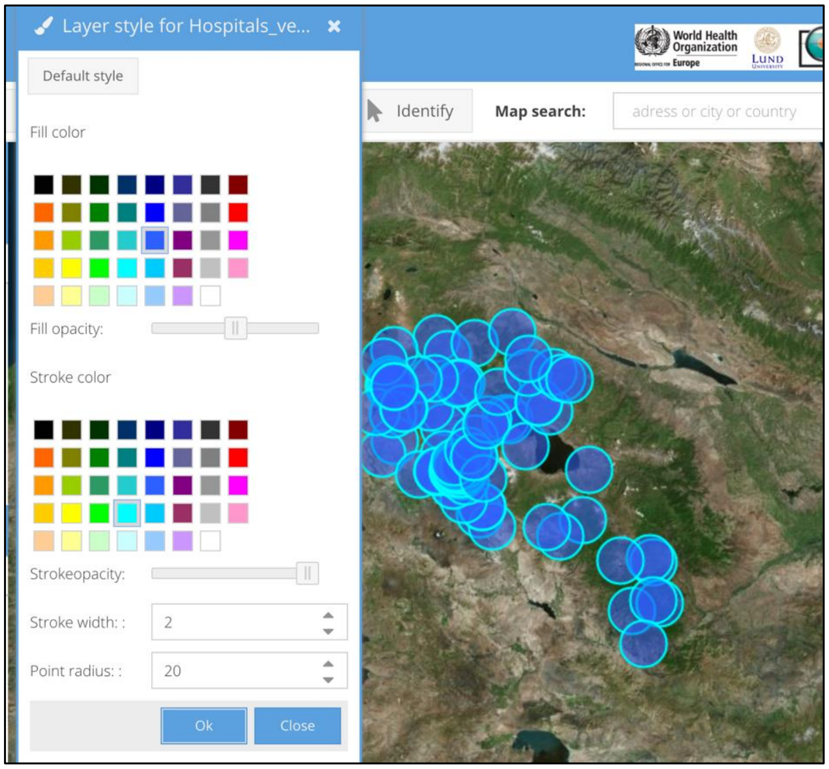


Figure 25 Map search tool.

The *Layer Style* tool provides functionality for styling raster and vector data in the client. Raster data can only be styled by changing opacity of the image while vector data offers more extensive styling. Styling raster data on the client is complicated as it involves manipulation of pixel values. (Pliutau and Prasad 2013). In WGRAS, raster data is instead styled server side by using Styled Layer Descriptor (SLD) which is supported by GeoServer. Styling of vector data is done by utilizing the built in styling functionality of OpenLayers (OpenLayers 2016). The interface for styling can be seen in figure 26.



Stroke width: : 2

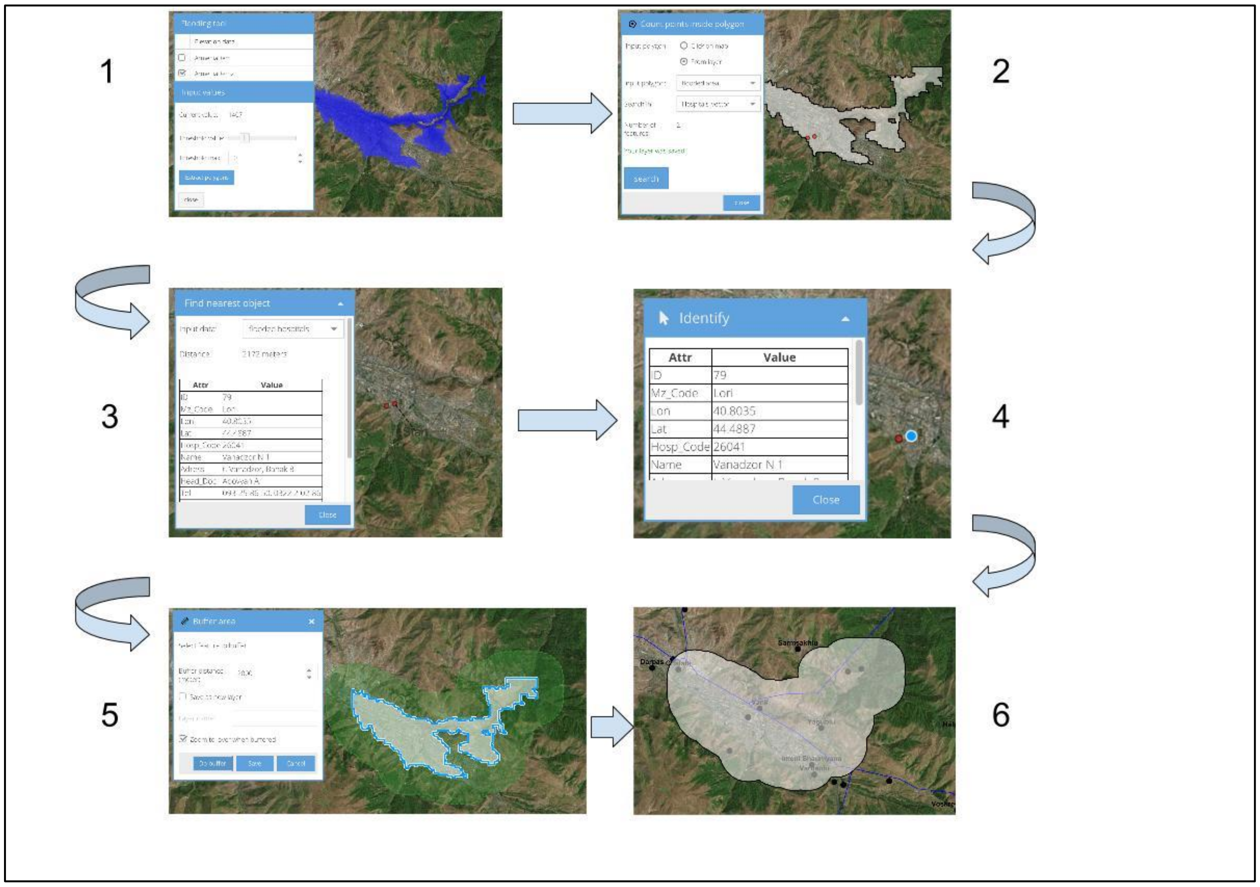
Point radius: : 20

Figure 26, Styling of vector features with the Layer Style tool.

4.10 Sample User Scenario

In this section a sample user scenario is demonstrated. The WGRAS system was developed for visualizing risk data and performing risk analysis in the context of disaster management. The scenario therefor depicts a situation where a user wants to find all hospitals within a flooded area. Among these hospitals the user also searches the name of the head doctor and the telephone number of a certain hospital. The user searches for the closest hospital in this selection from a point on the map where he/she knows there is an evacuation point. In the last step a buffer zone is created around the flooded area. This can be seen in figure 27 and the workflow is described below.

1. The flooded area is visualized by inserting a rise in water level. The flooded area is shown in blue.
2. The polygon for that area is extracted and saved as a new layer. The Point in Polygon tool is then used to search for points (hospitals) inside the polygon. The found points are saved into a new layer.
3. The Identify tool is then used for showing the attributes of the points. In this list, the head doctors name and telephone number can be found. If the name of the doctor is already known the *Search by Attribute* tool can also be used in this step.
4. A buffer area is extracted around the flooded area using the *Buffer Area* tool.
5. This polygon is then shown with a road layer and a layer showing villages to visualize the impact of the flooded area with a buffer zone.



Attr	Value
ID	79
Mz_Code	Lori
Lon	40.8035
Lat	44.4887
Hosp_Code	26041
Name	Vanadzor N 1
Adress	[illegible]
Head_Doc	[illegible]
Tel	[illegible]

Figure 27, Sample user scenario in WGRAS

5 Discussion

In this section, the results from the implementation are discussed based on the requirements in section 1.5.

5.1 Open Source

All the techniques used in this project are licensed as open source software. The most restrictive license used is the GPL v3, which states that if the software is to be sublicensed, it also has to use the GPL v3 license. The source code of such a software also has to be publicly available. This has been achieved by storing the source code on a public domain called GitHub during development.

Developing a whole system with open source software means merging many different coding libraries and systems together into one. During development, this has been somewhat difficult since some of these techniques are not fully compatible with each other. The many JavaScript libraries used in WGRAS sometimes caused namespace conflicts meaning that workarounds had to be implemented by tweaking the source code of those libraries.

Documentation of coding libraries was sometimes poor or unorganized which slowed down development. For the OpenLayers library, the documentation is rich but quite hard to navigate. In these cases, information had to be searched in other places such as development forums on the web.

The positive side of using open source software for WGRAS is that the system can be extended and work can continue outside the scope of the project. Since open source software is free and accessible to everyone, the learning curve for mastering the techniques in the WGRAS project is not as high if it was developed with proprietary software.

During the project and the development process, new GIS professionals were involved. It then became clear that a switch from the Java framework to Python for server development could be beneficial for the project. Python is more widely used and more familiar in the GIS community and the use of Python could therefor lower the barrier for other people to get involved.

5.2 Data Storage

The decision to use a file/folder based storage in WGRAS depended solely on the requirements from WHO. The advantages associated with this type of storage only applies to small systems where little or no spatial processing is to be done. A spatial database offers a substantial increase in processing power and speeds up data distribution in the system.

During development, this meant that a substantial amount of time was spent on finding ways around problems that could easily be solved with a spatial database. One such issue was the heavy use of WPS for spatial processing in GeoServer. The use of such processes are streamlined and very easy to use in a database like PostgreSQL with PostGIS.

The possibility of storing user defined maps were investigated but the idea was abandoned early on in the project. The problem was to write information to the files in a thread safe way. This had also been possible with the integration of a database, since GeoServer is built for database integration.

5.3 User Authentication

User authentication was successfully implemented in WGRAS. No major difficulties occurred even though storage of user credentials would have been easier with a database. If data storage is migrated to a database, tasks such as creating, updating and deleting users in the system will be more secure and easier to implement.

5.4 Visualizing Risk Data

Visualization of risk data in WGRAS is accomplished by the ability to visualize both vector and raster data without restrictions on what category or origin the data has. The system can serve data from its internal storage and once the tool for creating external connections has been tested it can also be fully integrated into the system. The only restriction is that the external service has to be OGC compliant. Further development could be done to support other sources as well.

The styling tool in WGRAS currently only supports layer based styling meaning that all features in a layer are given the same style. Additional development could be done to integrate attribute based styling, which could improve the analysis process in the system. With improved storage capabilities, user defined styles could also be stored for later retrieval.

Styling of raster data is done on the server through the use of Styled Layer Description (SLD). The process of creating SLD styles is time consuming and provides little flexibility and user interaction. Since it is very hard to achieve effective and good styling of raster data on the client a possibility would be to develop a tool for creating SLD on the client and upload the style and apply it to a map layer through the REST API in GeoServer. New and emerging techniques for client oriented styling of raster data through pixel manipulation is also a possibility (Pliutau and Prasad. 2013). This technique was tested in the flooding tool with good results but has to be further evaluated.

5.5 Spatial Processing

WGRAS uses both client side and server side spatial processing. Smaller processes such as measuring length and area and searching for points inside polygons was successfully achieved with the turf JavaScript library. The advantages are fast processing time and easy development. The decision on where to perform the process was taken during development since it required a lot of testing. It was also considered that all the users in the targeted group might not have the same processing power at hand.

The flooding tool as seen in section 4.9.1, used some experimental techniques such as pixel manipulation in the browser. This technique is based on HTML5 and makes use of the canvas for extracting image data from the raster image and perform pixel classification.

The combination of server side WPS and in-browser manipulation of raster data is introduced as a new possibility for web based geospatial processing. The ability to give instant visual feedback based on user input, means that the user can get an idea of the final result before initializing the process on the server. This could be used to reduce trial and error behavior when using the tool and thus reduce the load on the server.

Certain browsers such as Internet Explorer do not allow extraction of image data and therefor the tool suffers from browser incompatibility, which is a known issue in web development. The tool was successfully tested on the major browsers such as Chrome and Firefox and the application also warns the user if it detects an Explorer browser.

The system would benefit from adding more tools for geospatial processing. The sample user scenario in section 4.10 shows that the system is already capable of some spatial processing which provides tools suitable for disaster management activities. In many scenarios, tools for printing, styling by attribute, checking for points outside a polygon, intersection etc., would serve well.

Flooding analysis as implemented in this first version of WGRAS is quite naïve and should be improved to produce correct results. A more correct model for flood analysis is therefor needed. However, within the scope of this thesis there was not enough time to achieve this. The tool does however show the capabilities and techniques made available for further development.

5.6 External Data Integration

The ability to integrate external data sets into WGRAS is based on OGC compliance of the external data provider. The strength of using a standardized communication protocol in WGRAS was that the module for parsing and extracting data from the capabilities document could make use of the standard definition of each service version. The standardized structure of the capabilities document only required two different XML parsing modules to be developed, one for version 1.1.1 and one for version 1.3.0.

The tool was meant to be easy to use even for a user not familiar with the OGC standard. However, the user must know where to find the *GetCapabilities* URL of the external service for giving input to the connection process. The functionality of this tool could extended to include support for other services such as WFS and WCS and take the form of a Geoportal (Toomanian, A. 2013).

5.7 Future development

The future development of WGRAS should be based on the lessons learned from this work, the workshop in Yerevan and further input from users of the system. The improvements presented in sections 5.1 to 5.6 are all relevant and should be further investigated to prepare for a growing number of users and better data distribution in the system. This section also brings up the concept of Big Data as a way forward for WGRAS. A proposed architecture for scaling up WGRAS to meet these needs can be seen in figure 28.

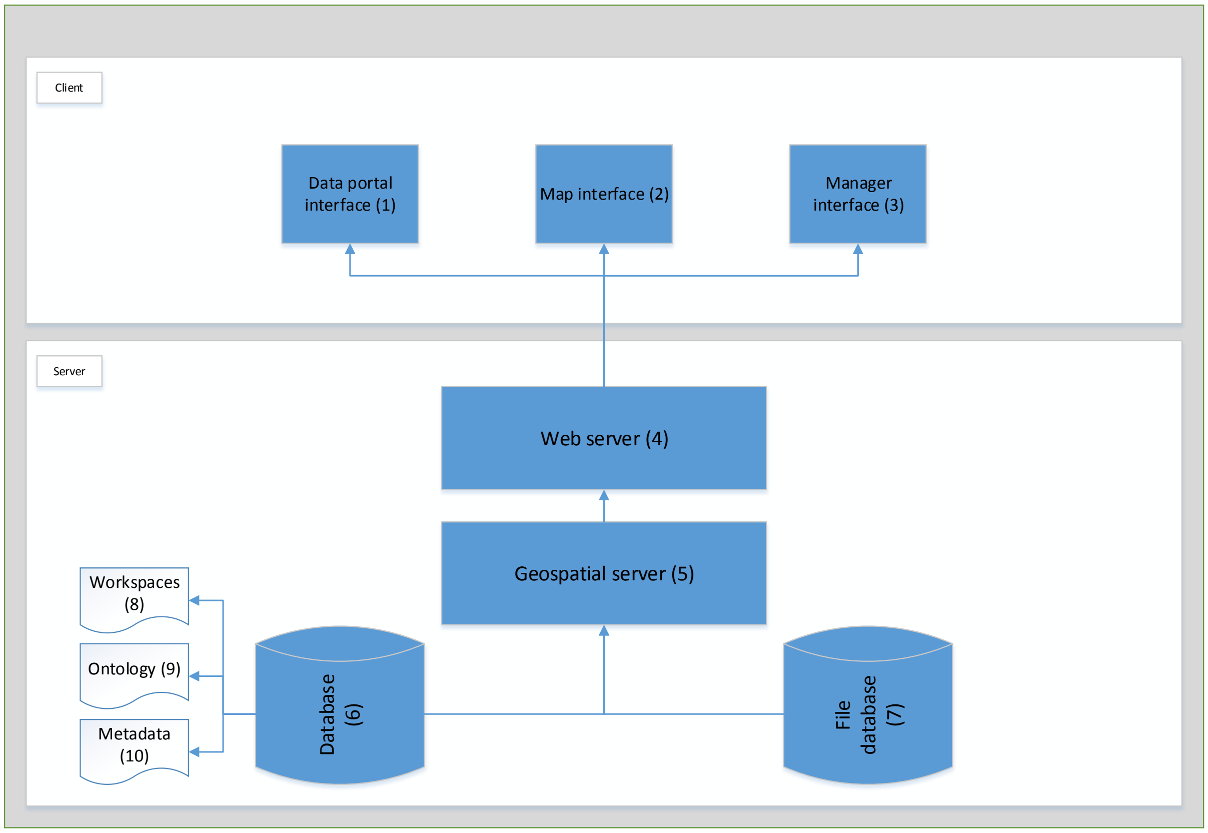


Figure 28 Proposed future architecture for WGRAS

1) **Data portal**: Interface for publishing/sharing data.

2) **Map:** Interface able to present map data from the geospatial service (5) and external service defined in the portal (1).

3) **Manager:** Interface for admin tasks such as adding/deleting users in the system.

4) **Web server**: Acts as an entry point to the network. Serves static files (JavaScript, HTML and CSS) to build up the client applications (1-3). Also handles authentication of users throughout the system.

5) **Geospatial server:** Serves map data from the database (6) and the file system (7). It also has geospatial processing capabilities. The role of the Geospatial server in this architecture is similar to the map-server presented in section 4.8.1.

6) **Database:** Geospatial database that stores GIS data and is able to perform geospatial processing.

7) **File database:** File system to store raster data in the case it cannot be stored directly in the Database (6).

8) **Workspaces:** A configuration specific to each user. Links datasets to a user and makes it possible for that user to store a map definition he/she created in the map (2).

9) **Ontology:** Details the data and the semantic relationships it might have with other entities in the Database (6).

10) **Metadata:** Metadata for the entity, such as projection, description, creation date etc.

The architectural overview seen in figure 28, adds additional components to the system presented in section 4.3, figure 11.

The Data Portal (1) is introduced as a module for import/export functionality of data to/from the system. In the first phase, connections to external OGC map services were implemented and the functionality resided within the map interface. The idea with including this module is to prepare the system for scalability by separating the data import/export functionality from the viewing functionality in the map interface. Linkage to the external data sources can for example be stored in the database and be used by the web server to put that service on to the map.

The Database (6) is also an additional component. In the architectural overview in figure 28, the database is depicted as a single component. In a real implementation, there exists several interconnected database servers to handle large datasets, large data processes and data backups for the system. As mentioned earlier, the main advantages with delegating spatial data processing tasks to a database are enhanced geospatial processing capabilities, security and speed compared to the existing implementation of a file based storage system. The database also offers the opportunity to store user defined data collections that can be created, shared, updated and deleted by users, see Workspaces (8) in figure 28.

In section 3.5, techniques for interoperability were presented as a way for a web GIS to create, maintain and include semantic linkage between different datasets, both external and internal. Ontologies can be implemented in a variety of ways and on different levels in a system and a more detailed explanation of how this would be implemented in WGRAS is not given here, as this has to be further investigated and tested. However, it can be concluded from the development- and testing process of WGRAS that it is highly motivated to find a suiting solution. The dynamic nature of disaster management activities demands quick and reliable combination and merging of new and changing datasets into one system. Such datasets need to be easily discovered by users meaning that semantic and meaningful links between data sets or entities in these data sets are needed. Otherwise, each data set risks becoming isolated entities. The ontology component in the proposed architecture can be seen in figure 28, Ontology (9), where it is depicted as a part of the database because of its tight relation to the data residing in this component.

The Metadata (10) in figure 28 is partly a new component. The existing system has a metadata service infrastructure exposed by a REST API; see section 4.8.2 but the reason for bringing it up as a new feature is that it could be extended with for example more linkages in the representation. A proposed enhancement is that each entity represented as in section 4.8.2, figure 13 could contain URIs for linked metadata entities. As an example, a metadata representation for lakes in Armenia could contain a collection of URIs pointing to metadata for lakes in countries bordering Armenia.

For WGRAS it is also important to prepare for future integration of big data sets. The concept of *Big Data* in web GIS has not been discussed in this thesis as it was not required for this first version. However, as the system expands and users are allowed to create, update and delete (CRUD) their own data sets or import external data sets for integration into their workflow, the available techniques must be considered. *Big Data* does not have a single clear definition but key features are high volume, high velocity (near real time creation), fine-grained (high resolution), flexible and relational. Popular examples of Big Data systems are Facebook, NASA and online Flight-booking systems (Kitchin 2013).

Spatial data is voluminous and fined grained by nature. A web GIS offering CRUD operations on spatial data can therefor benefit from using Big Data techniques for allowing data mining, high volume transactions and analytical processing. In WGRAS, such functionality would affect the system architecture on all levels, both server and client side. For example, additional storage capacities for handling a broader variety of data could be achieved by integrating NoSQL databases alongside SQL based systems. Data mining in Big Data systems requires a process for splitting up larger data sets into smaller chunks that can be handled by a multiple parallel processes (Xindong et al. 2014). Open Source software such as Apache Hadoop MapReduce could be used to create a service layer for handle such tasks (Apache Hadoop 2016).

6 Conclusions

The aim of this work was to develop an application for disaster and emergency management, based on a set of requirements and a proposed system architecture, with focus on usability, accessibility and interoperability.

Usability has been achieved by using modern techniques for web mapping, on both server and client side. The use of open source software has forced the system to be open and extendible, as it required integration of software and frameworks from many different sources. The use of open source software has also made the system more compatible with different development environments but has partially slowed down the development process due to poor documentation.

Usability has also been achieved by providing tools for geospatial processing by combining the strength of WPS for server side processing and client side manipulation of raster data using modern browser techniques. This combination introduces a new approach to geospatial processing in web GIS since it offers instant in-browser feedback and visualization of server side geospatial processes.

Accessibility has been achieved by using standardized OGC web service interfaces and standardized communication protocols. Accessibility has also been improved by providing a REST based service layer that simplifies access of metadata describing the available data sets in the system and can be used as a compliment to OWS for data consumers less familiar with OGC standards.

OGC web services has also proved to improve the interoperability of the system as it provides a standardized interface for communicating with- and extracting data sets from external services. Interoperability can be further improved by utilising techniques from ontology based architecture and context driven architecture.

The combination of OGC web services for data distribution, client- and server side geospatial processing and modern web mapping technology, has thus been a successful approach for the WGRAS system and provides a good foundation for further development.

Future development should firstly focus on the integration of a database management system for geospatial data as well as metadata and linkage between data sets. Ontology driven architecture is mentioned as a way forward for integrating such functionality as well as a further extending the existing REST API to include URIs for linking to related metadata entities. Techniques for Big Data integration into the system are also discussed and should be considered for further development.

7 References

Online Resources

Apache Hadoop 2016. What is Apache Hadoop? Retrieved 2016-01-25 from.
http://hadoop.apache.org/

ExtJs 2016, Sencha ExtJs API version 6. Retrieved 2016-01-13, from.
https://www.sencha.com/products/extjs/#overview

ESRI. 1988. ESRI Shapefile Technical Description, An ESRI White Paper. Environmental Systems Research Institute, Inc. Retrieved 2016-01-20, from.
https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

FIRMS 2016, FIRMS Web Fire Mapper. Retrieved 2016-01-12 from.
https://firms.modaps.eosdis.nasa.gov/firemap/

GeoJSON 2016, GeoJSON. Retrieved 2016-01-17, from.
http://geojson.org/

GeoServer. 2016. GeoServer User Manual. Retrieved 2016-01-15, from.
http://docs.geoserver.org/latest/en/user/

GNU 2016, A Quick Guide to GPLv3. Retrieved 2016-01-15, from.
http://www.gnu.org/licenses/quick-guide-gplv3.html

Google 2016, Google Maps API. Retrieved 2016-01-12, from.
https://developers.google.com/maps/

IETF 2016, The JavaScript Object Notation (JSON) Data Interchange Format. Retrieved 2016-01-06, from
https://tools.ietf.org/html/rfc7159

IFRC 2015 a, International Federation of Red Cross and Red Crescent Societies, what is a disaster. Retrieved 2016-01-11 from.
http://www.ifrc.org/en/what-we-do/disaster-management/about-disasters/what-is-a-disaster/

IFRC 2016 b, About disaster management. Retrieved 2016-01-18, from.
http://www.ifrc.org/en/what-we-do/disaster-management/about-disaster-management/

IFRC 2016 c, Responding to disasters. Retrieved 2016-01-10, from.
http://www.ifrc.org/en/what-we-do/disaster-management/responding/

MIT 2016, The MIT License (MIT). Retrieved 2016-01-14 from.
https://opensource.org/licenses/MIT

Nominatim 2016, Nominatim wiki. Retrieved 2016-12-20, from.
http://wiki.openstreetmap.org/wiki/Nominatim

OGC 2016a, OGC Members. Retrieved 2016-01-10, from.
http://www.opengeospatial.org/ogc/members

OGC 2016b, Open GIS Consortium Inc., Web Map Service Implementation Specification, version 1.1.1. Retrieved 2016-01-14, from.
http://www.opengeospatial.org/standards/wms

OGC 2016c, Open GIS Consortium Inc., Web Feature Service 2.0 Interface Standard. Retrieved 2016-01-14, from.
http://www.opengeospatial.org/standards/wfs

OGC 2016e, OGC WCS 2.0 Interface Standard - Core, version 2.0.1. Retrieved 2016-01-14, from.
http://www.opengeospatial.org/standards/wcs

OGC 2016e, Open GIS Consortium Inc., OGC WPS 2.0 Interface Standard. Retrieved 2016-01-14, from.
http://docs.opengeospatial.org/is/14-065/14-065.html

OGC 2016f, OpenGIS Geography Markup Language (GML) Encoding Standard. Retrieved 2016-01-15, from.
http://www.opengeospatial.org/standards/gml

OpenLayers 2016, OpenLayers version 3, Web Mapping API. Retrieved 2016-01-19 from.
http://openlayers.org/

OSGeo 2016, The Open Source GeoSpatial Foundation. Retrieved 2016-01-16, from.
http://www.osgeo.org/

SCALGO 2016, SCALGO LIVE GLOBAL Map. Retrieved 2015-12-20, from.
http://scalgo.com/live/

Turf 2016 Advanced Geospatial Analysis for Browsers and Node. Retrieved 2016-01-15, from.
http://turfjs.org/

Turf 2016b Turf API – inside function. Retrieved 2016-01-15, from.
http://turfjs.org/static/docs/module-turf_inside.html

UNISDR 2016 The Sendai Framework. Retrieved 2016-01-10, from.
http://www.un-spider.org/risks-and-disasters/sendai-framework-drr

UN-SPIDER 2016 a, Disaster Risk Management. Retrieved 2016-01-02, from.
http://www.un-spider.org/risks-and-disasters/disaster-risk-management

UN-SPIDER 2016 b, Information Management for Disaster Management. Retrieved 2016-01-18, from.
http://www.un-spider.org/risks-and-disasters/emergency-and-disaster-management/information-management

WHO 2016 a, Emergency and disaster risk management for health. Retrieved 2016-01-11, from.
http://www.who.int/hac/techguidance/preparedness/partnerships/en/

WHO 2016 b, Emergency Risk Management for Health Fact Sheets. Retrieved 2016-01-18, from.
http://www.who.int/hac/techguidance/preparedness/risk_management_overview_17may2013.pdf?ua=1

WHO 2016c The WHO e-atlas for Disaster Risk for the European Region. Retrieved 2016-01-02, from. http://data.euro.who.int/e-atlas/europe/

W3 2016a, The Original HTTP as defined in 1991. Retrieved 2016-01-18, from. http://www.w3.org/Protocols/HTTP/AsImplemented.html

W3 2016b, Hypertext Transfer Protocol –HTTP/1.1. Retrieved 2016-01-05, from. http://www.w3.org/Protocols/rfc2616/rfc2616.html

W3 2016c, XML Technology. Retrieved 2016-01-10, from. http://www.w3.org/standards/xml/

W3 2016d, Semantic Web Activity. Retrieved 2016-01-17, from. http://www.w3.org/2001/sw/

W3 2016e, HTTP Status Codes. Retrieved 2016-01-17, from. https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

W3C 2016 Cross-Origin Resource Sharing. Retrieved 2016-01-10, from. https://www.w3.org/TR/cors/

W3Schools 2016 W3 Schools. Retrieved 2016-01-10, from. http://www.w3schools.com/html/html5_webstorage.asp

Journal Articles

Albano et al 2015. READY: a web-based geographical information system for enhanced flood resilience through raising awareness in citizens. Natural Hazards and Earth System Sciences, 15: 1645-1658. DOI: 10.5194/nhess-15-1645-2015

Ahmed 2013. Proposed Model of GIS- Based Cloud Computing Architecture for Emergency System. International Journal of Engineering and Advanced Technology, 3.

Aye et al 2015. Prototype of a Web-based Participative Decision Support Platform in Natural Hazards and Risk Management. ISPRS International Journal of Geo-Information, 4: 1201-1224. DOI: 10.3390/ijgi4031201

Balbo et al. 2013. A PUBLIC PLATFORM FOR GEOSPATIAL DATA SHARING FOR DISASTER RISK MANAGEMENT. Role of Geomatics in Hydrogeological Risk, 40-5-W3: 189-195. DOI: 10.5194/isprsarchives-XL-5-W3-189-2013

Bogdanovic et al. 2015a. Methodology for geospatial data source discovery in ontology-driven geo-information integration architectures. Journal of Web Semantics, 32: 1-15. DOI: 10.1016/j.websem.2015.01.002

Bogdanovic et al. 2015b. An Approach for the Development of Context-Driven Web Map Solutions Based on Interoperable GIS platform. Computer Science and Information Systems, 12: 1055-1078. DOI: 10.2298/csis141031010b

Brinkhoff 2007. Increasing the Fitness of OGC-Compliant Web Map Services for the Web 2.0. European Information Society: Leading the Way with Geo-Information: 247-264. DOI: 10.1007/978-3-540-72385-1_15

Chung. 2015. Ontology-driven slope modelling for disaster management service. Cluster Computing-the Journal of Networks Software Tools and Applications, 18: 677-692. DOI: 10.1007/s10586-015-0424-1

Evans, B., and C. E. Sabel. 2012. Open-Source web-based geographical information system for health exposure assessment. International Journal of Health Geographics, 11: 11. DOI: 10.1186/1476-072x-11-2

Han and Feng. 2015. GIS Application based on Cloud Storage for Atmospheric Environmental Monitoring. Proceedings of the 3rd International Conference on Material, Mechanical and Manufacturing Engineering, 27: 972-976.

Hung et al. 2014. Optimal collaboration of thin-thick clients and resource allocation in cloud computing. Personal and Ubiquitous Computing, 18: 563-572. DOI: 10.1007/s00779-013-0673-z

Karnatak et al. 2012. Spatial mashup technology and real time data integration in geo-web application using open source GIS - a case study for disaster management. Geocarto International, 27: 499-514. DOI: 10.1080/10106049.2011.650651

Kawasaki and Sadohara 2005. Investigation of the utilization of GIS in emergency response to the 2001 WTC building collapse disaster no.2: through the interviews in New York City. Proceedings of the Annual Conference of the Institute of Social Safety Science. 16. pp. 1–4.

Kawasaki et al. 2013. The growing role of web-based geospatial technology in disaster response and support. Disasters, 37: 201-221. DOI: 10.1111/j.1467-7717.2012.01302.x

Kitchin 2013. Big Data and Human Geography: Opportunities, Challenges and Risks. Dialogues in Human Geography 3:262-267. DOI: 10.11777

Mansourian, A., A. Rajabifard, M. J. V. Zoej, and I. Williamson. 2006. Using SDI and web-based system to facilitate disaster management. Computers & Geosciences, 32: 303-315. DOI: 10.1016/j.cageo.2005.06.017

Maso et al. 2014. Building the World Wide Hypermap (WWH) with a RESTful architecture. International Journal of Digital Earth, 7: 175-193. DOI: 10.1080/17538947.2012.669414

Mazzetti et al. 2008. Integration of REST style and AJAX technologies to build Web applications. 2008 3rd International Conference on Information and Communication Technologies: from Theory to Applications, Vols 1-5: 2237-2242.

Mesbah and van Deursen 2007. Migrating multi-page web applications to single-page AJAX interfaces. Csmr 2007: 11th European Conference on Software Maintenance and Reengineering, Proceedings: Sofware Evolution in Complex Software Intensive Systems: 181-190.

Muller et al. 2006. CEDIM Risk Explorer - a map server solution in the project "Risk Map Germany". Natural Hazards and Earth System Sciences, 6: 711-720.

Nosal 2008. Flood-fill algorithms used for passive acoustic detection and tracking. Passive '08: 2008 New Trends for Environmental Monitoring Using Passive Systems: 13-17.

Pliutau and Prasad. 2013. Usage of data-encoded web maps with client side color rendering for combined data access, visualization and modeling purposes. In Conference on Geospatial InfoFusion III. Baltimore, MD: Spie-Int Soc Optical Engineering.

Tamayo et al. 2012. Measuring complexity in OGC web services XML schemas: pragmatic use and solutions. International Journal of Geographical Information Science, 26: 1109-1130. DOI: 10.1080/13658816.2011.626602

Toomanian, A. 2013. Automatic integration of spatial data in viewing services using a semantic based expert system. Journal of Spatial Information Science: 43–58. DOI: doi:10.5311/JOSIS.2013.6.87

Vaccari et al. 2012. An evaluation of ontology matching in geo-service applications. Geoinformatica, 16: 31-66. DOI: 10.1007/s10707-011-0125-8

Xindong et al. 2014. Data Mining With Big Data. IEEE Transactions on knowledge and data engineering 26:97-107

Books

Ashmore, D. C. 2014. The JAVA EE Architect's Handbook. DVT Press Bolingbrook, IL.

Ding et al. 2007. A Handbook of Principles, Concepts and Applications in Information Systems. Springer Science. pp 79-113
Elmasri. 1994. Fundamentals of Database Systems. Bridge Parkway, Redwood City: The Benjamin/Cummings Publishing Company Inc. pp 2-33

Fielding 2000. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine

Harrie and Svensson 2012. Lagring av geografiska data. I Geografisk informationsbehandling –teori, metoder och tillämpningar, L. Harrie, pp 139-160. Studentlitteratur: Lund.

Li, S Zlatanova, A Fabbri. 2007, Geomatics Solutions for Disaster Management, Springer-Verlag Berlin Heidelberg

Martin 2003. Agile Software Development, Principles, Patterns and Practices, Pearson Education Inc

Michaels and Ames 2008. Web Feature Service and Web Map Service. Encyclopedia of GIS. Shekhar & H. Xiong, pp 1259-1271. New York: Springer Science

Peggion, M., Bernardini, A., and Masera, M. 2008. Geographic Information Systems and Risk Assessment, Scientific and Technical Research series EUR, Office for Official Publications of the European Communities, Luxembourg