Hochschule Karlsruhe – Technik und Wirtschaft

University of Applied Sciences

Fakultät für Fakultät für Informationsmanagement und Medien

Master´s Programme Geomatics

Master´s Thesis

of

Alberto Miguel Rodrigo Martínez

Migration of ESRI Server data to an Apache Server with Geoserver and Openlayers. *Settlement development in Vel'ké Pole, Pila and Radobica, Slovakia*.

First Referee:	Prof. Dr.-Ing. H. Saler
Second Referee:	Prof. Dr. Ing. José Carlos Martínez Llario
Supervisor:	Dipl.-Ing. (FH) Christian Stern

Karlsruhe, September 2017

Abstract.

Nowadays there is a trend in adding a map viewer to commercial websites. It is the best way of locating their own shops, or the place where you can find some product, and create a GPS track from your location. There are also more specific websites involved in showing cartographical objects (water, electrical, gas pipes, river, roads…) to their users. They can interact with these objects checking their length, if they are lines (pipes, roads, rivers) or areas (lakes, land parcels, buildings, pools…) and when they don’t need to look for the information in the databases, the website is programmed to prevent the users from doing this task.

These map viewers are easy to create with commercial software. Geomatics engineers are taught the knowledge to create this, but with free software. Therefore, it is our duty to teach companies how to avoid these unnecessary payments and save money.

The most used software in GIS world is ArcGIS, the GIS from ESRI company. This software has an expensive license and they require an annual payment. It is easy to install and has a friendly interface for creating and managing spatial objects like polylines, polygon or raster files. There are a lot of companies paying for it and sometimes it is not necessary to have this program. There are other free software programs (open source software) which can do the same functions as those provided by ESRI.

That will be the basis of this project. One existing website, which is being developed by the students from Geomatics department of Hochschule Karlsruhe, shows cadastral information from four Slovakian villages. These data are composed of old cadastral, altitude, solar radiation, slope maps, agricultural, forest or settlement development, river and roads during more than 200 years.

The website shows all this data with ESRI web map viewer and an open source software alternative will be found and programmed to be used instead of it. There is a lot of software to test, but the chosen one has to be able to run the same functions like ESRI website has

The tests that will be performed will be about speed and features comparisons. Maybe the fastest alternative would be not the best choice if it doesn’t have all the functions used on the website with ESRI software.

As concluding part, another test will be made for comparing the existing website version with ESRI software with the new version. It will be analysed beyond being a free alternative, also if the features and functions are working better or not.

Table of Content

List of Figures

1. Introduction

1.1. Working goals

The Geomatics department from the Information Management & Media faculty of the Hochschule Karlsruhe - Technik und Wirtschaft is developing a website about the Vel'ké Pole, Píla and Radobica villages, in Slovakia. This website is about the development of these villages regarding roads, settlement, agriculture, forest and rivers from 1782 to today, showing their land use, land use changes, and settlement development.

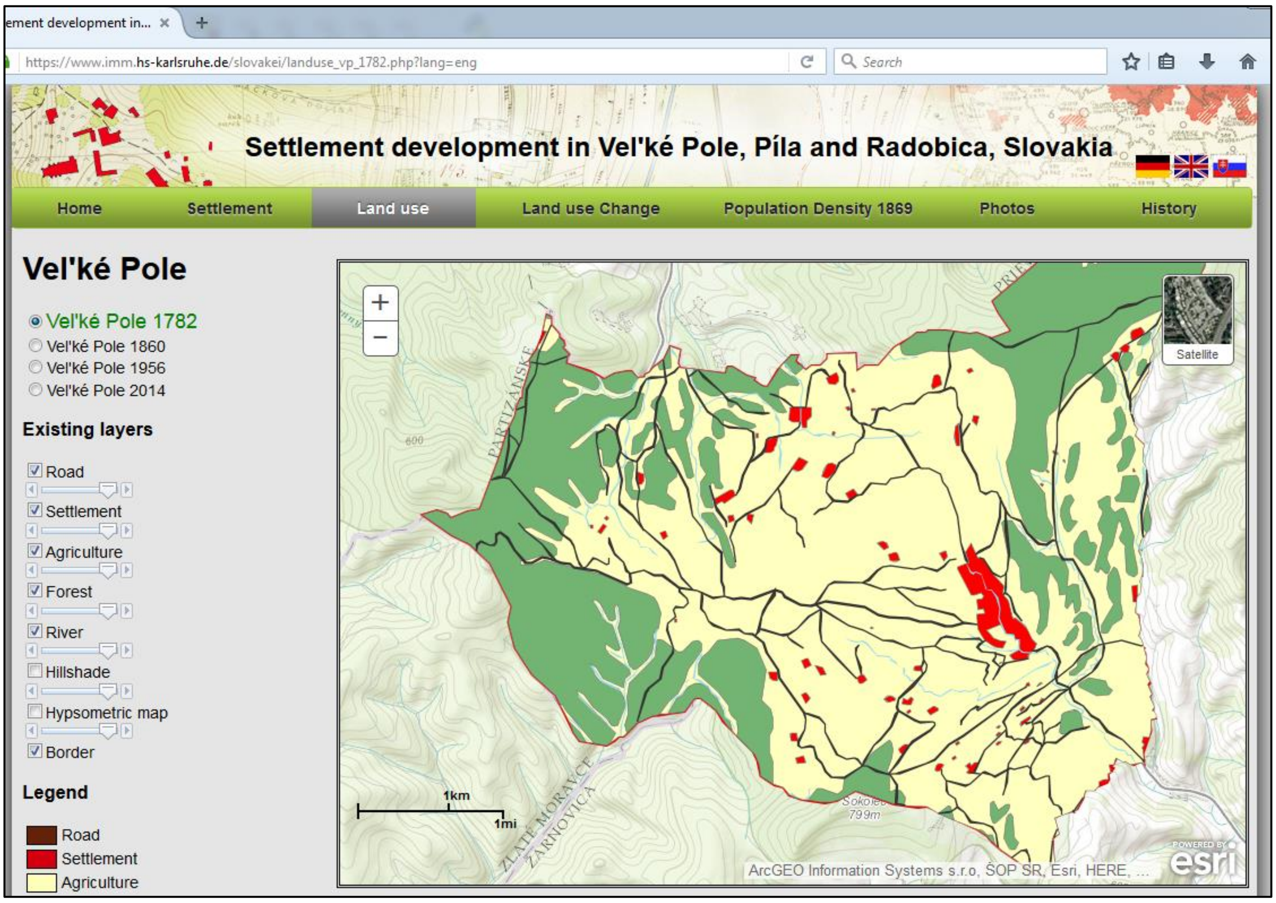


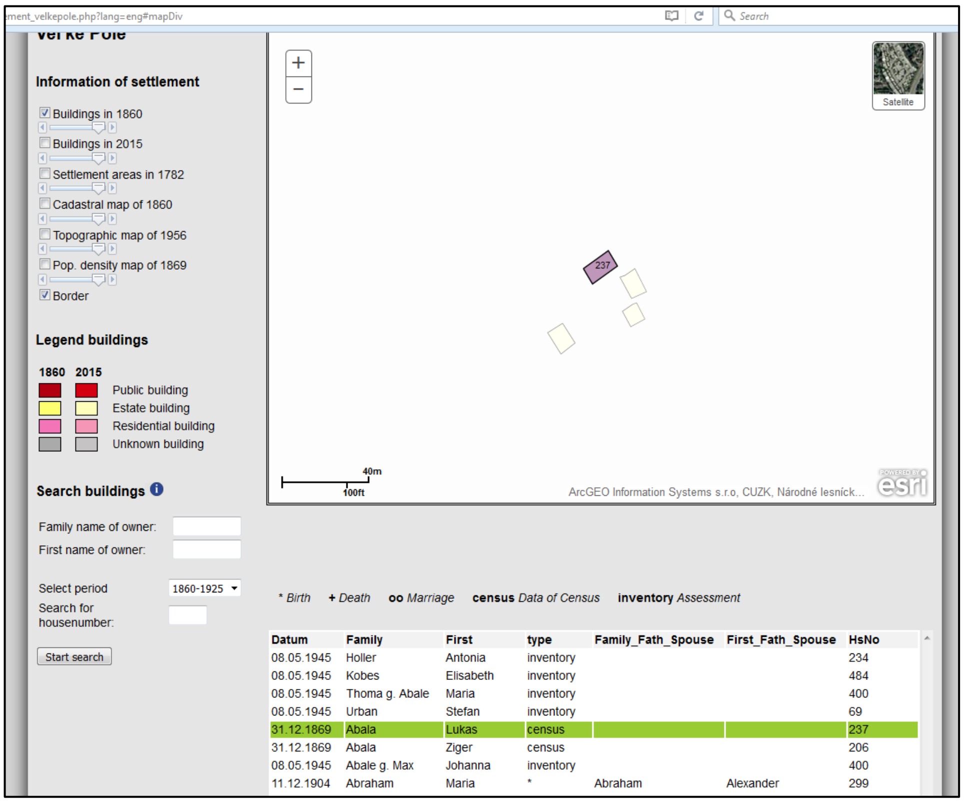
Figure 1.1: First website view

As a first view, there is not so much data involved in this project. The data is loaded from an ESRI server and this has to be migrated into an **open source server.** The ESRI server is a commercial server, so it requires an annual payment.

Shapefiles (points, polylines and polygons) and raster files (hill shades, density population maps, old cadastral maps) are loaded into this website by a ESRI web viewer from the ESRI server and, a research will be made to choose the best alternative open source server (MapServer, Geoserver, Mapnik…) and open source viewer (Openlayers or Leaflet javascript libraries) for this project.

This research is not based only on performance, it is not about looking for the fastest alternative. Maybe the fastest option doesn't have the same functionalities as the original website. Those functionalities include displaying a layer control with opacity switcher, popup windows showing information about the object which is being clicked on the map… Besides, the original website has a query table which is loaded when a user looks for buildings by the owner's name or by the building's year. This data is loaded from a MySql database.

In figure 2, the original website's search buildings function is shown here. The table is showing the data for the period between 1860 – 1925 where, the building with the house number 237 is being shown after clicking on the same building of that table.



* *Birth* **+** *Death* **oo** *Marriage* **census** *Data of Census* **inventory** *Assessment*

Datum	Family	First	type	Family_Fath_Spouse	First_Fath_Spouse	HsNo
08.05.1945	Holler	Antonia	inventory			234
08.05.1945	Kobes	Elisabeth	inventory			484
08.05.1945	Thoma g. Abale	Maria	inventory			400
08.05.1945	Urban	Stefan	inventory			69
31.12.1869	Abala	Lukas	census			237
31.12.1869	Abala	Ziger	census			206
08.05.1945	Abale g. Max	Johanna	inventory			400
11.12.1904	Abraham	Maria	*	Abraham	Alexander	299

Figure 1.2: Original website search buildings function

Thus, the chosen alternative must be able to do these functions.

1.2. Study area

The website depicts the villages of Vel'ké Pole, Píla, Brezova and Radobica, from Slovakia. This area is more or less 170 kilometres from Bratislava, on north-west direction. This area has a mild weather.

The theme of the website is about the settlement development by the German migration to Slovakia since the 12th century. This area was uninhabited at that moment, especially after the Mongol invasion of Europe. These Germans are collectively called Karpatendeutsche, or Carpathian Germans, because they lived in the northern part of the great arc formed by the Carpathian Mountains. (Genealogy: Slovakia., 2000)

Here it is shown the study area with a red polygon. Figure 1.3, and figure 1.4, show the borders of the four villages.

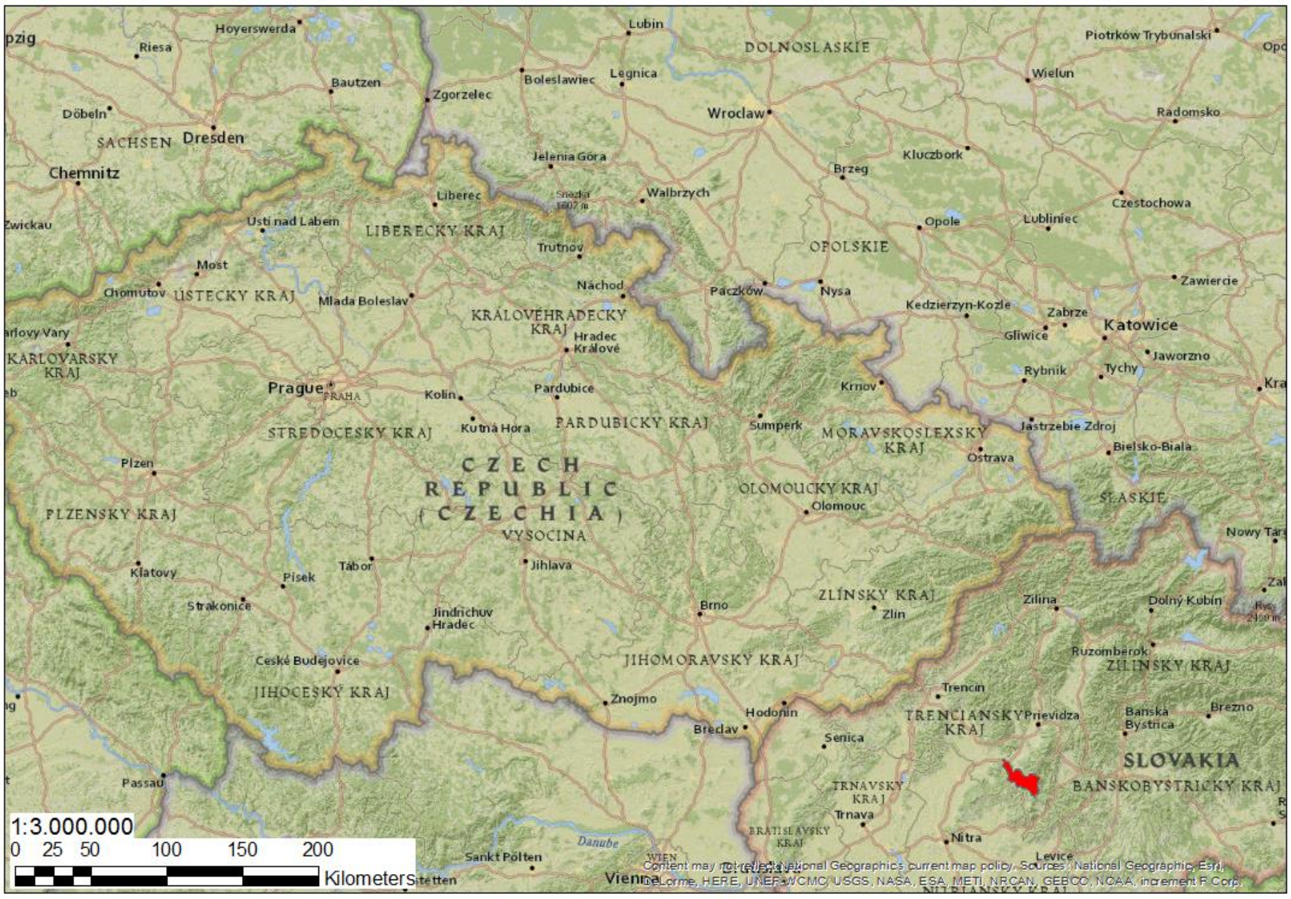


Figure 1.3: Study area highlighted in red. Scale: 1:3.000.000

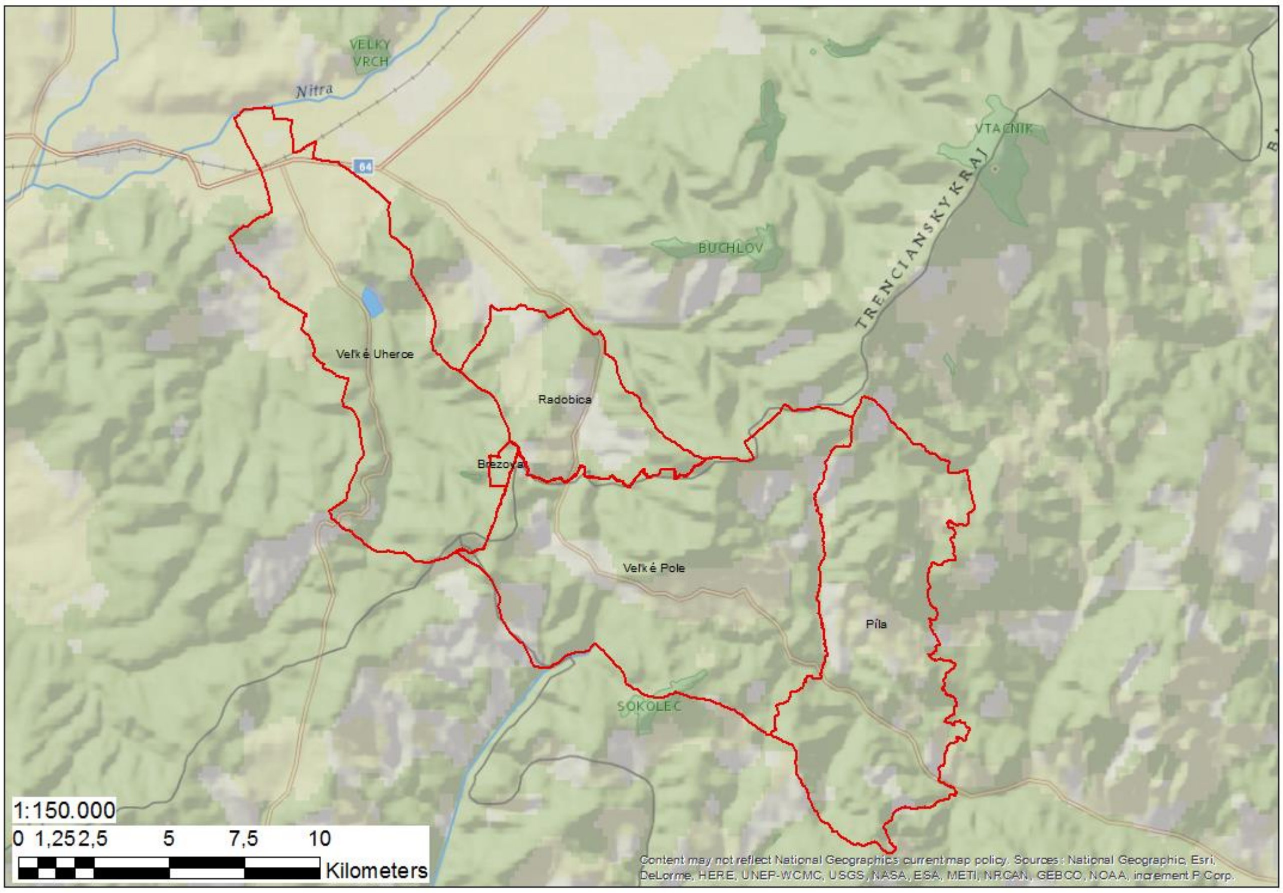

1:150.000
0 1,25 2,5 5 7,5 10
Kilometers
Nitra
VELKY VRCH
BUCHLOV
VTACNIK
TRENCIANSKY KRAJ
SOKOLEC
Veľké Uherce
Radobica
Brezova
Veľké Pole
Píla
Content may not reflect National Geographic's current map policy. Sources: National Geographic, Esri, DeLorme, HERE, UNEP-WCMC, USGS, NASA, ESA, METI, NRCAN, GEBCO, NOAA, increment P Corp.

Figure 1.4: Study area. Small scale 1:150.000

1.2.1. Vel'ké Pole

The first documents from the Vel’ké Pole settlement date from 1335. Specialized German workers were needed by the nobles and kings. That made the settlement grow and increased different industries, like commerce, trade and mining around this area. There was a decrease in population during the 19th century because of mass emigration of the working class. Anyway, the population increased again until the beginning of the 20th century. There was another big decrease in population after the World War II due to the deportation of Germans. This made the settlement lose around 2200 people, from 2900 to 700 people. After 1950 the population levels stabilized again, but there were less people than in previous years. (Settlement development in Vel’ké Pole, Pila ad Radobica, Slovakia, 2016).

1.2.2. Píla

The first Píla documents date from 1429. Píla has changed its name several times until 1920, when the current name Píla was used by the first time.

The first Germans arrived to Píla during the 15th century. They arrived mostly from Bavaria and Swabia and at that moment the languages in the area were the dialects from these places. They had no changes until the World War II, during which the same decrease in population issue as in Vel’ké Pole happened. The most important industries in Píla were mining, forestry and agriculture. The population increase to this area was steady until the 20th century, when that decrease in population happened and also because of the lack of work places for its inhabitants.

In 1944, 92% of the inhabitants of the region were German, with the remaining 8% being Slovak. After World War II, as in Vel’ké Pole, many German were deported but less people than Vel´ké Pole, around 838 of the 1463 existing residents.

From this time, the population was decreasing until nowadays, because the young families move away looking for a job. (Settlement development in Vel’ké Pole, Pila ad Radobica, Slovakia, 2016).

1.2.3. Brezova

Brezova is in the south-eastern part of the municipality Vel'ké Uherce and thus in the Western middle Slovakia. The settlement is situated at an altitude of 550 m.

Brezova has no inhabitants before 1780. The municipality Vel'ké Uherce which includes the village Brezova was first mentioned in 1274 as 'Vgrych'. 1941 Brezova (in German 'Birkenhain') was briefly incorporated into the municipality of Vel'ké Pole. However, after a few years Brezova was back to the municipality Vel'ké Uherce.

Like Vel’ké Pole and Pila, after World War II many Germans were deported from the region. Nowadays, the 99% of the inhabitants are Slovaks, while the rest are Czechs and Poles. Also like Píla village, Brezova also had a steady decreasing population until the present day, where only the older generations remain and the young generations moved out looking for work. This village was inhabited some last decades. (Settlement development in Vel’ké Pole, Pila ad Radobica, Slovakia, 2016).

1.2.4. Radobica

The first documents found of Radobica are from 1324 and belong to the district of Prievidza. Nowadays, around 530 people are living in Radobica. This place is a famous recreational area because of its landscapes, nature and calm and, because of this, the mainly industry here is the touristic sector. However, this land needed craftsmen and farmers, who settled in Radobica. Around 1950, Radobica had the highest population, at around 1000 people, but the people preferred working at the neighbouring villages, also because they are not so far away from Radobica. Besides, the people moved out to the Czech Republic or to Austria.

Unlike Vel'ké Pole and Pìla, Radobica has never been a German municipality. However, a minority of German settlers were inhabitants of Radobica. At present, 91.51% are of Slovak nationality and 8.49% belong to other nationalities. (Settlement development in Vel'ké Pole, Pila ad Radobica, Slovakia, 2016).

1.3. Schedule for the project

Once the background of the website has been explained and how far the project will be, the schedule for the whole thesis can be stablished. The following table was being updated during the project. All the meetings have the date in which they took place and the topic discussed on those meetings

Activity	Start Time	Length	Finish Time
1.Market Research of Open Source Servers.	04/04/2017	3	07/04/2017
2. Testing the servers.	08/04/2017	11	19/04/2017
3. First Report.	04/04/2017	28	02/05/2017
4. Choice of two servers for our purposes.	18/04/2017	1	19/04/2017
5. Check the data base reliability.	25/04/2017	7	02/05/2017
6. Building server with MapServer and Geoserver.	20/04/2017	17	07/05/2017
7. First meeting. Schedule the first project's steps	04/04/2017	1	05/04/2017
8. Second meeting. Getting website data files.	19/04/2017	1	20/04/2017
9. Third meeting. Looking for a server	02/05/2017	1	03/05/2017
10. Test the servers and decide which is better for the project.	07/05/2017	1	08/05/2017
11. Fourth meeting. Openlayers chosen as javascript library.	08/05/2017	1	09/05/2017
12. HTML based on ESRI server change into HTML open source server.	08/05/2017	82	29/07/2017
13. Fifth meeting. Javascript Issues	17/05/2017	1	18/05/2017
14. Sixth meeting. Javascript Issues.	31/05/2017	1	01/06/2017
15. Seventh meeting. Javascript Issues.	12/06/2017	1	13/06/2017
16. Eighth meeting. Javascript Issues.	26/06/2017	1	27/06/2017
17. Ninth meeting. Javascript Issues.	17/07/2017	1	18/07/2017
18. Tenth meeting. Javascript Issues.	18/07/2017	1	19/07/2017
19. Eleventh meeting. Javascript Issues.	19/07/2017	1	20/07/2017
20. Twelveth meeting. Showing the new website done.	29/07/2017	1	30/07/2017
20. Testing the final server on the University server.	01/08/2017	20	21/08/2017
21. Final Report.	13/04/2017	130	13/08/2017

Figure 1.5: Table with Project schedule

1.4. Introduction about open source web servers

Thanks to the recent improvement in household internet speed, dealing with geographical information data with internet browsers is trending nowadays. Those web map browsers are called web GIS websites. Unlike the usual desktop GIS software, web GIS websites are able to get updated data on real time from the developers. It makes easier for the unexperienced users who has never handled geographical information before. One of the main reasons is that the files have to be downloaded from official State institutions to get updated geographical data for desktop GIS. The web GIS developer is able to program the web GIS to make this process on background, and the users don't need to do this.

A GIS web needs a web server for being online. In this figure, it is depicted how a web server works with an example. The user makes HTTP requests on the background when he or she asks for data, for example, asking the server for one Terrain map on the website where is working on. Then the server sends back a HTML response with the desired map.

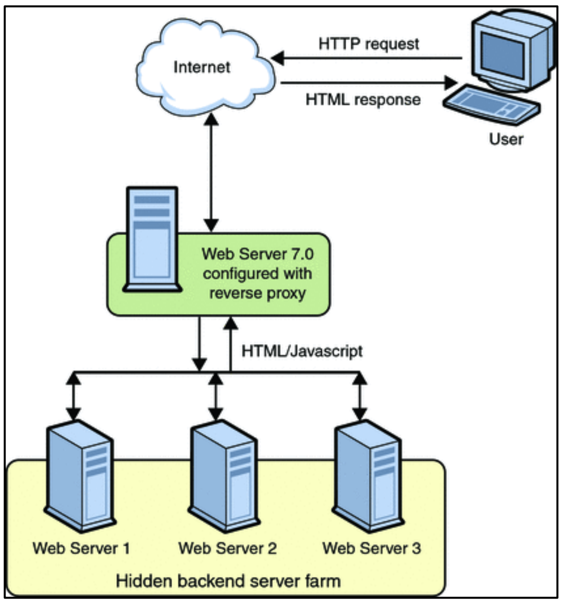


Figure 1.6: Server Architecture

This is how a GIS website is designed. Therefore, this website needs a server for being online 24 hours every day.

The easiest way for a company is to rent one private server, but this is money they lose because private servers could be expensive. For this reason, setting up an open source web server with the newest releases is the best and cheapest choice a small company (or university) could do.

This is the project basis that is mentioned in 1.1. section, migrating a web server from a private server into an open source web service.

1.5. Comparison between the most usual open source web services

First of all, a market research has been done with the different alternatives that are available on the web.

One web post has been found where the performance between the most used map servers is compared. (Maxim Rylov, Benchmarking Mapping Toolkits in Tile Seeding, 2015)

These map servers are: Geoserver, MapServer, Map Suit, Mapsurfer.NET and Mapnik. Subsequently QGIS server would be considered as a choice, although this is not part of this comparison.

The best server according to this post is Mapsurfer.NET, but this is not open source. Anyway, the information about the rest of the servers could be useful for taking a choice between them, according to the needs of the Slovakia website.

Mapnik and MapServer look like the best servers after Mapsurfer.NET. They lose its main advantage if our project works on large scales. In fact, Mapnik deals with re-projection using proj4 library, which makes Mapnik slower in comparison with the other map servers. Furthermore, Mapnik and MapServer are good at dealing with huge data because of its RAM management. In this part, Geoserver is the worst because it needs so much RAM for processing tiles but, as there are not too many layers and base maps in this website, it's not necessary to care about RAM processing.

Mapnik and MapServer are the best servers if the data contains many tiles, base maps, and database files. Both of them save RAM processes and, especially Mapnik, have a faster tile response in small scales where there is a lot of tiles to process. Mapnik works especially well on Linux, also on windows, but it's difficult to set up.

Then, MapServer and Geoserver could be a choice because there is not too much data files on the Slovakia website.

1.6. Open source software available and how to set up

1.6.1. Mapnik 3.0.12

Mapnik is a Free Toolkit for developing mapping applications. It is written in modern C++ and has Python bindings that support fast-paced agile development. It can comfortably be used for both desktop map design and web development.

Mapnik is about making beautiful maps. It uses the AGG graphics library, which offers world-class anti-aliasing rendering with subpixel accuracy for geographic data.

It can read ESRI shapefiles, PostGIS, TIFF raster. osm files, any GDAL or OGR supported formats, CSV files, and more. Pre-built packages are available for OS X and Windows and can be found at Mapnik.org/download. Many Linux distributions provide packages - learn about them at the Mapnik Installation Wiki. For this reason, it's better work on Linux for set up the server and its management. (Mapnik, 2016)

1.6.1.1. Install on Windows and test (Gerd Katzenbeisser, 2015)

1. First download the first of the two windows packages:
- Runtime package for use via Python: http://mapnik.s3.amazonaws.com/dist/v2.2.0/mapnik-win-v2.2.0.zip
- Full SDK with headers and .lib files for compiling C++ programs: http://mapnik.s3.amazonaws.com/dist/v2.2.0/mapnik-win-sdk-v2.2.0.zip

2. Extract the archive to c:\\mapnik-v2.2.0
3. Setup environment variables in your shell
- set PATH=c:\mapnik-v2.2.0\lib;%PATH% set PATH=c:\mapnik-v2.2.0\bin;%PATH%
4. Test the c++ demo
- cd c:\mapnik-v2.2.0\demo\c++ rundemo ....\
  It doesn't run on windows.
5. Test the python demo

First ensure you have 32 bit python 27 installed. Then do:

- set PYTHONPATH=c:\\mapnik-v2.2.0\python\2.7\site-packages;%PYTHONPATH%
- set PATH=c:\\Python27;%PATH%
- cd c:\\mapnik-v2.2.0\demo\python
- python rundemo.py

This python file makes several files inside the same folder as rundemo.py file.

This map image is generated from 5 shapefiles within demo folder.

Inside of rundemo.py file the code is written for building this map, so Mapnik doesn't have a GUI for managing the web maps like Geoserver does.

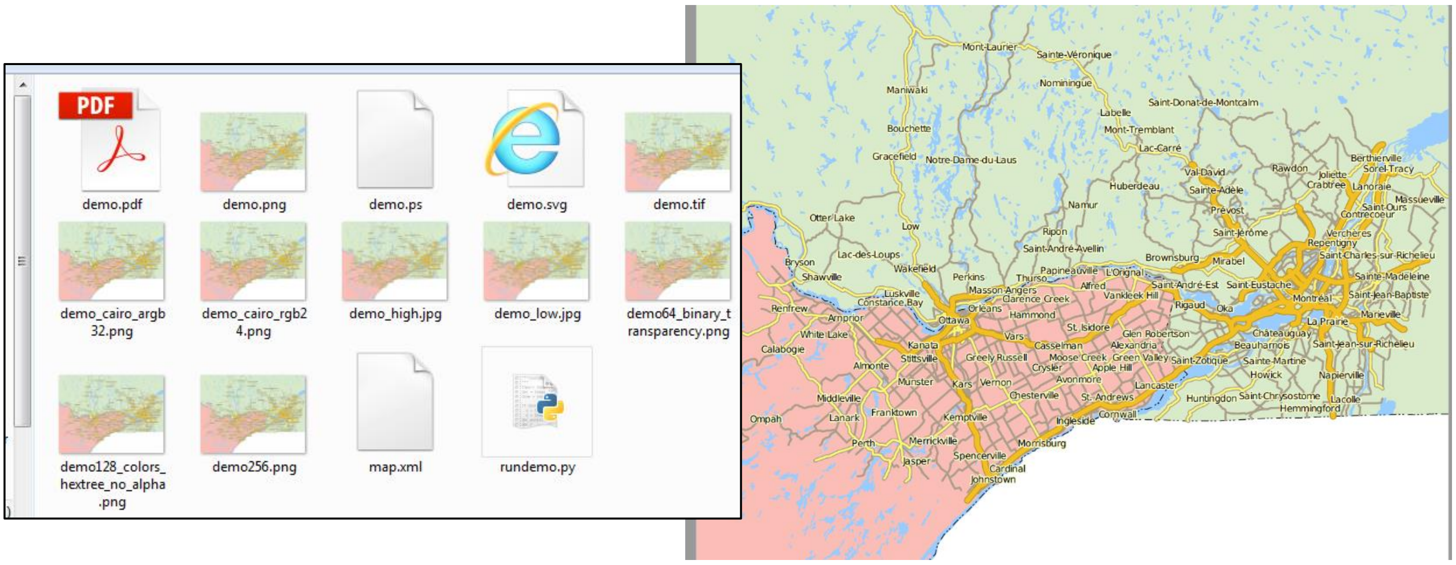


Figure 1.7: Map Web made with Mapnik

This step is about creating a map composition for publishing an online WMS afterwards. For doing this, a Mapnik server has to be set up.

1.6.1.2. Running the Mapnik server

The documentation for installing Mapnik server for Windows is limited and the installation could be annoying, because it is necessary to handle several python libraries for running it on Windows. Furthermore, the Mapnik libraries have to be in PATH on windows. The Mapnik wiki GitHub webpage explains only about Mapnik library itself and one demo for testing if it works. This demo is the example explained in the last point. But there is no explanation about how to run the server.

One more python library has been found: OGCServer. The developers explain that this is the Mapnik python library to publish maps through an open and standard WMS interface.

But there is a problem, the OGCServer wiki page is broken and there is no chance to find an easy explanation of how it works.

This is the GitHub website:

https://github.com/mapnik/OGCServer/tree/master/docs

There is also another google code archive group containing an example to run OGC WMS Server, but the test downloads links within this website is up to date or broken.

https://code.google.com/archive/p/mapnik-utils/wikis/WmsInstallGuide.wiki

Also, there are some explanations of how to set up the OGCServer files for working with Mapnik in this website. OGCServer works like Mapnik with Python scripts files and understanding it could be difficult if there are no available examples.

In this website, one can find the explanation of how to install this server. The dependencies are this:

- Mapnik python bindings (which will also install the `ogcserver` module code)
- PIL (http://www.pythonware.com/products/pil)

For the CGI/FastCGI interface also install:

- jonpy (http://jonpy.sourceforge.net/)

These files are Python scripts packages also so, if you need to change the settings for publishing a WMS, it is necessary to know exactly which lines of code are involved in this.

There are also posts explaining that Mapnik works with Apache server, but there isn't any information about this within OGCServer topic from the Mapnik GitHub website.

1.6.1.3. Advantages

There are many python libraries for working on Linux machine with his documentation. Mapnik is faster with request responses of tile images than Geoserver or MapServer, especially with big databases.

Tilemill is the only Mapnik program with graphical user interface. With this software, it's possible to edit all the objects merged with Mapnik. Besides, the use of this library is widespread so there are a lot of tools and developers which can help you through some of the online help groups.

1.6.1.4. Disadvantages

Mapnik works for OSX, Windows and Linux, but it is recommendable to make it work on Linux. For Windows, some required python libraries have to be downloaded, which is not necessary on Linux or OSX.

It is difficult to find any help for building the server with OGCServer. Also, it is difficult to understand all the python files which are involved in each OGC manager's tools.

There is not graphical user interface for setting up and running Mapnik server. Only the python files are involved in the connection with the server. For changing any setting, there are line codes inside some python files which have to be modified.

The documentation for running the server is limited to the Windows version. OGCServer is mentioned on GitHub MapServer posts as the server Mapnik works with, but the website about how it works with an example is not working anymore and it is not possible to understand how it works.

1.6.2. QGIS Server 2

QGIS Server is an open source WMS 1.3 and WFS 1.0.0 implementation which, in addition, implements advanced cartographic features for thematic mapping. The QGIS Server is a FastCGI/CGI (Common Gateway Interface) application written in C++ that works together with a webserver (e.g. Apache, Lighttpd). It is funded by the EU projects Orchestra, Sany and the city of Uster in Switzerland.

It uses QGIS as backend for the GIS logic and for map rendering. Furthermore, the Qt library is used for graphics and for platform independent C++ programming. In contrast to other WMS software, the QGIS Server uses cartographic rules as a configuration language, both for the server configuration and for the user-defined cartographic rules.

Moreover, the QGIS Server project provides the 'Publish to Web' plugin, a plugin for QGIS desktop which exports the current layers and symbiology as a web project for QGIS Server (containing cartographic visualization rules expressed in SLD).

As QGIS desktop and QGIS Server use the same visualization libraries, the maps that are published on the web look the same as in desktop GIS. The 'Publish to Web' plugin currently supports basic symbolization, with more complex cartographic visualization rules introduced manually. As the configuration is performed with the SLD standard and its documented extensions, there is only one standardised language to learn, which greatly simplifies the complexity of creating maps for the Web.

With OSGeo4W Network Installer, it's possible to install also the QGIS server web development package. Apache Server is also installed with this package. (Documentation about QGIS Server, n.d.)

How to run the QGIS server can be found here:

http://hub.qgis.org/projects/quantum-gis/wiki/QGIS_Server_Tutorial

The WMS online resource for QGIS Server is:

http://localhost/qgis/qgis_mapserv.fcgi.exe?

Professional Geomatics' Groups describe QGIS as a potential server solution with its new version QGIS Server 3 but, the performance of the features on QGIS Server 2 are far away from Mapnik's, Mapsserver's or Geoserver's performance for now.

1.6.2.1. Advantages

Create one server directly after managing one GIS file. Besides, it's possible to print with a plugin.

1.6.2.2. Disadvantages

Very limited documentation help and WMS tiling results in edge effects on labels and polygon fill patterns. No support for users or Groups also. Furthermore, there aren't any inbuilt caching options and its performance is slower than Geoserver. QGIS Server 3 release is expected for July 2017. This project was finished at the end of August and it was not released yet.

1.6.3. Geoserver 2.11 and Apache Tomcat server

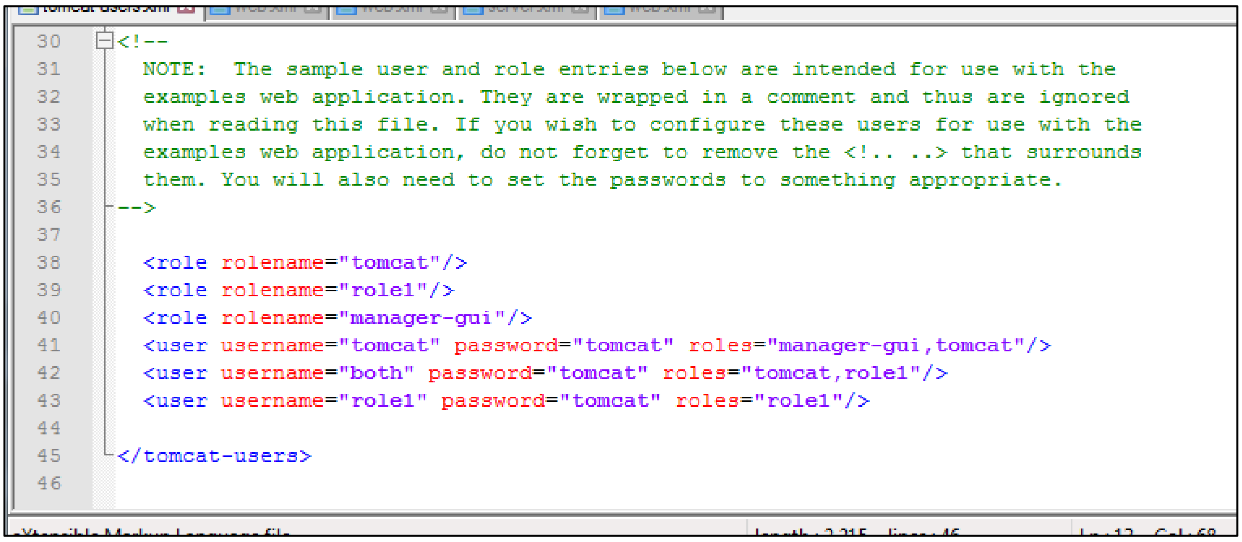
Geoserver is an open source server written in Java that allows users to share and edit geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards. Geoserver has evolved to become an easy method of connecting existing information to Virtual Globes such as Google Earth and NASA World Wind as well as web-based maps such as Google Maps and Windows Live Local. Geoserver is the reference implementation of the Open Geospatial Consortium Web Feature Service standard, and also implements the Web Map Service and Web Coverage Service specifications. (Geoserver, 2011).

Geoserver war file (Web archive for servlet containers) download can be found here: https://sourceforge.net/projects/geoserver/files/GeoServer/2.11.2/geoserver-2.11.2-war.zip/download

It's recommendable to install it as Apache Tomcat's war package. http://tomcat.apache.org/download-90.cgi

After install apache server and, before executing 'Configure Tomcat.exe', a couple of files must be modified:

Go to C:\Tomcat 9.0\conf and then open tomcat-users.xml, here you are able to modify the default user and the password.

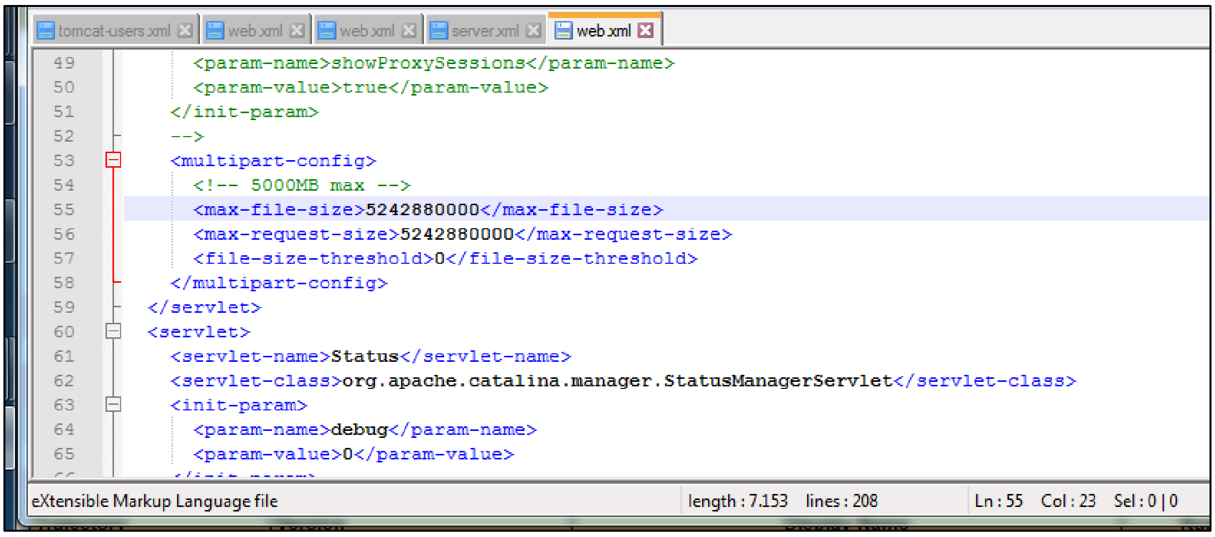


```
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->

  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename="manager-gui"/>
  <user username="tomcat" password="tomcat" roles="manager-gui,tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>

</tomcat-users>
```

Figure 1.8: Modifying user and password name from Tomcat server

By default, it’s only possible to install less than 50 MB war packages into Tomcat server, the war file is bigger so, another file must be modified. Go to C:\Tomcat 9.0\webapps\manager\WEB-INF directory and open web.xml file.

Modify max file size and max request size variables. Then, you are allowed to install geoserver.war file.

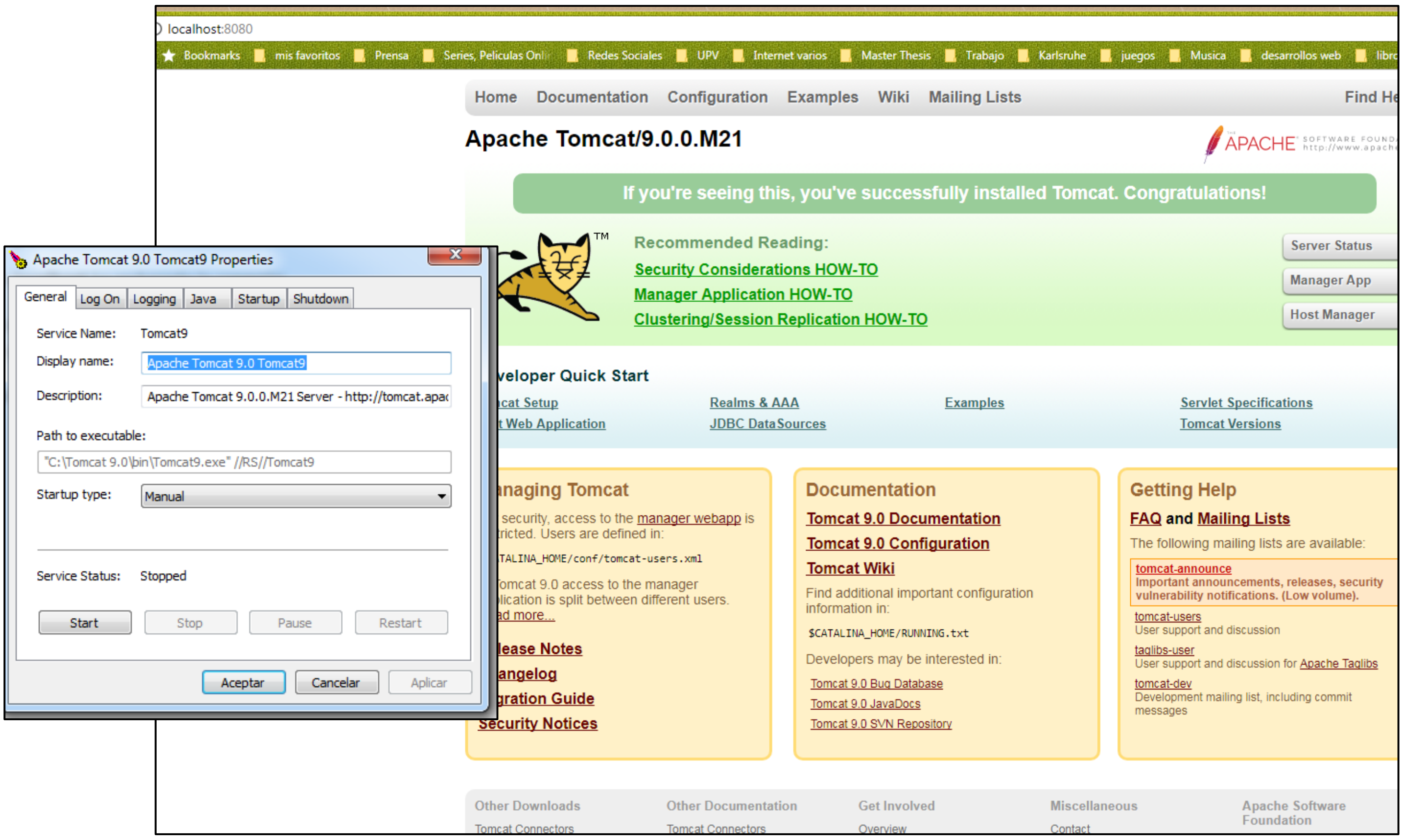


```
        <param-name>showProxySessions</param-name>
        <param-value>true</param-value>
      </init-param>
      -->
      <multipart-config>
        <!-- 5000MB max -->
        <max-file-size>5242880000</max-file-size>
        <max-request-size>5242880000</max-request-size>
        <file-size-threshold>0</file-size-threshold>
      </multipart-config>
    </servlet>
    <servlet>
      <servlet-name>Status</servlet-name>
      <servlet-class>org.apache.catalina.manager.StatusManagerServlet</servlet-class>
      <init-param>
        <param-name>debug</param-name>
        <param-value>0</param-value>
```

```
eXtensible Markup Language file    length : 7.153    lines : 208    Ln : 55    Col : 23    Sel : 0 | 0
```

Figure 1.9: Modifying the file size allowed to install into Tomcat

Open ‘Configure Tomcat.exe’ and click on ‘Start’ button and then open the localhost:8080 website for installing the Geoserver.war file. Then, go to the ‘Manager App’ section on the left side.



Apache Tomcat/9.0.0.M21

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Service Name: Tomcat9
Display name: Apache Tomcat 9.0 Tomcat9
Description: Apache Tomcat 9.0.0.M21 Server - http://tomcat.apac
Path to executable: "C:\Tomcat 9.0\bin\Tomcat9.exe" //RS//Tomcat9
Startup type: Manual
Service Status: Stopped

veloper Quick Start
cat Setup | Realms & AAA | Examples | Servlet Specifications
t Web Application | JDBC DataSources | | Tomcat Versions

naging Tomcat
security, access to the manager webapp is ricted. Users are defined in:
TALINA_HOME/conf/tomcat-users.xml
omcat 9.0 access to the manager lication is split between different users.
ad more...
lease Notes
angelog
gration Guide
Security Notices

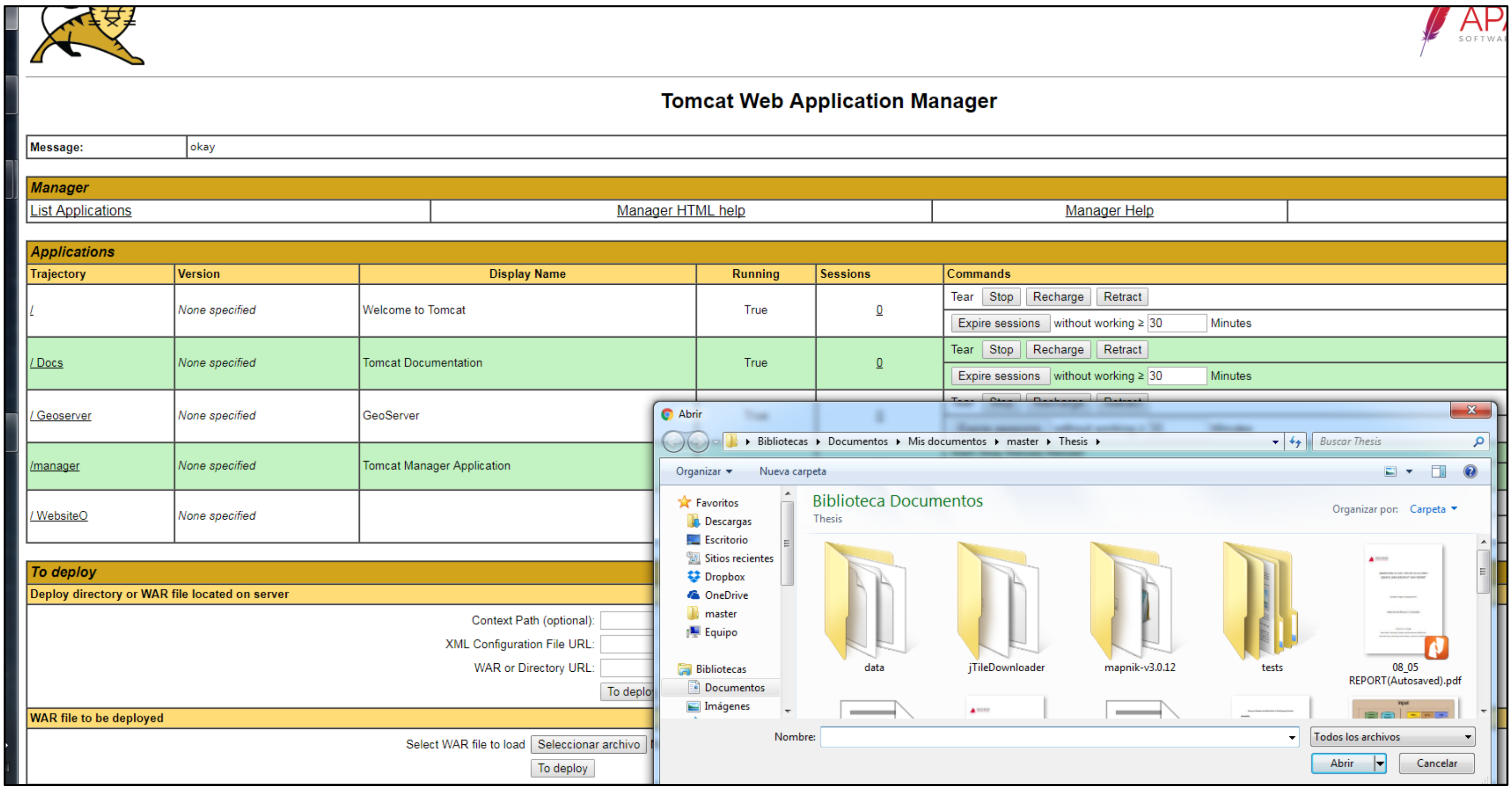
Documentation
Tomcat 9.0 Documentation
Tomcat 9.0 Configuration
Tomcat Wiki
Find additional important configuration information in:
$CATALINA_HOME/RUNNING.txt
Developers may be interested in:
Tomcat 9.0 Bug Database
Tomcat 9.0 JavaDocs
Tomcat 9.0 SVN Repository

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).
tomcat-users
User support and discussion
taglibs-user
User support and discussion for Apache Taglibs
tomcat-dev
Development mailing list, including commit messages

Other Downloads | Other Documentation | Get Involved | Miscellaneous | Apache Software Foundation
Tomcat Connectors | Tomcat Connectors | Overview | Contact |

Figure 1.10: Installing Geoserver.war into Apache Tomcat Server. Part 1

On ‘To deploy’ section, WAR file to be deployed, you must search for the geoserver.war file and then click on ‘to deploy’ (This website is Spanish version translated into English, the buttons’ name may change).


Tomcat Web Application Manager

Message:	okay

Manager

List Applications | Manager HTML help | Manager Help

Applications

Trajectory	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	True	0	Tear Stop Recharge Retract / Expire sessions without working ≥ 30 Minutes
/ Docs	None specified	Tomcat Documentation	True	0	Tear Stop Recharge Retract / Expire sessions without working ≥ 30 Minutes
/ Geoserver	None specified	GeoServer			
/manager	None specified	Tomcat Manager Application			
/ WebsiteO	None specified				

To deploy

Deploy directory or WAR file located on server

Context Path (optional):
XML Configuration File URL:
WAR or Directory URL:

WAR file to be deployed

Select WAR file to load

Figure 1.11:Installing Geoserver.war into Apache Tomcat Server. Part 2

Once installed, go to the next link and should be work.

http://localhost:8080/geoserver

The WMS online resource for Geoserver is:

http://localhost:8080/geoserver/wms?

The maps on Geoserver are created on a Layer Groups section, on the left row of the Geoserver interface, inside Data section. This layer groups can be made of vector and raster files. After doing this, Geoserver creates one WMS URL that is possible to share or include on any web GIS or desktop GIS project.

1.6.3.1. Advantages

It is the easiest opensource map server in the market. It is easy to install, and easy to set and modify the servers from its GUI. It also has its own projection tool for building a web map.

Geoserver can work with all type of data sources: mysql, postgis database, shapefiles from ArcGis…

1.6.3.2. Disadvantages

Geoserver is one of the slower maps server on tile processing and handling small scales on the market.

1.6.4. MapServer MS4W 3.2.1

MapServer is an open source development environment for building spatially-enabled web mapping applications and services. It is fast, flexible, reliable and can be integrated into just about any GIS environment. Originally developed at the University of Minnesota, MapServer is now maintained by developers around the world.

MapServer runs on all major operating systems and will work with almost any web server. MapServer features MapScript, a powerful scripting environment that supports many popular languages including PHP, Python, Perl, C# and Java. Using MapScript makes it fast and easy to build complex geospatial web applications. (Jeff McKenna, 2017)

MapServer for windows download can be found here: http://www.ms4w.com/

MapServer doesn't have a Graphical User Interface by itself, but there are some applications packaged for MS4W which has one GUI. Installing it is as easy as installing Geoserver, but the server management is not so easy. The GUIs available in this web are only viewers. In order to change the server settings, the files within the ms4w folder must be changed. These demos can be used for default websites, switching a few settings depending of the purposes of your website.

MapServer publishes maps on web by .map files. The Mapfile contains a text depicting one map with layers or shapefiles and their styles. The next example shows how it's done.

In this file, the name of the map, the border extent, the directory where the shapefiles are, the size, the image type, the projection and the layers with the shapefiles which will be included in this map are defined.

```
MAP
  NAME "MAPSERVER QUICKSTART"
  EXTENT -137 29 -53 88
  UNITS DD
  SHAPEPATH "/home/user/data/natural_earth2/"
  SIZE 800 600

  IMAGETYPE PNG24

  PROJECTION
    "init=epsg:4326"
  END

  WEB
    METADATA
      ows_enable_request "*"
    END
  END

  LAYER
    NAME "Admin Countries"
    STATUS ON
    TYPE POLYGON
    DATA "ne_10m_admin_0_countries"
    CLASS
      STYLE
        COLOR 246 241 223
        OUTLINECOLOR 0 0 0
      END
    END
  END

END
```

Figure 1.12: Example of MapFile

There is only one java library for handling maps, Openlayers package for MS4W but, it doesn't mean that Leaflet does not work with MapServer.

The WMS online resource for MapServer is: http://localhost/cgi-bin/mapserv.exe?

It's possible to work with .SLD files to add styles to the layers. There are two ways:

-SLD parameter pointing to remote SLD. The SLD link is included on the MapServer request. This link is an example of this way:

```
http://localhost/cgi-
bin/mapserv.exe?map=c:\ms4w\apps\chameleon\cesi\map\mapfile1.map&service=wms&version=1.1.1&request=getlege
ndgraphic&layer=pearseshape&format=image/gif&sld=http://localhost/chameleon/cesi/sldpearseshape.xml
```

In this link, the blue letters are the address which renders the map created, and in green is the address to the SLD file in localhost with xml format.

-SLD_BODY: parameter to send the SLD definition in the URL. All the xml code of one SLD is included on the URL request. If the SLD is depicting many layers, the URL length could be huge.(MapServer Quickstart, n.d.)

MapServer also supports MySQL databases. Inside Mapfile, the connection with a MySQL database can be stablished with a line code, the next line could be an example:

CONNECTION "MySQL:test,user=root,password=mysql,port=3306". (MapServer Quickstart, n.d.).

1.6.4.1. Advantages

Faster tile response than Geoserver and easy to install.

1.6.4.2. Disadvantages

Changing settings is difficult without any GUI like the one Geoserver has, and MapServer management works modifying configuration files.

It’s possible to download Openlayers as packaged applications for MS4W, but not Leaflet. It’s mentioned that there are not Leaflet examples for MapServer, although **Leaflet works on MapServer.**

Creating the Mapfile can be difficult, especially if this file is composed by a lot of layers. Unlike Geoserver, MapServer has not GUI also for making map packages.

1.6.5. Conclusion

MapServer and Geoserver with Apache Tomcat have been chosen as the servers that are going to be tested for this project. They are not the fastest servers, but the website is not working with a lot of tiles at the same time so, the advantage with the other servers doesn’t matter. Furthermore, Geoserver has an easy and intuitive user interface, which makes it the perfect choice for the new website.

2. Handling the settlement development in Vel’ké Pole, Píla and Radobica website data base

After doing the study of the different options on the market, Geoserver and MapServer have been chosen as the final choices for testing the data from the website and, after the test with the data has been performed, choose the best option which works with the same functions than the ESRI website has.

2.1. Preparing the data. Postgres Database

Regardless of working on MapServer or Geoserver, the shapefiles are loaded into a Postgres database. MapServer and Geoserver will load the shapefiles from the same Postgres database. This step gives geographical objects support to the shapefiles and also, it is a safer way for the data integrity because Postgres has also a password system for protecting the database.

Postgres is an open source object-relational database system and provides geographical information to the shapefiles stored within. This software will be used regardless of choosing between Geoserver, MapServer, Openlayers, Leaflet… This software is necessary for all databases that need geographical features. The software can be downloaded from https://www.postgresql.org/.

There are many ways of loading shapefiles into Postgres, the best way is with its shapefile loader, PostGIS. It’s possible to load all the shapefiles at the same time with Postgres but, with several shapefiles to load is better to choose another way. The reason is, there are a few shapefiles with the same name and an error is returned in these situations. Another reason is, if there are some shapefiles with bad integrity data, for example if one shapefiles has recurrent IDs, Postgres will not return any error. Besides, when these wrong layers are loaded into Geoserver or MapServer, the layer isn’t shown. They need a well-structured ID, and if there are two objects with the same ID, there will be a load error to the server.

QGIS Desktop has a tool for loading shapefiles into a Postgres database, and this tool give advices to the users if there is this ID issue.

All the shapefiles are checked when they are loaded into a QGIS project, and this tool returns 52 shapefiles with some ID repeated inside them.

All of these shapefiles are shown on the next list sorted by the folders where are from: slovakei 2014, slovakei 2015 and slovakei 2016.

slovakei2014	**slovakei2015**	**slovakei2016**
Settlement settlement	vp_vp_2014 Settlement_all_2014_update	ArcGIS_Markers. All of them.
Landuse2014agriculture2014	vp_p_1956 River_all_1956	LandUse_1844 RA_Road_1844
Landuse1956river.	vp_p_1956 Roads_all_1956	Buildings_1844_2016 RA_Buildings_1844
Landuse1956road.	vp_p_1782 Rivers_all_1782	
Landuse1965settlement.	vp_p_1782 Roads_all_1782	
Landuse1782landuse	vp_2014 Settlement_VP_2014_update	
Landuse1782River	vp_1956 River_VP_1956	
Landuse1782Road	vp_1956 Roads_VP_1956	
Landuse1782settlement	vp_1782 River_VP_1782	
VP_and_P_2014 Agriculture_Today	vp_1782 Roads_VP_1782	
VP_and_P_2014 Forest_Today	pila_1782 River_Pila_1782	
VP_and_P_2014 Rivers_Today		
VP_and_P_2014 Roads_Today		
VP_and_P_2014 Settlements_Today		
VP_and_P_1860 Agriculture_1860		
VP_and_P_1860 Rivers_1860		
VP_and_P_1860 Forest_1860		
VP_and_P_1860 Roads_1860		
VP_and_P_1860 Settlements_1860		
data agriculture_decrease		
data agriculture_increase		
data agriculture_no_change		
data forest_decrease		
data forest_increase		
data settlements_no_change		
data settlements_decrease		
data settlements_increase		
1956-2014 ForestCommon		
1956-2014 AgriculturalGrowing		
1956-2014 AgriculturalCommon		
1956-2014 UrbanCommon		
1956-2014 UrbanDecreasing		
1860-1956 ForestCommon		
1782-1860 UrbanCommon		
1860-1956 UrbanGrowing		
1782-1860 UrbanDecreasing		

Figure 2.13: Shapefiles with wrong ID

There are two types of errors:

Landuse1783River has only one column: ID. With this sort of files, a new field is created as ID1, and this field is keeping '0' values. For example, Geoserver doesn't create a new column with the ID, takes the existing columns so, in this case, Geoserver takes '0' as ID, but without showing any data because is the only column inside this shapefile. Then, '0' has changed in ID field by the number of the column with QGIS.

Landuse1783Settlement is the other type of files. This shapefile depicts polygons and each ID is for the owners of this polygons which depicts buildings. In this case, the '0' has changed by the row's number, because its original reason of being appears to be only to call all the elements of this shapefile.

Furthermore, there are also a couple of shapefiles more with another problem. Those files are:

- vp_p_1782 Settlement_all_1782
- vp_1782 Settlement_VP_1782

Booth of they from slovakei2015 folder. The changes made are the next:

For vp_p_1782 Settlement_all_1782:

- id. 10 twice, change it to 24
- id.35 twice, change it to 121
- id.43 twice, change it to 122
- id 82 twice, change it to 83
- id 90 twice, change it to 94
- id 99 twice, change it to 97
- id 33 twice, change it to 36
- id 73 twice, change it to 76

For vp_1782 Settlement_VP_1782:

- id. 33 twice, change it to 121
- id.35 twice, change it to 122
- id.43 twice, change it to 123
- id 73 twice, change it to 124
- id. 82 twice, change it to 125
- id.90 twice, change it to 126
- id.99 twice, change it to 127

Besides, there are problems with duplicated shapefiles, as photo shape orientation files.

MapServer, as well as Geoserver, works with map compositions, in other words, a map file has to be created with shapefiles and rasters as layers.

The structure of the website shows geographical information depending on the development's type selected by the user. These developments are settlement, land use, land use change and population density. Likewise, depending of the development which a user can select, it's possible to choose between the villages of Brezova, Vel'ké Pole, Píla and Radobica. Then, the idea is to create one map composition for each village and each development that there are.

Then, the website is able to load all the necessary data and the users are going to be able to switch on or off the layers, which are included in those different map files.

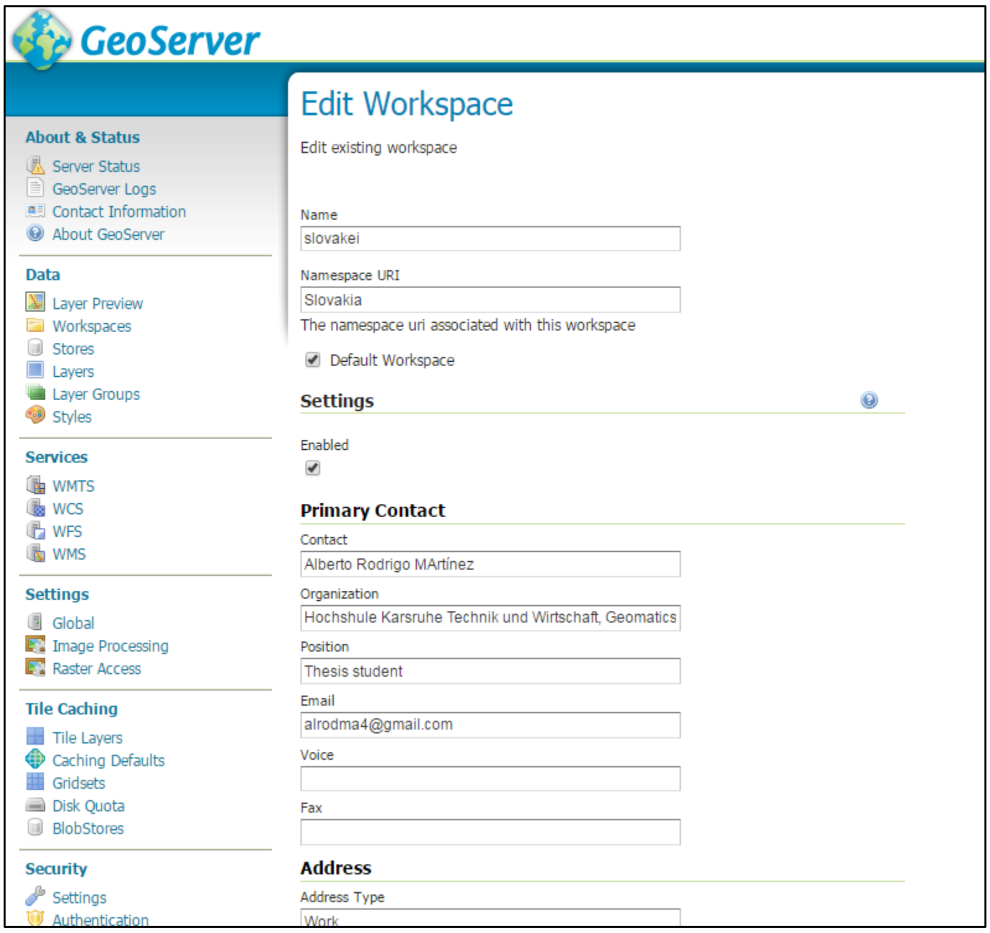
2.2. Geoserver

Geoserver has a GUI unlike MapServer, and with this interface, all the publishing settings can be managed.

Then, a Workspace must be created.

The project's name, the developers name, contact and email is set up in this part. After this, a data store has to be created.

The data source is created depending of the file type which is going to be uploaded. A vector data source, raster data source and even WMS connections can be created.



Edit Workspace

Edit existing workspace

Name: slovakei

Namespace URI: Slovakia

The namespace uri associated with this workspace

☑ Default Workspace

Settings

Enabled ☑

Primary Contact

Contact: Alberto Rodrigo MArtínez

Organization: Hochshule Karsruhe Technik und Wirtschaft, Geomatics

Position: Thesis student

Email: alrodma4@gmail.com

Voice:

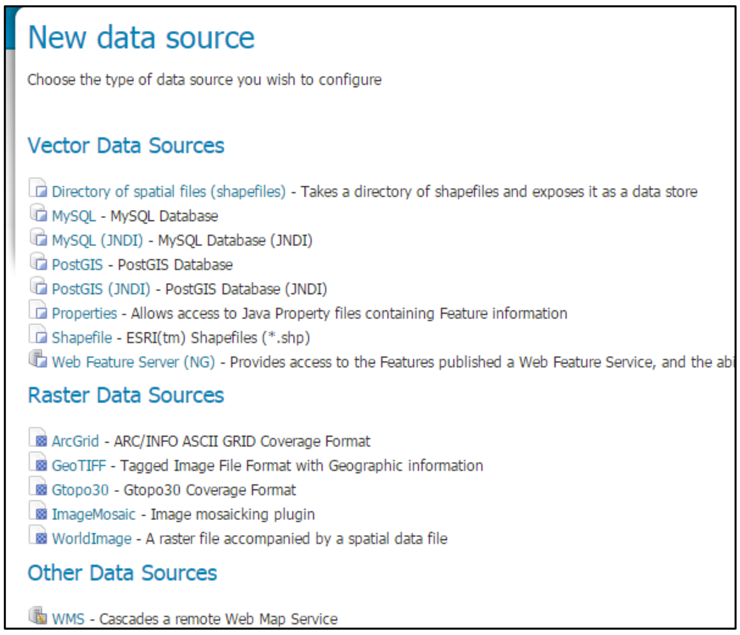
Fax:

Address

Address Type: Work

Figure 2.14: Edit Workspace with Geoserver

There are many choices for uploading different types of data.



New data source

Choose the type of data source you wish to configure

Vector Data Sources

- Directory of spatial files (shapefiles) - Takes a directory of shapefiles and exposes it as a data store
- MySQL - MySQL Database
- MySQL (JNDI) - MySQL Database (JNDI)
- PostGIS - PostGIS Database
- PostGIS (JNDI) - PostGIS Database (JNDI)
- Properties - Allows access to Java Property files containing Feature information
- Shapefile - ESRI(tm) Shapefiles (*.shp)
- Web Feature Server (NG) - Provides access to the Features published a Web Feature Service, and the abi

Raster Data Sources

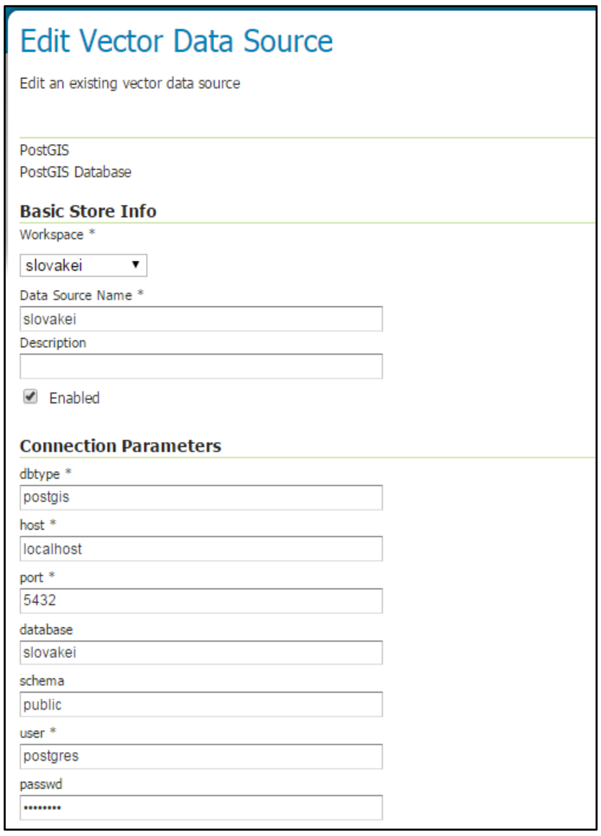
- ArcGrid - ARC/INFO ASCII GRID Coverage Format
- GeoTIFF - Tagged Image File Format with Geographic information
- Gtopo30 - Gtopo30 Coverage Format
- ImageMosaic - Image mosaicking plugin
- WorldImage - A raster file accompanied by a spatial data file

Other Data Sources

- WMS - Cascades a remote Web Map Service

Figure 2.15: Different types of data source can be created in Geoserver

The project works with Postgres data base, for this reason, a PostGIS data base connection with Geoserver has to be created.


Edit Vector Data Source

Edit an existing vector data source

PostGIS
PostGIS Database

Basic Store Info

Workspace *: slovakei

Data Source Name *: slovakei

Description:

☑ Enabled

Connection Parameters

dbtype *: postgis

host *: localhost

port *: 5432

database: slovakei

schema: public

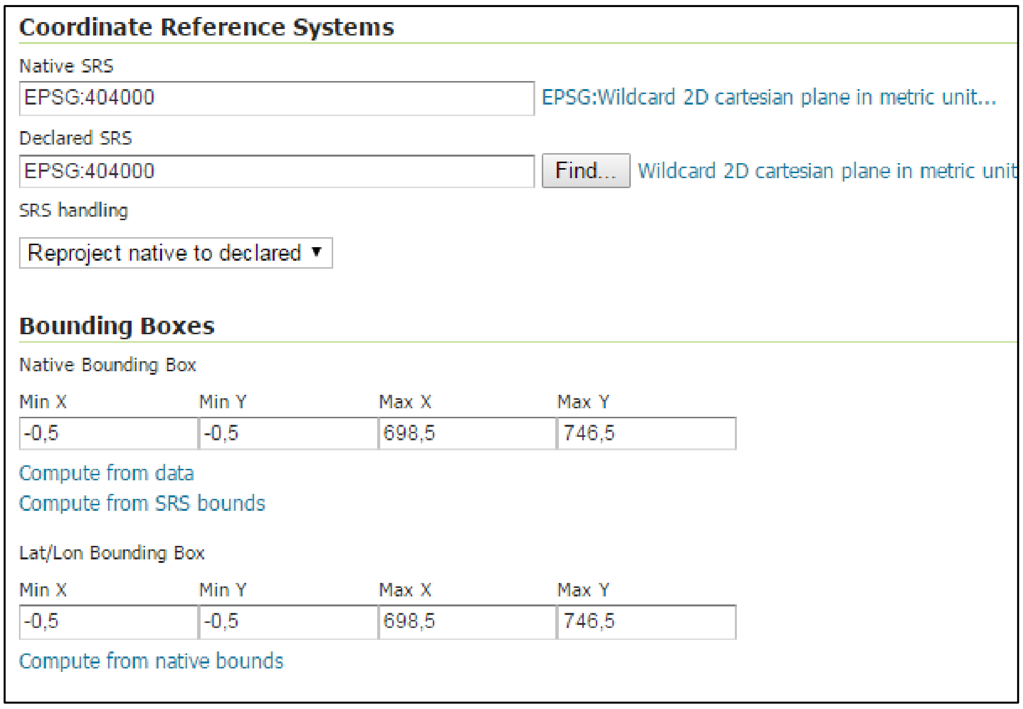
user *: postgres

passwd: ••••••••

Figure 2.16: Creating new Postgres vector data source

Most of the raster files in the project are saved from ArcGIS and, for some reason, this .tif files from ArcGIS doesn’t work after loading them to Geoserver, the problem is with their projection. Although these raster files were saved with EPSG:3857 projection in ArcGIS, when its loaded to Geoserver, Geoserver doesn’t recognize this projection and adds one standard projection for those raster files which doesn’t have native projection EPSG:404000.

Here is an example: Geoserver is able to fill a different Declared SRS, and create the bounding box from this Declared SRS but, is not the solution for this case.


Coordinate Reference Systems

Native SRS: EPSG:404000 EPSG:Wildcard 2D cartesian plane in metric unit...

Declared SRS: EPSG:404000 Find... Wildcard 2D cartesian plane in metric unit

SRS handling: Reproject native to declared

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
-0,5	-0,5	698,5	746,5

Compute from data
Compute from SRS bounds

Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y
-0,5	-0,5	698,5	746,5

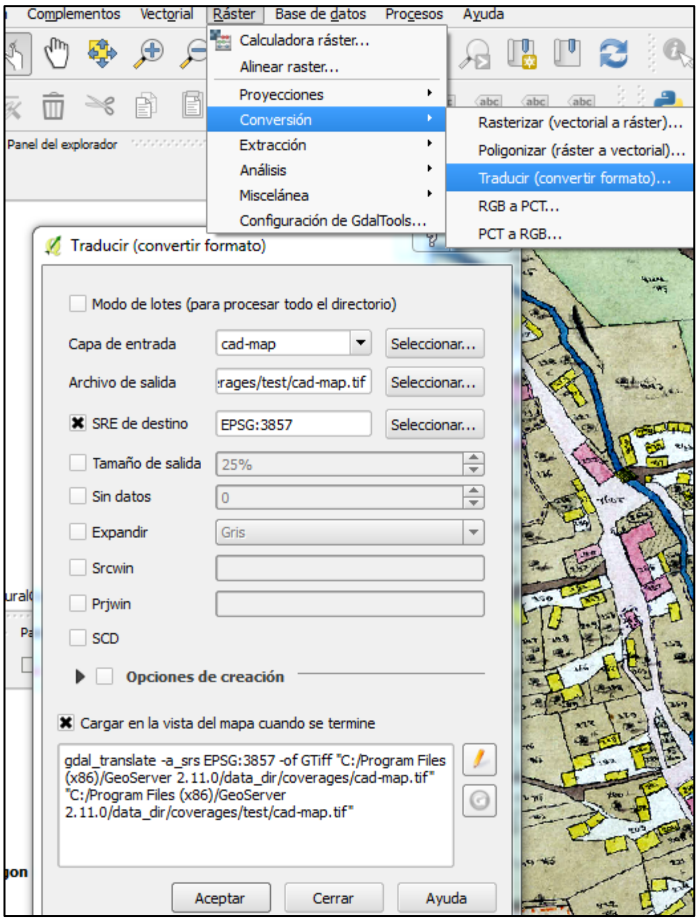
Compute from native bounds

Figure 2.17: Problem loading Raster files to Geoserver

The solution for this problem is opening the raster file from QGIS Desktop Software and save it again as .tif file with his projection and the pyramid image builders has to be allowed. In this way, Geoserver recognize the native projection as the saved previously with QGIS. Maybe it's not a problem working with ArcGIS Server because this server reads perfectly the projection from another ArcMap file, **unlike an open source web server.**

This option can be found on QGIS menu, Raster/ Conversion/ Translate (Convert format) …

Inside the conversion tool window, in the SRS field, its mandatory to assign EPSG:3857 to apply the projection to the output file.



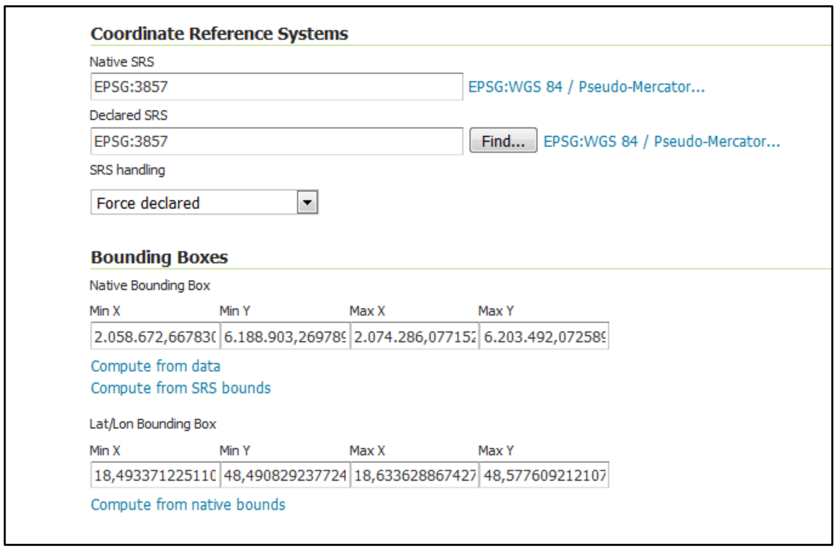
Traducir (convertir formato)

☐ Modo de lotes (para procesar todo el directorio)		
Capa de entrada	cad-map	Seleccionar…
Archivo de salida	rages/test/cad-map.tif	Seleccionar…
☒ SRE de destino	EPSG:3857	Seleccionar…
☐ Tamaño de salida	25%	
☐ Sin datos	0	
☐ Expandir	Gris	
☐ Srcwin		
☐ Prjwin		
☐ SCD		
☐ Opciones de creación		
☒ Cargar en la vista del mapa cuando se termine		

```
gdal_translate -a_srs EPSG:3857 -of GTiff "C:/Program Files
(x86)/GeoServer 2.11.0/data_dir/coverages/cad-map.tif"
"C:/Program Files (x86)/GeoServer
2.11.0/data_dir/coverages/test/cad-map.tif"
```

Figure 2.18:Exporting cad_map to .tif from QGIS Desktop

This capture shows the right coordinates that one layer must have. Bounding Boxes with values like '0.5, -0.5, or 180, 90' are not valid then, if there are layers with those values, it means that this layer is not being loaded right and the last step with QGIS Desktop will be done.



Coordinate Reference Systems

Native SRS

EPSG:3857 EPSG:WGS 84 / Pseudo-Mercator...

Declared SRS

EPSG:3857 Find... EPSG:WGS 84 / Pseudo-Mercator...

SRS handling

Force declared

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
2.058.672,66783(	6.188.903,26978(	2.074.286,07715:	6.203.492,07258(

Compute from data
Compute from SRS bounds

Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y
18,49337122511(	48,490829237724	18,633628867427	48,577609212107

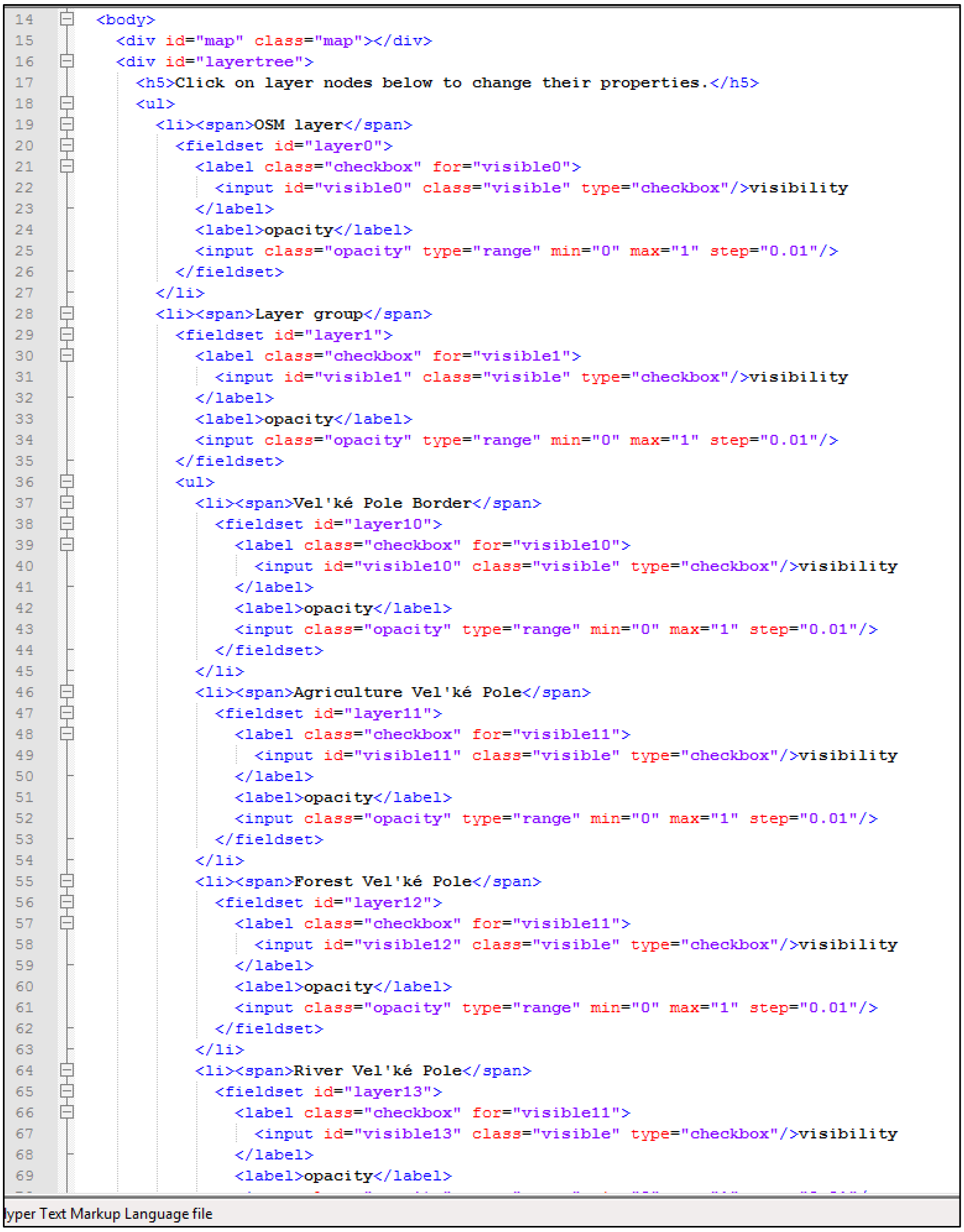
Compute from native bounds

Figure 2.19: Right Bounding Boxes for the layers of the project

The idea for now is about creating some map compositions depending of each user. An example for Vel'ké Pole in 1782 was created for explaining how to make one of these map compositions. This is an example about a simple HTML working with Openlayers 3, which loads the shapefiles from Geoserver, after loading them from Postgres, and that HTML is placed on apache server then, this HTML will work on localhost.

2.2.1. Building an HTML example for Geoserver with the original database

Previously it has been mentioned how Layer Groups Control in Geoserver can help us to create these compositions but, for creating the HTML it is better to load them separated. If a layer group is called from the HTML, it’s difficult to assign a ID for each layer which is inside this Openlayers layer group control. This library works saving the layers in the same order that is defined inside the map function.



```html
<body>
  <div id="map" class="map"></div>
  <div id="layertree">
    <h5>Click on layer nodes below to change their properties.</h5>
    <ul>
      <li><span>OSM layer</span>
        <fieldset id="layer0">
          <label class="checkbox" for="visible0">
            <input id="visible0" class="visible" type="checkbox"/>visibility
          </label>
          <label>opacity</label>
          <input class="opacity" type="range" min="0" max="1" step="0.01"/>
        </fieldset>
      </li>
      <li><span>Layer group</span>
        <fieldset id="layer1">
          <label class="checkbox" for="visible1">
            <input id="visible1" class="visible" type="checkbox"/>visibility
          </label>
          <label>opacity</label>
          <input class="opacity" type="range" min="0" max="1" step="0.01"/>
        </fieldset>
        <ul>
          <li><span>Vel'ké Pole Border</span>
            <fieldset id="layer10">
              <label class="checkbox" for="visible10">
                <input id="visible10" class="visible" type="checkbox"/>visibility
              </label>
              <label>opacity</label>
              <input class="opacity" type="range" min="0" max="1" step="0.01"/>
            </fieldset>
          </li>
          <li><span>Agriculture Vel'ké Pole</span>
            <fieldset id="layer11">
              <label class="checkbox" for="visible11">
                <input id="visible11" class="visible" type="checkbox"/>visibility
              </label>
              <label>opacity</label>
              <input class="opacity" type="range" min="0" max="1" step="0.01"/>
            </fieldset>
          </li>
          <li><span>Forest Vel'ké Pole</span>
            <fieldset id="layer12">
              <label class="checkbox" for="visible11">
                <input id="visible12" class="visible" type="checkbox"/>visibility
              </label>
              <label>opacity</label>
              <input class="opacity" type="range" min="0" max="1" step="0.01"/>
            </fieldset>
          </li>
          <li><span>River Vel'ké Pole</span>
            <fieldset id="layer13">
              <label class="checkbox" for="visible11">
                <input id="visible13" class="visible" type="checkbox"/>visibility
              </label>
              <label>opacity</label>
```

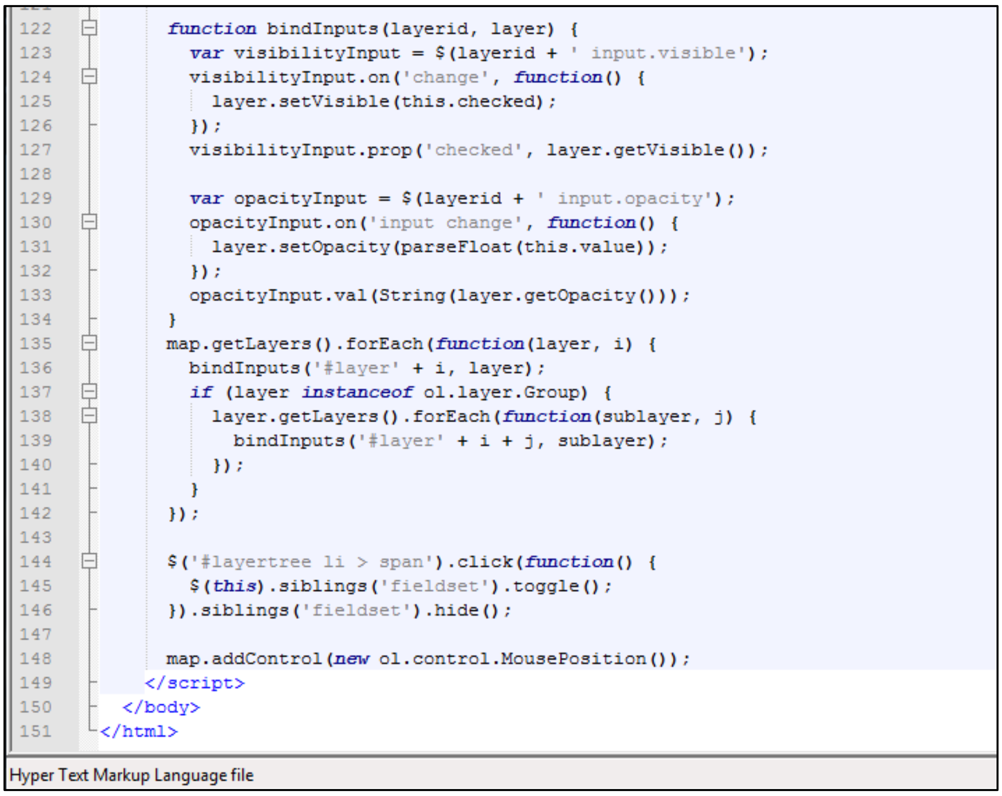
Figure 2.20: Openlayers Layers Groups with ID from Geoserver. Part one

Slovakia website was designed after loading the data from ESRI and a website was programmed with JavaScript scripts but, some of these tools can be managed with the new Openlayers tools, like ‘Layer Groups’. The documentation can be found in References (Layer Groups, n.d.).

The last figure shows how this Layer Groups Control is defined.

Here it is shown how the layers are called by each checkbox for switching on or off the layer on the map.

The function (figure 2.21) works saving the layers defined on map script in the same order than they are called, inside the layer source (check figure 2.26) the objects with id 'layer10', 'layer11', 'layer12', and so on. Then, inside the fieldset, those ids are called as it is shown in last figure 2.21.



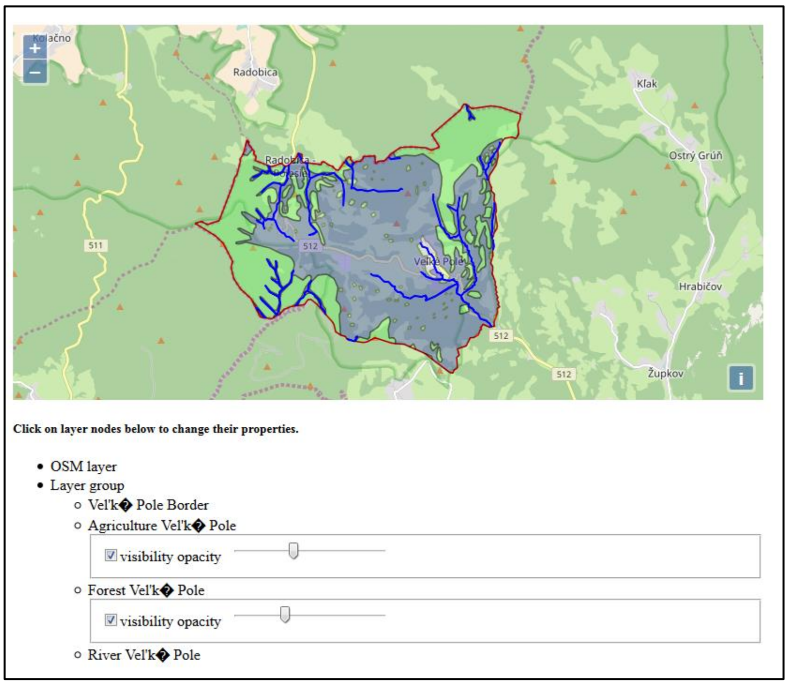
```
      function bindInputs(layerid, layer) {
        var visibilityInput = $(layerid + ' input.visible');
        visibilityInput.on('change', function() {
          layer.setVisible(this.checked);
        });
        visibilityInput.prop('checked', layer.getVisible());

        var opacityInput = $(layerid + ' input.opacity');
        opacityInput.on('input change', function() {
          layer.setOpacity(parseFloat(this.value));
        });
        opacityInput.val(String(layer.getOpacity()));
      }
      map.getLayers().forEach(function(layer, i) {
        bindInputs('#layer' + i, layer);
        if (layer instanceof ol.layer.Group) {
          layer.getLayers().forEach(function(sublayer, j) {
            bindInputs('#layer' + i + j, sublayer);
          });
        }
      });

      $('#layertree li > span').click(function() {
        $(this).siblings('fieldset').toggle();
      }).siblings('fieldset').hide();

      map.addControl(new ol.control.MousePosition());
    </script>
  </body>
</html>
```

Hyper Text Markup Language file

Figure 2.21: Openlayers Layers' Groups with ID from Geoserver. Part two

The code line for loading the shapefiles from Geoserver will be explained hereunder.

With this tool, it’s possible to switch the layer’s opacity also, just like the ESRI website does. In this example, the development for Vel’ké Pole in 1782 is shown.

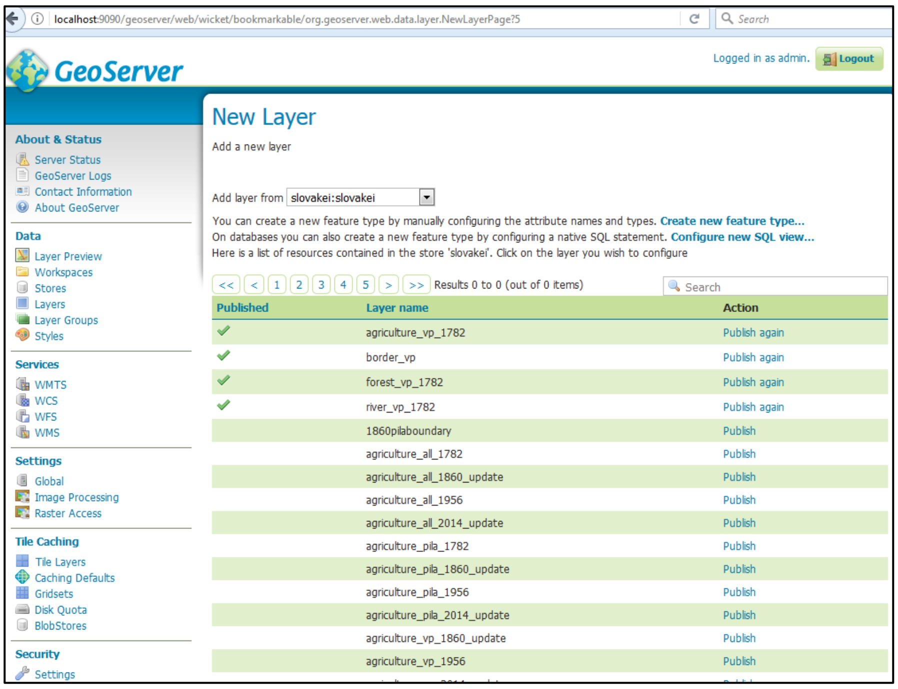


Click on layer nodes below to change their properties.

- OSM layer
- Layer group
  - Vel'k� Pole Border
  - Agriculture Vel'k� Pole
    - ☑ visibility opacity
  - Forest Vel'k� Pole
    - ☑ visibility opacity
  - River Vel'k� Pole

Figure 2.22: Openlayers Layers Group tool for Vel'ké Pole in 1782

First of all, the layers from Geoserver have to be published on this example:



New Layer

Add a new layer

Add layer from slovakei:slovakei

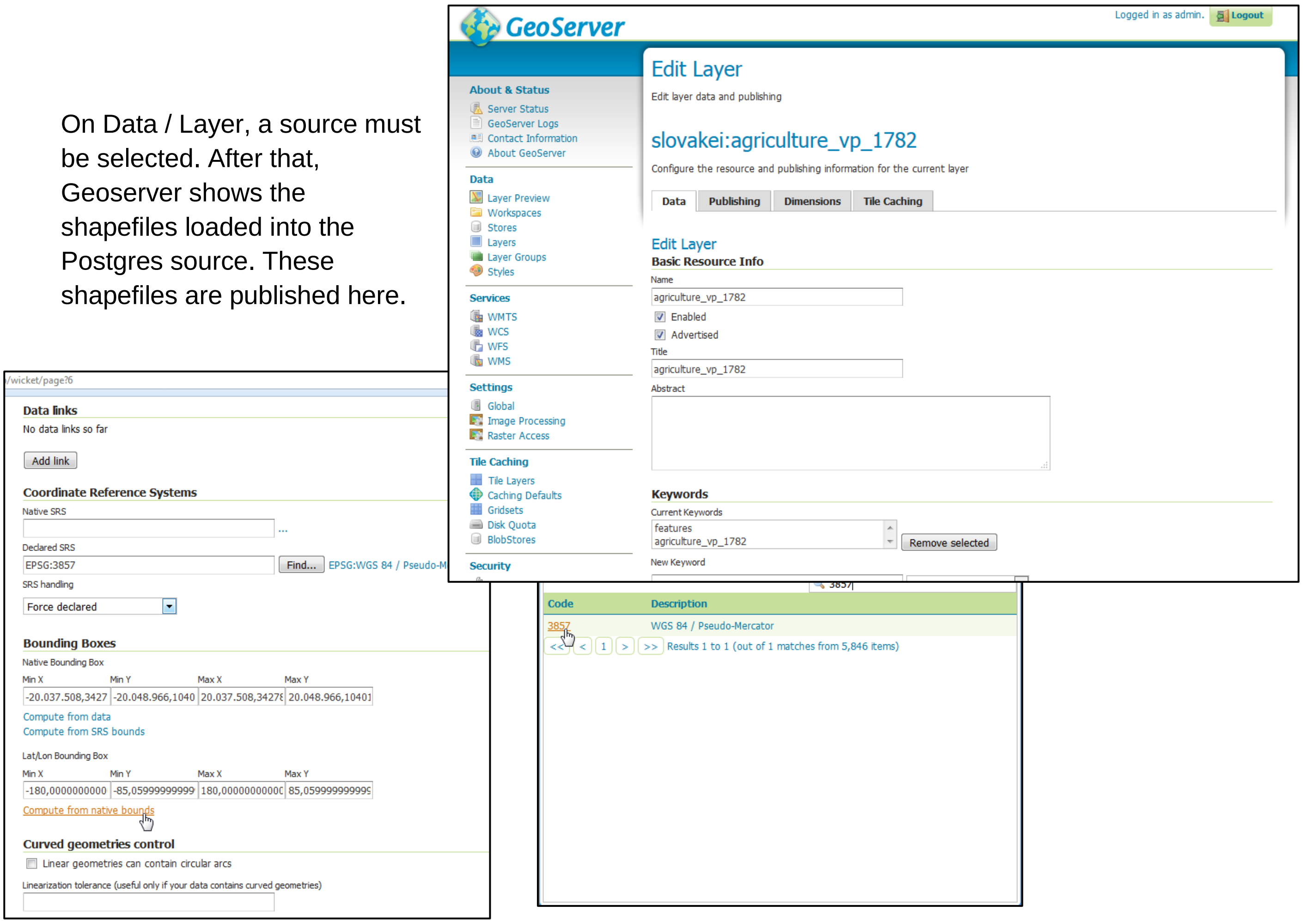
You can create a new feature type by manually configuring the attribute names and types. **Create new feature type...**
On databases you can also create a new feature type by configuring a native SQL statement. **Configure new SQL view...**
Here is a list of resources contained in the store 'slovakei'. Click on the layer you wish to configure

<< < 1 2 3 4 5 > >> Results 0 to 0 (out of 0 items)

Published	Layer name	Action
✓	agriculture_vp_1782	Publish again
✓	border_vp	Publish again
✓	forest_vp_1782	Publish again
✓	river_vp_1782	Publish again
	1860pilaboundary	Publish
	agriculture_all_1782	Publish
	agriculture_all_1860_update	Publish
	agriculture_all_1956	Publish
	agriculture_all_2014_update	Publish
	agriculture_pila_1782	Publish
	agriculture_pila_1860_update	Publish
	agriculture_pila_1956	Publish
	agriculture_pila_2014_update	Publish
	agriculture_vp_1860_update	Publish
	agriculture_vp_1956	Publish

Figure 2.23: Publishing layers from Geoserver

On Data / Layer, a source must be selected. After that, Geoserver shows the shapefiles loaded into the Postgres source. These shapefiles are published here.


GeoServer

Logged in as admin. Logout

About & Status
Server Status
GeoServer Logs
Contact Information
About GeoServer

Data
Layer Preview
Workspaces
Stores
Layers
Layer Groups
Styles

Services
WMTS
WCS
WFS
WMS

Settings
Global
Image Processing
Raster Access

Tile Caching
Tile Layers
Caching Defaults
Gridsets
Disk Quota
BlobStores

Security

Edit Layer

Edit layer data and publishing

slovakei:agriculture_vp_1782

Configure the resource and publishing information for the current layer

Data | Publishing | Dimensions | Tile Caching

Edit Layer

Basic Resource Info

Name: agriculture_vp_1782

☑ Enabled

☑ Advertised

Title: agriculture_vp_1782

Abstract

Keywords

Current Keywords: features, agriculture_vp_1782 — Remove selected

New Keyword

/wicket/page?6

Data links

No data links so far

Add link

Coordinate Reference Systems

Native SRS

Declared SRS: EPSG:3857 — Find... — EPSG:WGS 84 / Pseudo-M

SRS handling: Force declared

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
-20.037.508,3427	-20.048.966,1040	20.037.508,3427ε	20.048.966,10401

Compute from data

Compute from SRS bounds

Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y
-180,0000000000	-85,05999999999	180,000000000C	85,059999999995

Compute from native bounds

Curved geometries control

☐ Linear geometries can contain circular arcs

Linearization tolerance (useful only if your data contains curved geometries)

3857

Code	Description
3857	WGS 84 / Pseudo-Mercator

<< < 1 > >> Results 1 to 1 (out of 1 matches from 5,846 items)

Figure 2.24: Setting the publishing data

That is the first window which is shown on the publishing layer tool. It has four headers: Data, Publishing, Dimensions and Tile Caching. Data and Publishing contains the mandatory information which must be filled up so, for now, only these two parts will be changed.

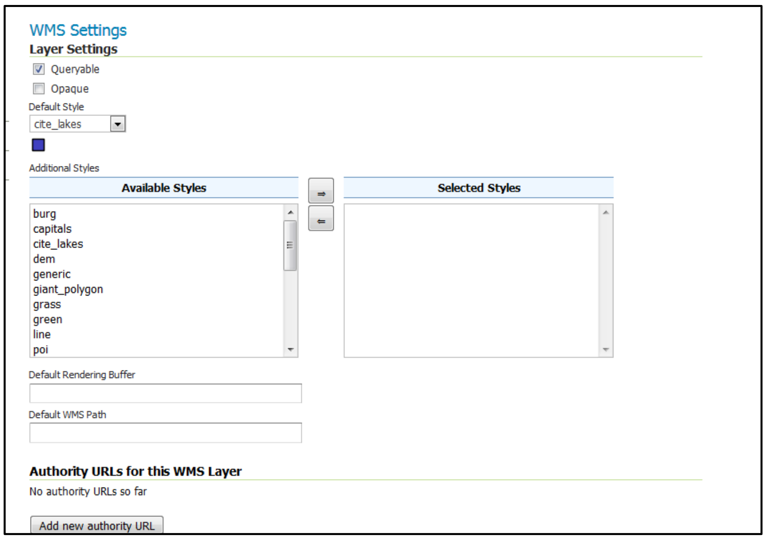
This mandatory information is about the projection of the shapefile and its bounding boxes. The EPSG of this shapefile which is working with can be found clicking on ‘Find…’ button.

After this, click on ‘Compute from SRS bounds’ button on Bounding Boxes section and ‘Compute from native bounds’ button.

With this step, the shapefile will have the right projection and bounding box.

Afterwards, inside Publishing header, the symbology can be stablished. It is not mandatory but, if this is not set up, the WMS will not be shown.

The symbology can be created or loaded in other section but for now, the default layer symbology from Geoserver will be used. Later it will be explained one example for styling raster layers.



WMS Settings

Layer Settings

☑ Queryable

☐ Opaque

Default Style: cite_lakes

Additional Styles

Available Styles: burg, capitals, cite_lakes, dem, generic, giant_polygon, grass, green, line, poi

Selected Styles

Default Rendering Buffer

Default WMS Path

Authority URLs for this WMS Layer

No authority URLs so far

Add new authority URL

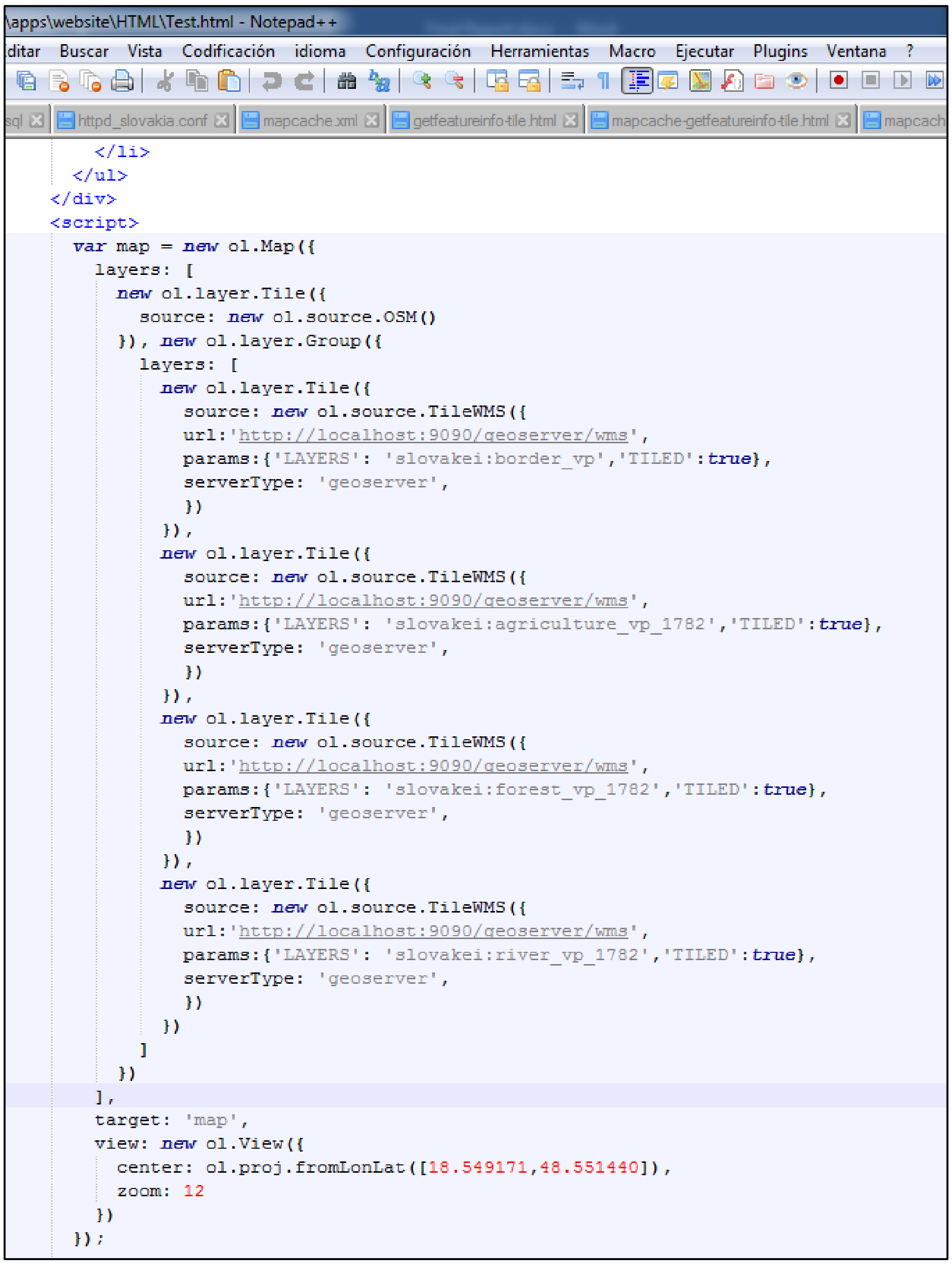
Figure 2.25: Setting the **shapefile symbology** on Geoserver

This process will be repeated with Forest, River and Border layer from Vel'ké Pole in 1782.

And this is all the necessary for publishing maps on web with Geoserver. From now, the Openlayers HTML will be created.

First of all, the website needs a base map layer to support the shapefiles, for this reason, the OSM layer for Openlayers will be loaded.

Then, for loading a shapefile from Geoserver to HTML with Openlayers, the code would be like the next figure.

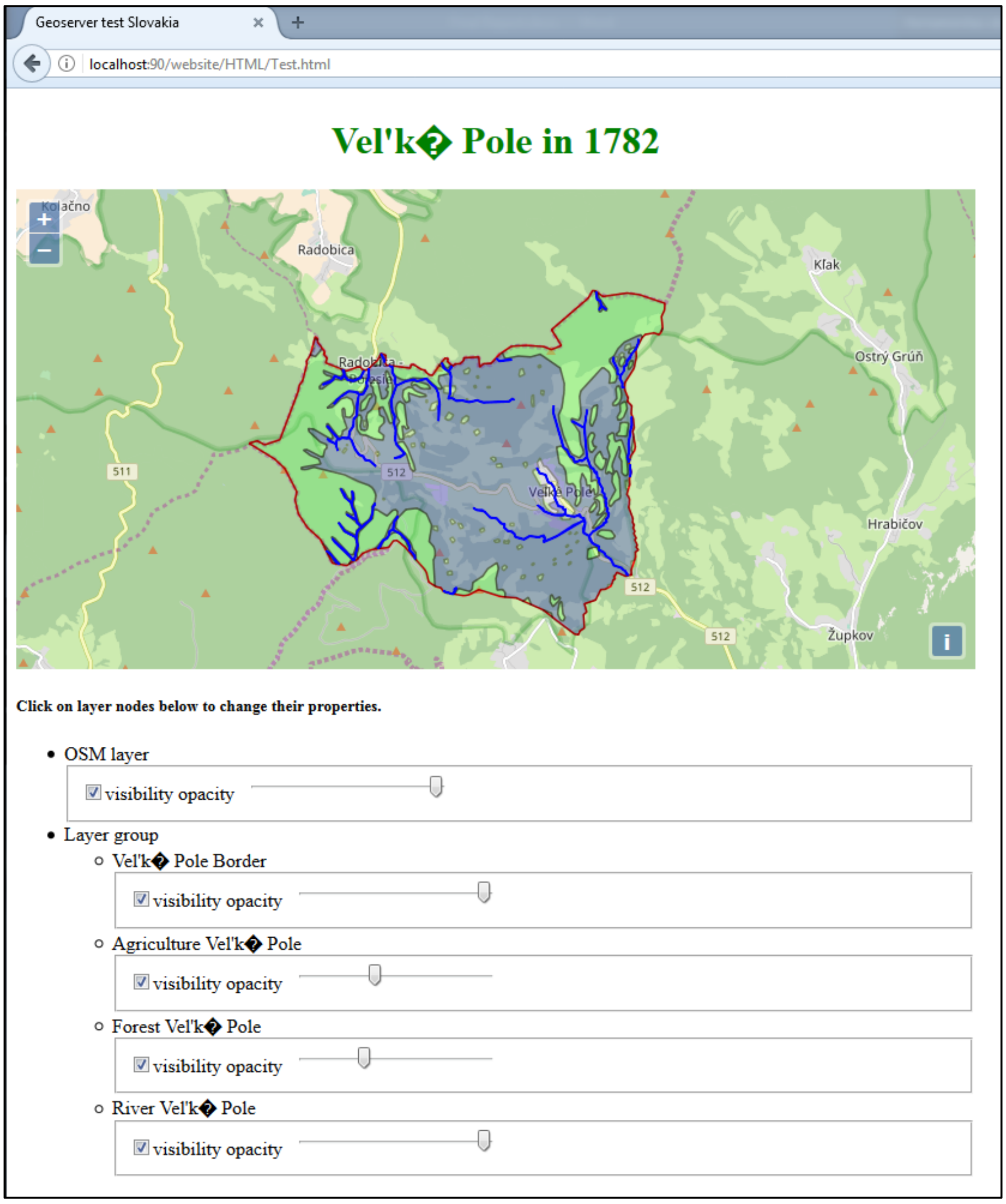

\apps\website\HTML\Test.html - Notepad++
ditar Buscar Vista Codificación idioma Configuración Herramientas Macro Ejecutar Plugins Ventana ?
sql
httpd_slovakia.conf
mapcache.xml
getfeatureinfo-tile.html
mapcache-getfeatureinfo-tile.html
mapcach

```
      </li>
    </ul>
  </div>
  <script>
    var map = new ol.Map({
      layers: [
        new ol.layer.Tile({
          source: new ol.source.OSM()
        }), new ol.layer.Group({
          layers: [
            new ol.layer.Tile({
              source: new ol.source.TileWMS({
              url:'http://localhost:9090/geoserver/wms',
              params:{'LAYERS': 'slovakei:border_vp','TILED':true},
              serverType: 'geoserver',
              })
            }),
            new ol.layer.Tile({
              source: new ol.source.TileWMS({
              url:'http://localhost:9090/geoserver/wms',
              params:{'LAYERS': 'slovakei:agriculture_vp_1782','TILED':true},
              serverType: 'geoserver',
              })
            }),
            new ol.layer.Tile({
              source: new ol.source.TileWMS({
              url:'http://localhost:9090/geoserver/wms',
              params:{'LAYERS': 'slovakei:forest_vp_1782','TILED':true},
              serverType: 'geoserver',
              })
            }),
            new ol.layer.Tile({
              source: new ol.source.TileWMS({
              url:'http://localhost:9090/geoserver/wms',
              params:{'LAYERS': 'slovakei:river_vp_1782','TILED':true},
              serverType: 'geoserver',
              })
            })
          ]
        })
      ],
      target: 'map',
      view: new ol.View({
        center: ol.proj.fromLonLat([18.549171,48.551440]),
        zoom: 12
      })
    });
```

Figure 2.26: Part of code about the loading from Geoserver

The HTML created is shown in this figure.

The example shows the possibility of adapting the new website version with Openlayers javascript library.



Vel'k� Pole in 1782

Click on layer nodes below to change their properties.

- OSM layer
  - ☑ visibility opacity
- Layer group
  - Vel'k� Pole Border
    - ☑ visibility opacity
  - Agriculture Vel'k� Pole
    - ☑ visibility opacity
  - Forest Vel'k� Pole
    - ☑ visibility opacity
  - River Vel'k� Pole
    - ☑ visibility opacity

Figure 2.27: Layer control of Openlayers example

2.3. MapServer

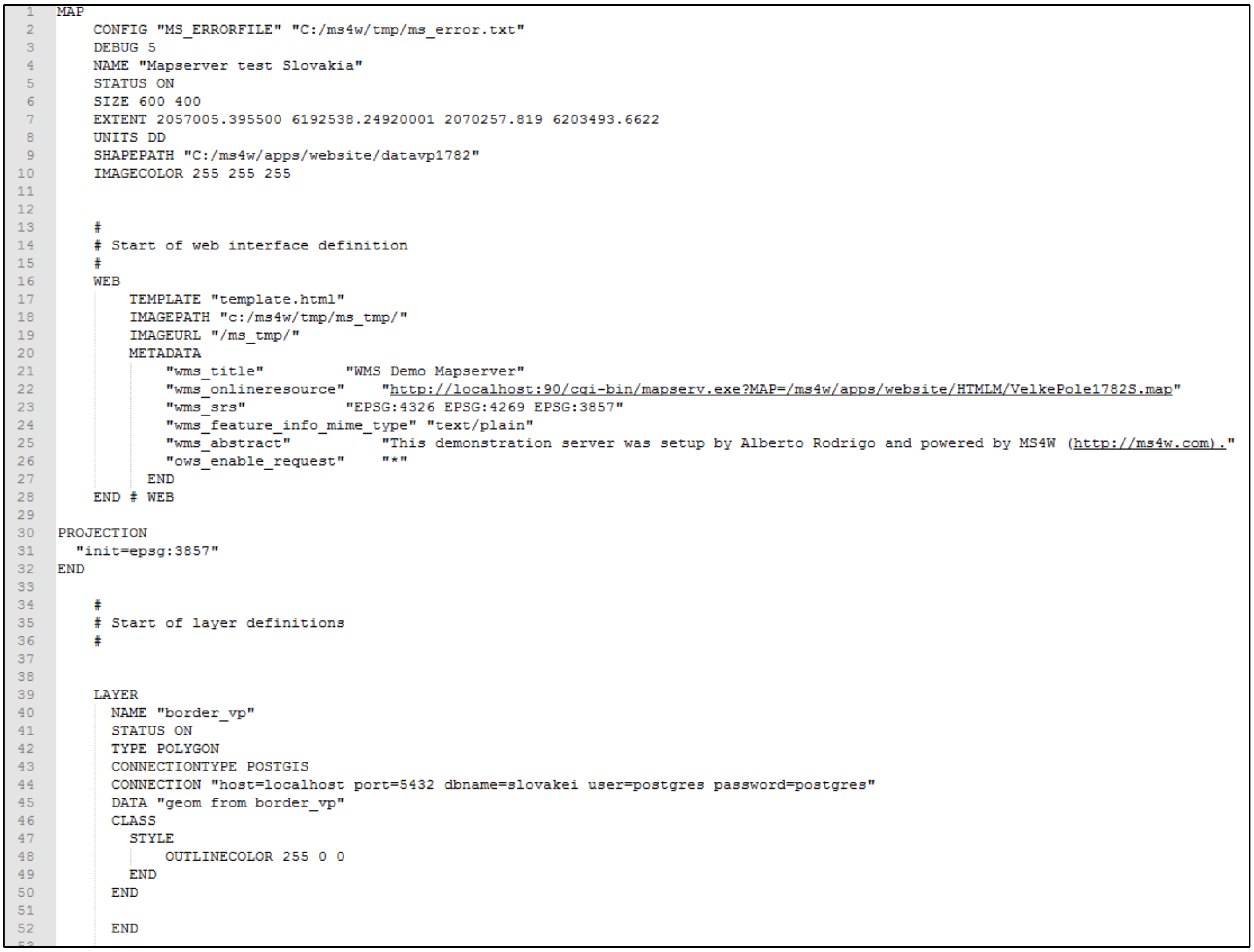
MapServer does not have a GUI. The WMS management and creation depends on line codes from some files. The most important file is the Mapfile, where the maps composition is defined by vector and/or raster files.

The Mapfile example from the MapServer section 1.5.4, page 21, is taken and changed by the new server options.

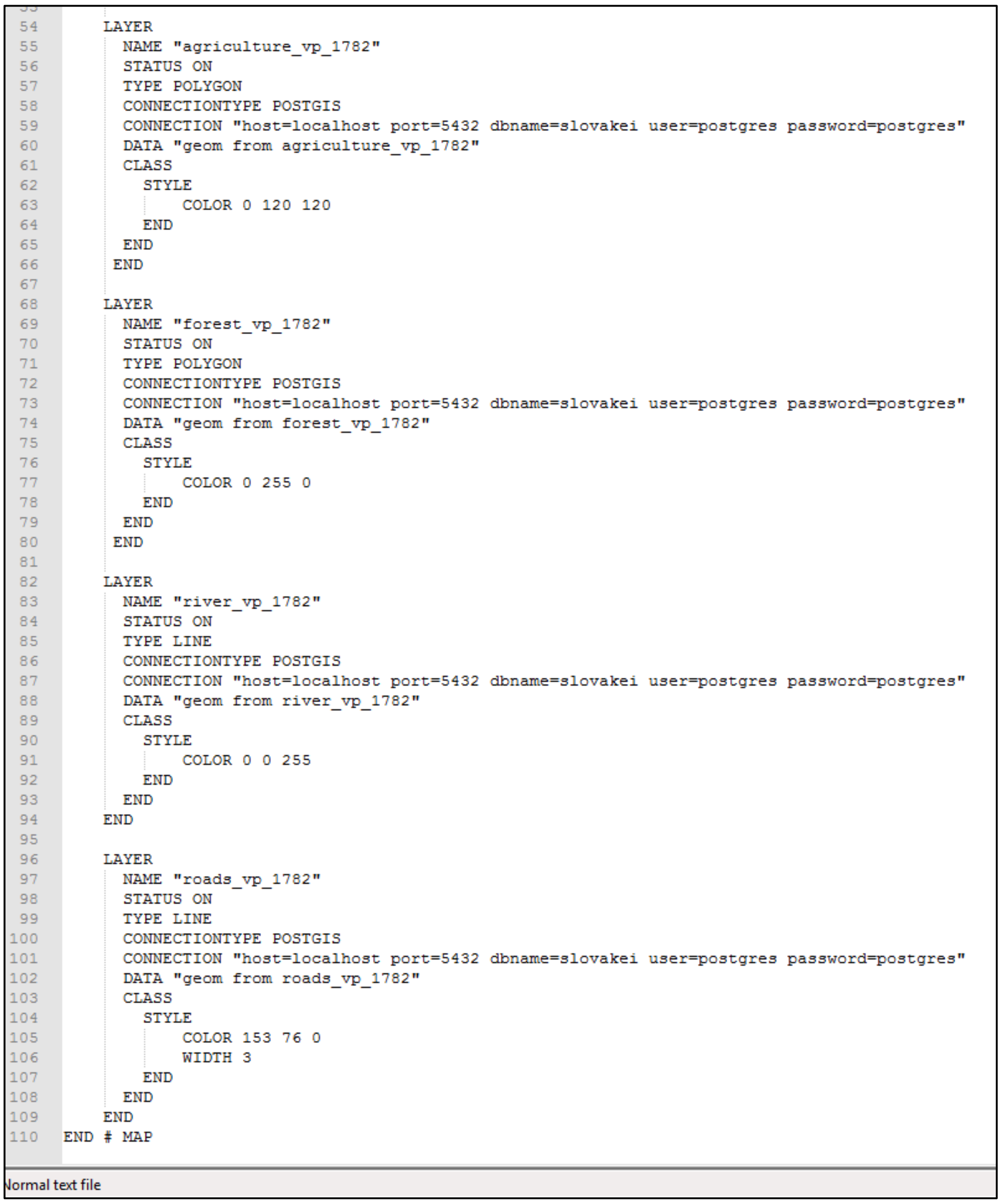
In this example, the shapefiles are taken from the hard drive but, in the new Mapfile, the shapefiles will be loaded by Postgres database. The next figure shows how the Postgres connection was created for all the layers. The layers are the same as Geoserver example has.

MapServer is able to create images from the shapefiles within this Mapfile.

On line 17, it is shown the template which is created for exporting images but, this is not relevant for the project.



```
MAP
    CONFIG "MS_ERRORFILE" "C:/ms4w/tmp/ms_error.txt"
    DEBUG 5
    NAME "Mapserver test Slovakia"
    STATUS ON
    SIZE 600 400
    EXTENT 2057005.395500 6192538.24920001 2070257.819 6203493.6622
    UNITS DD
    SHAPEPATH "C:/ms4w/apps/website/datavp1782"
    IMAGECOLOR 255 255 255


    #
    # Start of web interface definition
    #
    WEB
        TEMPLATE "template.html"
        IMAGEPATH "c:/ms4w/tmp/ms_tmp/"
        IMAGEURL "/ms_tmp/"
        METADATA
            "wms_title"          "WMS Demo Mapserver"
            "wms_onlineresource"    "http://localhost:90/cgi-bin/mapserv.exe?MAP=/ms4w/apps/website/HTMLM/VelkePole1782S.map"
            "wms_srs"            "EPSG:4326 EPSG:4269 EPSG:3857"
            "wms_feature_info_mime_type" "text/plain"
            "wms_abstract"          "This demonstration server was setup by Alberto Rodrigo and powered by MS4W (http://ms4w.com)."
            "ows_enable_request"    "*"
          END
    END # WEB

PROJECTION
  "init=epsg:3857"
END

    #
    # Start of layer definitions
    #


    LAYER
      NAME "border_vp"
      STATUS ON
      TYPE POLYGON
      CONNECTIONTYPE POSTGIS
      CONNECTION "host=localhost port=5432 dbname=slovakei user=postgres password=postgres"
      DATA "geom from border_vp"
      CLASS
        STYLE
            OUTLINECOLOR 255 0 0
        END
      END

      END
```



```
    LAYER
      NAME "agriculture_vp_1782"
      STATUS ON
      TYPE POLYGON
      CONNECTIONTYPE POSTGIS
      CONNECTION "host=localhost port=5432 dbname=slovakei user=postgres password=postgres"
      DATA "geom from agriculture_vp_1782"
      CLASS
        STYLE
            COLOR 0 120 120
        END
      END
     END

    LAYER
      NAME "forest_vp_1782"
      STATUS ON
      TYPE POLYGON
      CONNECTIONTYPE POSTGIS
      CONNECTION "host=localhost port=5432 dbname=slovakei user=postgres password=postgres"
      DATA "geom from forest_vp_1782"
      CLASS
        STYLE
            COLOR 0 255 0
        END
      END
     END

    LAYER
      NAME "river_vp_1782"
      STATUS ON
      TYPE LINE
      CONNECTIONTYPE POSTGIS
      CONNECTION "host=localhost port=5432 dbname=slovakei user=postgres password=postgres"
      DATA "geom from river_vp_1782"
      CLASS
        STYLE
            COLOR 0 0 255
        END
      END
    END

    LAYER
      NAME "roads_vp_1782"
      STATUS ON
      TYPE LINE
      CONNECTIONTYPE POSTGIS
      CONNECTION "host=localhost port=5432 dbname=slovakei user=postgres password=postgres"
      DATA "geom from roads_vp_1782"
      CLASS
        STYLE
            COLOR 153 76 0
            WIDTH 3
        END
      END
    END
END # MAP
```

Normal text file

Figure 2.28: VelkePole1782S.map file structure

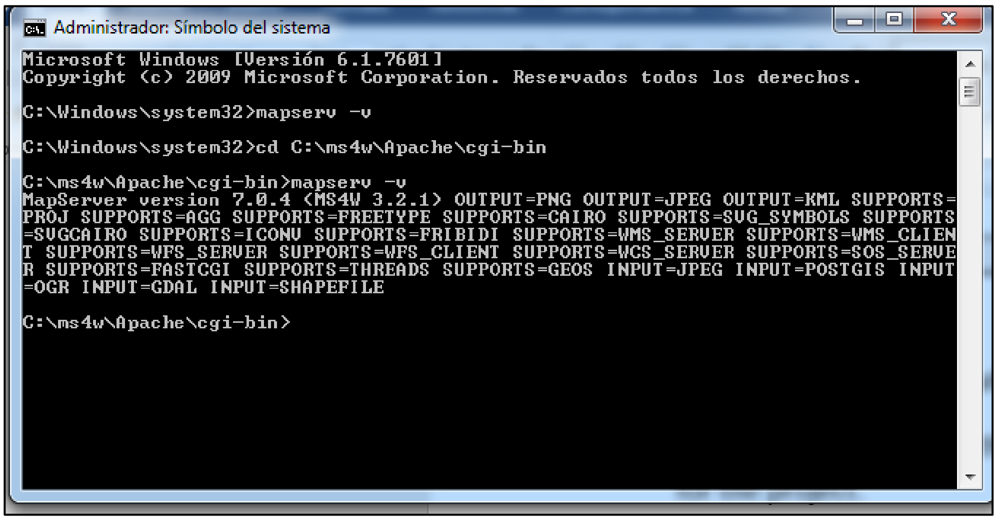
MapServer works with a lot of dependent files so, it's important to check if the actual installed version of MapServer is able to publish WMS.

For testing this, a command windows prompt has to be opened and execute "mapserv -v" inside cgi-bin folder within ms4w apache directory.

At the MapServer directory, there is a single file referred as "mapserv". With the MS4W package, it is a single executable file found at **C:\ms4w\Apache\cgi-bin\mapserv.exe** (by default). This file is the map production engine that uses a Mapfile to generate a map image.

Mapserv.exe is usually accessed internally by MapServer-based packages such as ka-map or Mapscript. But it can also be called directly from a web browser as long as some required parameters are included.

In this figure, you can see all the formats supported by the MapServer version 7.0.4. which is included on MS4W install



```
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Windows\system32>mapserv -v

C:\Windows\system32>cd C:\ms4w\Apache\cgi-bin

C:\ms4w\Apache\cgi-bin>mapserv -v
MapServer version 7.0.4 (MS4W 3.2.1) OUTPUT=PNG OUTPUT=JPEG OUTPUT=KML SUPPORTS=
PROJ SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=CAIRO SUPPORTS=SVG_SYMBOLS SUPPORTS
=SVGCAIRO SUPPORTS=ICONV SUPPORTS=FRIBIDI SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIEN
T SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVE
R SUPPORTS=FASTCGI SUPPORTS=THREADS SUPPORTS=GEOS INPUT=JPEG INPUT=POSTGIS INPUT
=OGR INPUT=GDAL INPUT=SHAPEFILE

C:\ms4w\Apache\cgi-bin>
```

Figure 2.29: Formats supported by MapServer

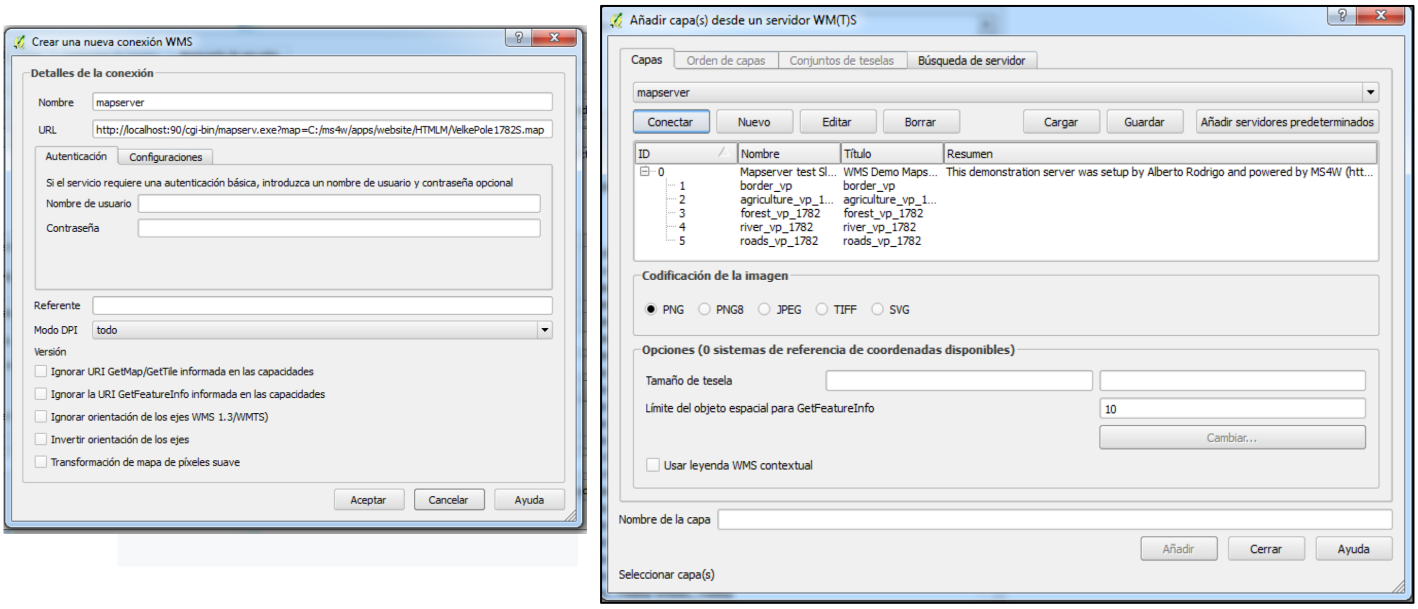
This command prompt is showing that this MapServer version works with WMS server and WMS client then, WMS server should work.

One good way to prove if MapServer WMS is working is testing with QGIS Desktop software. The Url which has to be added as WMS is:

http://localhost:90/cgi-bin/mapserv.exe?map=C:/ms4w/apps/website/HTMLM/VelkePole1782S.map

After creating a new server with the last link (left picture) and clicking on connect (Conectar), all the layers within the MapFile file will be shown.

In the next figure, the added layer from WM(T)S of the server is shown:



Detalles de la conexión

Nombre: mapserver

URL: http://localhost:90/cgi-bin/mapserv.exe?map=C:/ms4w/apps/website/HTMLM/VelkePole1782S.map

Si el servicio requiere una autenticación básica, introduzca un nombre de usuario y contraseña opcional

Nombre de usuario:

Contraseña:

Referente:

Modo DPI: todo

Versión

- [ ] Ignorar URI GetMap/GetTile informada en las capacidades
- [ ] Ignorar la URI GetFeatureInfo informada en las capacidades
- [ ] Ignorar orientación de los ejes WMS 1.3/WMTS)
- [ ] Invertir orientación de los ejes
- [ ] Transformación de mapa de píxeles suave

mapserver

ID	Nombre	Título	Resumen
0	Mapserver test Sl...	WMS Demo Maps...	This demonstration server was setup by Alberto Rodrigo and powered by MS4W (htt...
1	border_vp	border_vp	
2	agriculture_vp_1...	agriculture_vp_1...	
3	forest_vp_1782	forest_vp_1782	
4	river_vp_1782	river_vp_1782	
5	roads_vp_1782	roads_vp_1782	

Codificación de la imagen

[✓ PNG] PNG8 JPEG TIFF SVG

Opciones (0 sistemas de referencia de coordenadas disponibles)

Tamaño de tesela:

Límite del objeto espacial para GetFeatureInfo: 10

- [ ] Usar leyenda WMS contextual

Nombre de la capa:

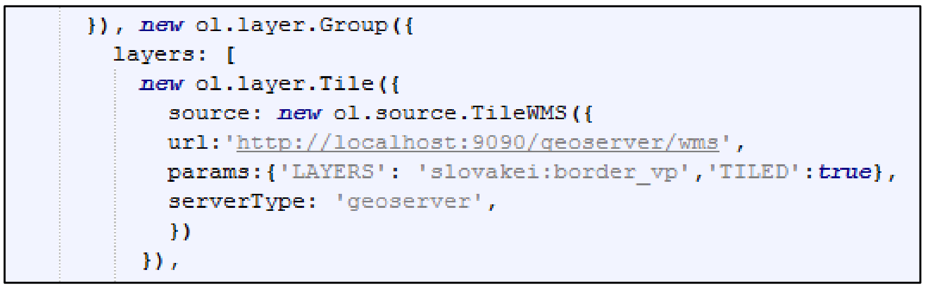
Seleccionar capa(s)

Figure 2.30: Testing MapServer with QGIS Desktop software

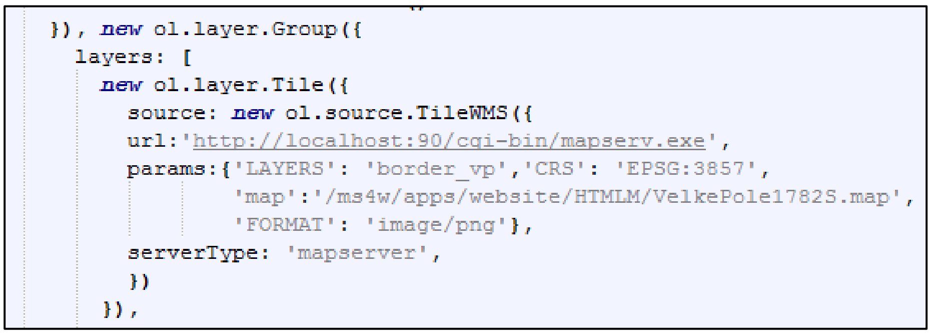
2.3.1. Building an HTML example for MapServer with the original database

The same HTML used for the Geoserver example was also used for doing this HTML, so it will be possible to compare the speed between these two servers with the project data.

The only difference between them is how the layers are called for each server. The figure on the right side shows the difference between working on Geoserver or MapServer with the layer 'border_vp'. These differences would be the same for all the layers.



```
}), new ol.layer.Group({
  layers: [
    new ol.layer.Tile({
      source: new ol.source.TileWMS({
      url:'http://localhost:9090/geoserver/wms',
      params:{'LAYERS': 'slovakei:border_vp','TILED':true},
      serverType: 'geoserver',
      })
    }),
```



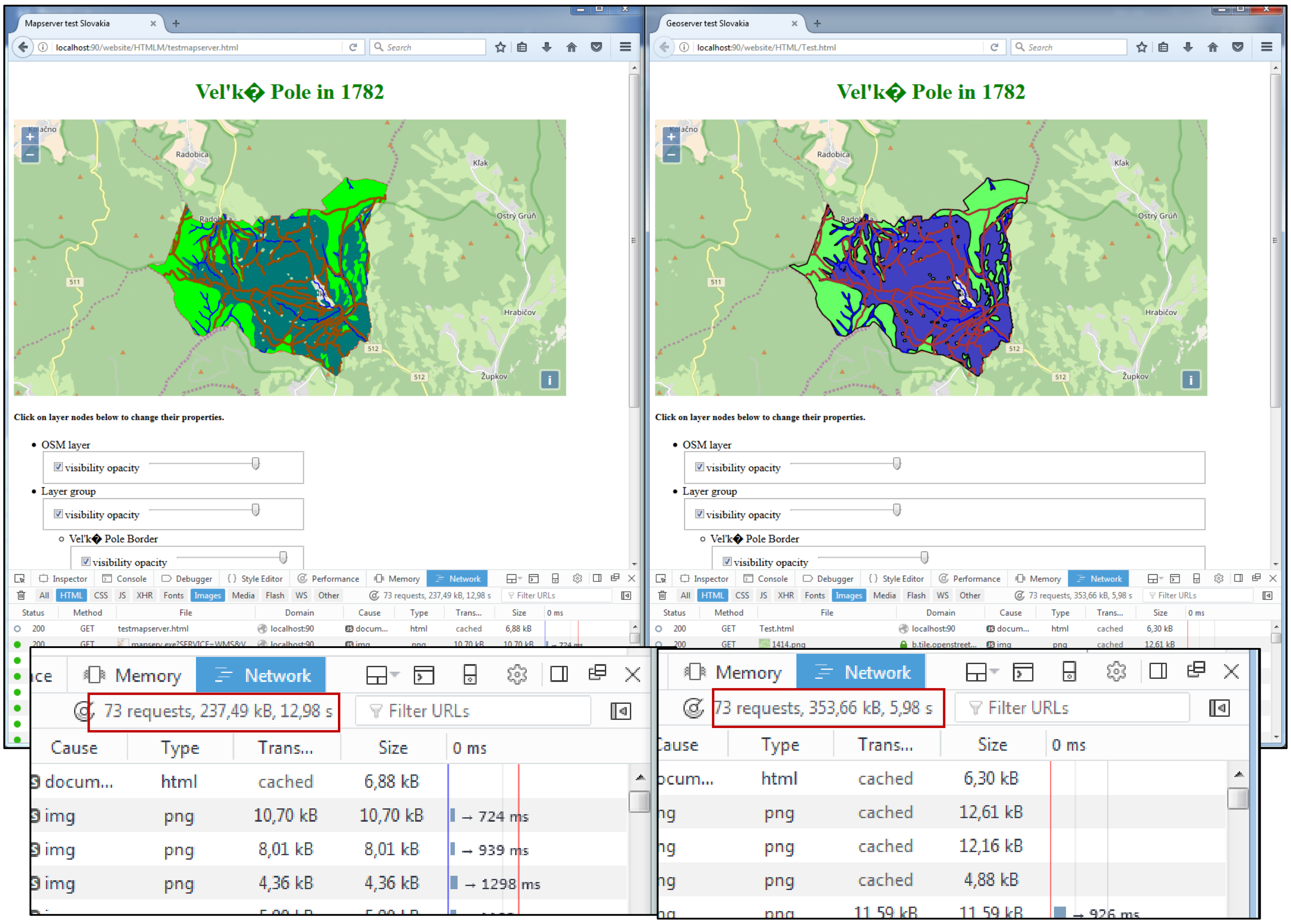
```
}), new ol.layer.Group({
  layers: [
    new ol.layer.Tile({
      source: new ol.source.TileWMS({
      url:'http://localhost:90/cgi-bin/mapserv.exe',
      params:{'LAYERS': 'border_vp','CRS': 'EPSG:3857',
              'map':'/ms4w/apps/website/HTMLM/VelkePole1782S.map',
              'FORMAT': 'image/png'},
      serverType: 'mapserver',
      })
    }),
```

Figure 2.31: Differences between Openlayers with Geoserver and Openlayers with MapServer HTML

2.4. Comparison between MapServer HTML and Geoserver HTML

Both of the HTML were tested on Firefox browser with its inspector tools and the results will be shown in the next figure.

This figure shows how Geoserver HTML is faster than MapServer HTML with the project data. Geoserver even loads more data from his server (353,66 kB in 5,98 seconds) than MapServer (237,49 kB in 12,98 seconds) in less time.



MapServer (left): 73 requests, 237,49 kB, 12,98 s

Cause	Type	Trans...	Size	0 ms
docum...	html	cached	6,88 kB	
img	png	10,70 kB	10,70 kB	→ 724 ms
img	png	8,01 kB	8,01 kB	→ 939 ms
img	png	4,36 kB	4,36 kB	→ 1298 ms

Geoserver (right): 73 requests, 353,66 kB, 5,98 s

Cause	Type	Trans...	Size	0 ms
docum...	html	cached	6,30 kB	
img	png	cached	12,61 kB	
img	png	cached	12,16 kB	
img	png	cached	4,88 kB	
img	png	11,59 kB	11,59 kB	→ 926 ms

Figure 2.32: Comparison between MapServer (left) and Geoserver (right)

With this test, it is proved that Geoserver works faster than MapServer with the Slovakia project data.

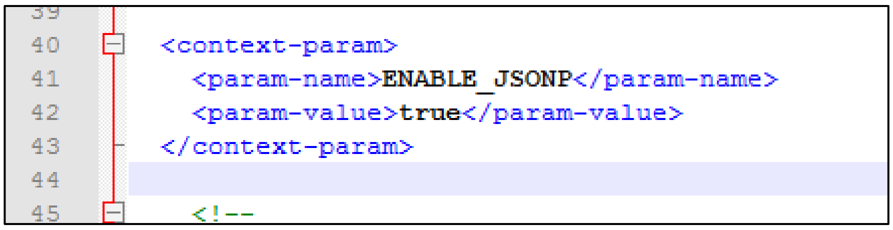
2.5. Handling WFS connections

2.5.1. Geoserver and Leaflet

Some lines of code of the web.xml have to be changed for using WFS with Geoserver. This file can be found here:

C:\Program Files (x86)\GeoServer 2.11.0\webapps\geoserver\WEB-INF\web.xml.

These lines are from the line 40 and this section is commented so, the "<!-- -->" symbol has to be deleted.



```
40    <context-param>
41      <param-name>ENABLE_JSONP</param-name>
42      <param-value>true</param-value>
43    </context-param>
44
45      <!--
```

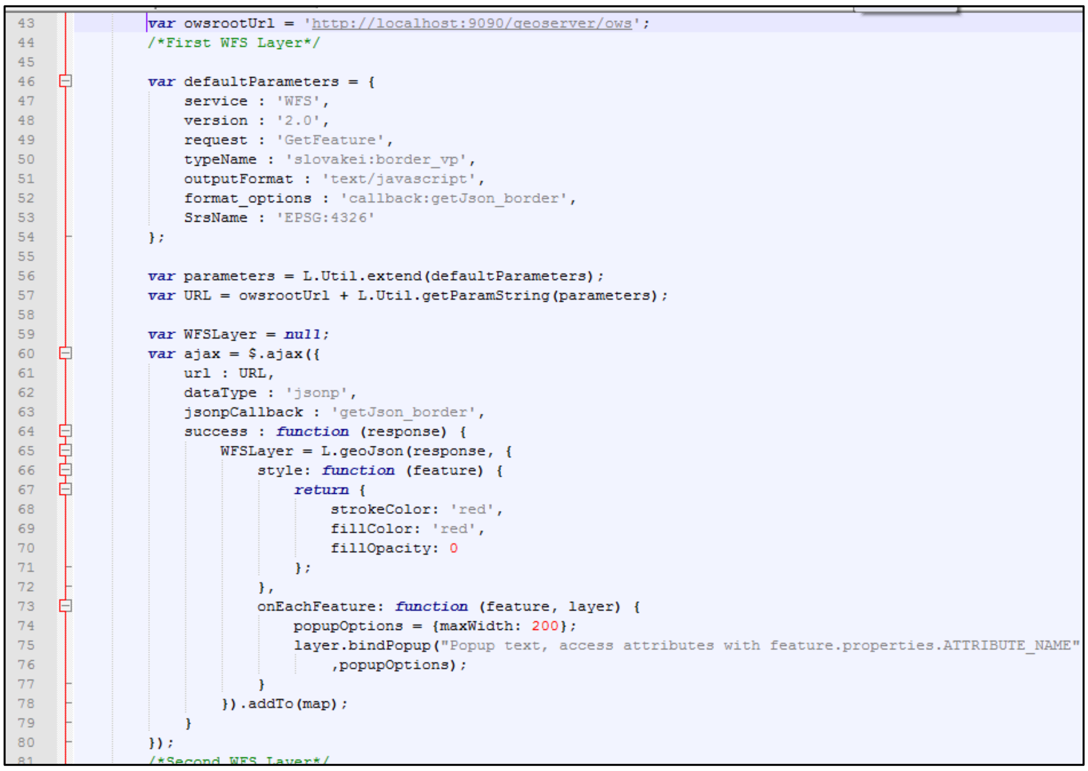
Figure 2.33: Enabling JSONP in Geoserver

There were problems publishing WFS with Openlayers about Cross-Origin Request (CORS). The reason is maybe because Geoserver was installed on the very first time with an .exe and not as Apache Tomcat package. With this way, the server is powered by Jetty server, and the website is working on Apache server.

Then, working with Openlayers and Geoserver it is recommended to install **Geoserver with Apache Tomcat server for solving this problem,** this change has been made for this project**.** Most of the CORS problems are solved installing Apache Tomcat, and there is not so much about CORS documentation for Jetty server, and this information doesn't fix the cross-origin problem.

Anyway, there is not any problem working with Leaflet JavaScript, and Leaflet is considered previously better than Openlayers for working on this project.

This is how the WFS layers are loaded. Each layer would have different format_options (line 52), jsonpCallback and of course, the style. In line 50 the data source from geoserver is required. All the layer will use the var owsrootURL as data source for whole WFS layers.

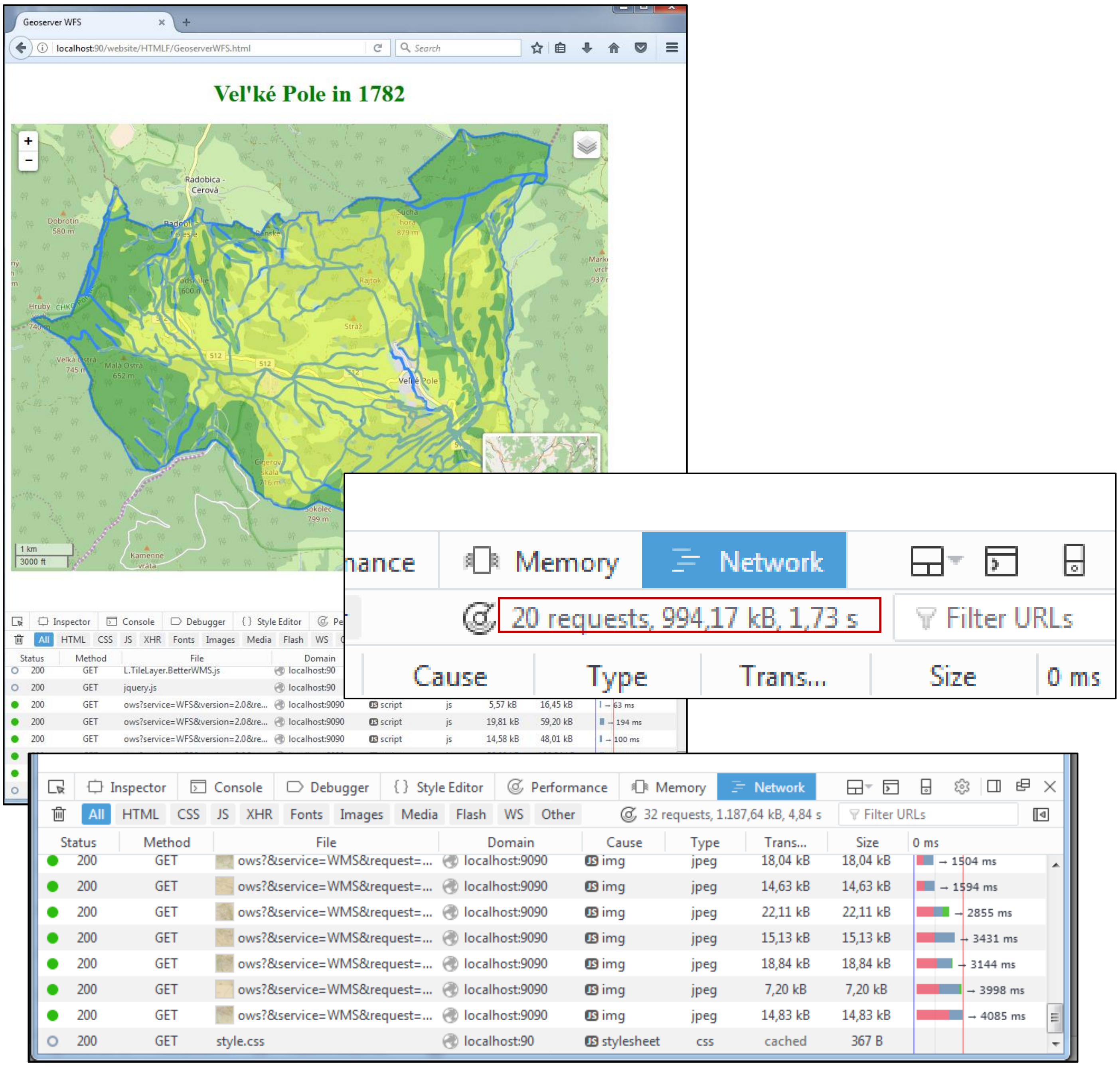


```
43  var owsrootUrl = 'http://localhost:9090/geoserver/ows';
44  /*First WFS Layer*/
45
46  var defaultParameters = {
47      service : 'WFS',
48      version : '2.0',
49      request : 'GetFeature',
50      typeName : 'slovakei:border_vp',
51      outputFormat : 'text/javascript',
52      format_options : 'callback:getJson_border',
53      SrsName : 'EPSG:4326'
54  };
55
56  var parameters = L.Util.extend(defaultParameters);
57  var URL = owsrootUrl + L.Util.getParamString(parameters);
58
59  var WFSLayer = null;
60  var ajax = $.ajax({
61      url : URL,
62      dataType : 'jsonp',
63      jsonpCallback : 'getJson_border',
64      success : function (response) {
65          WFSLayer = L.geoJson(response, {
66              style: function (feature) {
67                  return {
68                      strokeColor: 'red',
69                      fillColor: 'red',
70                      fillOpacity: 0
71                  };
72              },
73              onEachFeature: function (feature, layer) {
74                  popupOptions = {maxWidth: 200};
75                  layer.bindPopup("Popup text, access attributes with feature.properties.ATTRIBUTE_NAME"
76                      ,popupOptions);
77              }
78          }).addTo(map);
79      }
80  });
81  /*Second WFS Layer*/
```

Figure 2.34: Get WFS from Geoserver to Leaflet

Loading WFS layers into a HTML is much faster than loading WMS instead. In the test done with Geoserver and Openlayers, the time was 5,98 seconds to load all the features by WMS connections but, using Leaflet and Geoserver, the time goes down into 1,73 seconds, loading the same shapefiles but, from WFS server. In the next figure, its showed this fact.

In the same figure (down), one heavy cadastral map from 1850 (about 2.2 GB file) was loaded into the same HTML and the load time is even faster (4,84 seconds) than using only WMS on Openlayers (5,98 seconds) for loading features.



20 requests, 994,17 kB, 1,73 s

Status	Method	File	Domain	Cause	Type	Trans...	Size	0 ms
200	GET	L.TileLayer.BetterWMS.js	localhost:90					
200	GET	jquery.js	localhost:90					
200	GET	ows?service=WFS&version=2.0&re...	localhost:9090	script	js	5,57 kB	16,45 kB	→ 63 ms
200	GET	ows?service=WFS&version=2.0&re...	localhost:9090	script	js	19,81 kB	59,20 kB	→ 194 ms
200	GET	ows?service=WFS&version=2.0&re...	localhost:9090	script	js	14,58 kB	48,01 kB	→ 100 ms

32 requests, 1.187,64 kB, 4,84 s

Status	Method	File	Domain	Cause	Type	Trans...	Size	0 ms
200	GET	ows?&service=WMS&request=...	localhost:9090	img	jpeg	18,04 kB	18,04 kB	→ 1504 ms
200	GET	ows?&service=WMS&request=...	localhost:9090	img	jpeg	14,63 kB	14,63 kB	→ 1594 ms
200	GET	ows?&service=WMS&request=...	localhost:9090	img	jpeg	22,11 kB	22,11 kB	→ 2855 ms
200	GET	ows?&service=WMS&request=...	localhost:9090	img	jpeg	15,13 kB	15,13 kB	→ 3431 ms
200	GET	ows?&service=WMS&request=...	localhost:9090	img	jpeg	18,84 kB	18,84 kB	→ 3144 ms
200	GET	ows?&service=WMS&request=...	localhost:9090	img	jpeg	7,20 kB	7,20 kB	→ 3998 ms
200	GET	ows?&service=WMS&request=...	localhost:9090	img	jpeg	14,83 kB	14,83 kB	→ 4085 ms
200	GET	style.css	localhost:90	stylesheet	css	cached	367 B	

Figure 2.35: HTML loading WFS shapefiles and with a huge cadastral map by WMS

The results for the same task with Openlayers, maybe takes 1 second more but, at the end, **Openlayers will be used** (mixing WFS and WMS connections as the Leaflet example) because there are functions on the ESRI website which are not available on Leaflet. Especially the **layer control function**. The user only can select or unselect the layers on Leaflet library, but not switch the opacity, as Openlayers does.

3. Conversion original website HTML into Open Source Web HTML

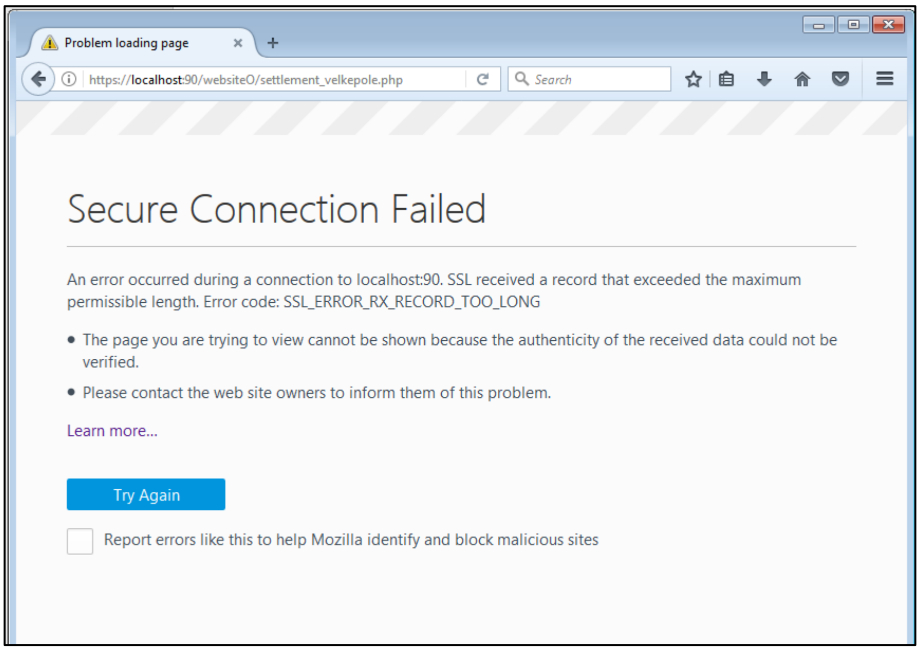
3.1. Troubleshooting

The website works with PHP files, which work through SSL certificates for safe connections.

Managing these files requires to set SSL certificates. That means changing PHP settings of Apache server by changing some lines of codes within some files. There are several tutorials for setting those files but, they are not effective. Handling PHP settings is difficult with Apache original setups.

The website folder has been added into the Apache server and when some PHP are loaded from there, the next message will be shown in the internet browser.

Despite " http://localhost "is written in the address bar, automatically it changes into " https://localhost " so, it is mandatory to set this SSL certificates for handle those files and adapt it to the open source web service.



Secure Connection Failed

An error occurred during a connection to localhost:90. SSL received a record that exceeded the maximum permissible length. Error code: SSL_ERROR_RX_RECORD_TOO_LONG

- The page you are trying to view cannot be shown because the authenticity of the received data could not be verified.
- Please contact the web site owners to inform them of this problem.

Learn more...

Try Again

Report errors like this to help Mozilla identify and block malicious sites

Figure 3.36: Troubles handling PHP files

After surfing through a lot of troubleshooting posts about this matter for original Apache installs, another way has been done: **Working with WAMPSERVER.**

3.2. Wampserver

Wampserver is the windows version for LAMP Linux architectures. LAMP is an acronym of the original four open-source components: Linux OS + Apache server + Mysql + PHP programming.

The good point of Wampserver is that, when the migration will be finished on this server, it will be easy to implement in the real LAMP hosting server rented by the University, or even on the University's server, because the structure is quite similar.

Inside Wampserver, Apache server, My SQL Database and PHP are included.

Furthermore, another advantage of working with this is: All the services are showed in a small Wampserver task windows, where you can find every file involved in Apache, PHP, and MySQL files settings like httpd.conf file, logs, connection settings, PHP settings,…and it's possible to access the projects inside Wampserver.

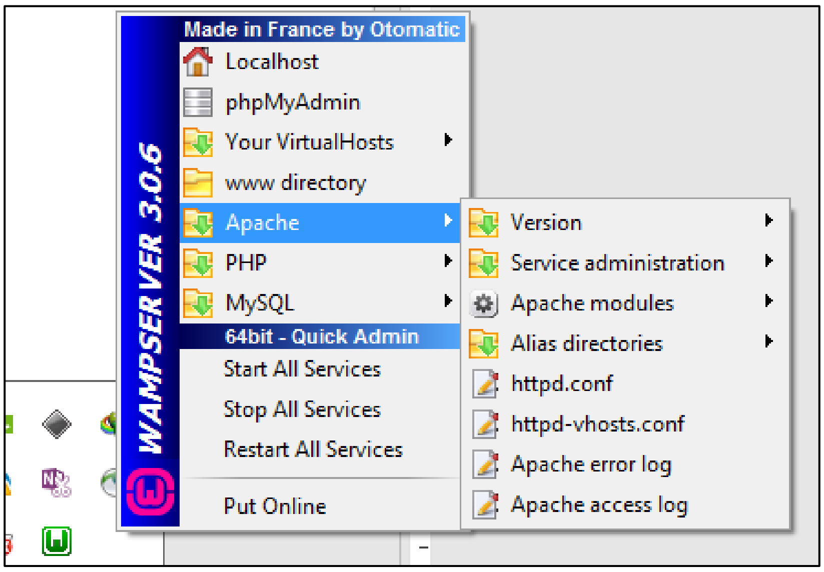


Figure 3.37: Wampserver

Anyway, the SSL certificates are not created by default then, they must be created.

For doing this, the next steps have to be done.

Step 1: Download & install WAMP (Assuming that WAMP is installed in C: directory)

Step 2: Download & install OpenSSL from this link for Windows 64 bits:

https://slproweb.com/download/Win64OpenSSL_Light-1_1_0f.exe

Step 3: Configure WAMP to use HTTP+SSL=HTTPS

1. Path to openssl.exe : C:\wamp64\bin\apache\apache2.4.23\bin\openssl.exe
2. Path to openssl.cnf : C:\wamp64\bin\apache\apache2.4.23\conf\openssl.cnf

Step 4: CMD in Path to C:\wamp64\bin\apache\apache2.4.23\bin\openssl.exe :

1. openssl genrsa -aes256 -out private.key 2048
2. openssl rsa -in private.key -out private.key
3. openssl req -new -x509 -nodes -sha1 -key private.key -out certificate.crt -days 36500 -config C:\wamp64\bin\apache\apache2.4.23\conf\openssl.cnf

Now copy the private.key & certificate.crt into folder C:\wamp64\bin\apache\apache2.4.23\conf\key

Step 5: Open httpd.conf on C:\wamp64\bin\apache\apache2.4.23\conf & uncomment following line code:

1. LoadModule ssl_module modules/mod_ssl.so.
2. Include conf/extra/httpd-ssl.conf.
3. LoadModule socache_shmcb_module modules/mod_socache_shmcb.so

Open php.ini (wamp64\bin\php\php5.5.12\php.ini) and uncomment the following line code:

1. extension=php_openssl.dll

Step 6: Modify C:\wamp64\bin\apache\apache 2.4.23\conf\extra\httpd-ssl.conf

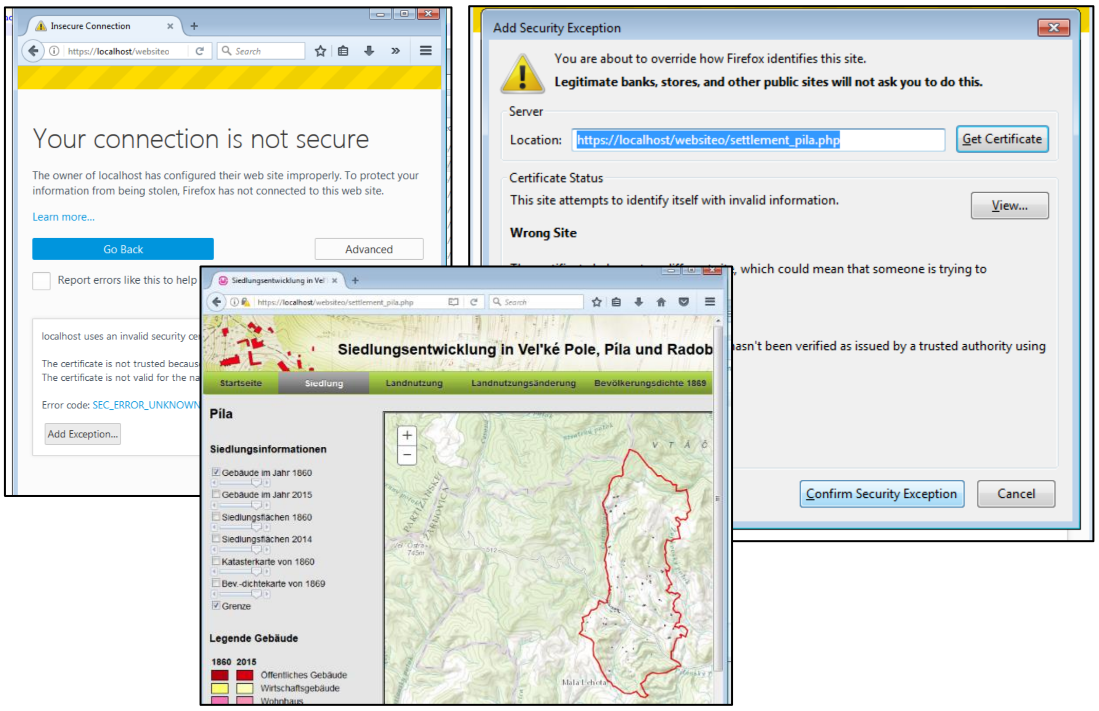
1. C:\wamp\bin\apache\apache x.x.x\conf\extra\httpd-ssl.conf.
2. <VirtualHost _default_:443> (Below this line check following parameters).

----

3. DocumentRoot "C:/wamp64/www".
4. ServerName localhost:443.
5. ServerAdmin admin@example.com.
6. ErrorLog "C:/wamp64/bin/apache/apache2.4.23/logs/ssl_error.log".
7. TransferLog "C:/wamp64/bin/apache/apache2.4.23/logs/ssl_access.log"
8. CustomLog "c:/wamp64/bin/apache/apache2.4.23/logs/ssl_request.log"
9. SSLCertificateFile "C:/wamp64/bin/apache/apache2.4.23/conf/key/certificate.crt"
10. SSLCertificateKeyFile "C:/wamp64/bin/apache/apache2.4.23/conf/key/private.key"

For checking if the changes were applied correctly, write in the common prompt httpd -t inside C:\wamp64\bin\apache\apache 2.4.23\conf\. Now SSL certificates are ready.

Here it is shown how those steps are made and how it works.



Insecure Connection

https://localhost/websiteo

Your connection is not secure

The owner of localhost has configured their web site improperly. To protect your information from being stolen, Firefox has not connected to this web site.

Learn more...

Go Back | Advanced

Report errors like this to help

localhost uses an invalid security ce

The certificate is not trusted becaus
The certificate is not valid for the na

Error code: SEC_ERROR_UNKNOWN

Add Exception...

Add Security Exception

You are about to override how Firefox identifies this site.

Legitimate banks, stores, and other public sites will not ask you to do this.

Server

Location: https://localhost/websiteo/settlement_pila.php | Get Certificate

Certificate Status

This site attempts to identify itself with invalid information. | View...

Wrong Site

, which could mean that someone is trying to

asn't been verified as issued by a trusted authority using

Confirm Security Exception | Cancel

Siedlungsentwicklung in Vel

https://localhost/websiteo/settlement_pila.php

Siedlungsentwicklung in Vel'ké Pole, Píla und Radob

Startseite | Siedlung | Landnutzung | Landnutzungsänderung | Bevölkerungsdichte 1869

Píla

Siedlungsinformationen

- [✓] Gebäude im Jahr 1860
- [ ] Gebäude im Jahr 2015
- [ ] Siedlungsflächen 1860
- [ ] Siedlungsflächen 2014
- [ ] Katasterkarte von 1860
- [ ] Bev.-dichtekarte von 1869
- [✓] Grenze

Legende Gebäude

1860	2015	
🟥	🟥	Öffentliches Gebäude
🟨	🟨	Wirtschaftsgebäude
🟪	🟪	Wohnhaus

Figure 3.38: Slovakia website working on Wampserver by HTTPS

3.3. Migration into Openlayers

At the end, Openlayers has been chosen as the javascript library that will be used for managing the development of the map viewer. The main reason is because Openlayers has more functionalities than Leaflet, and also has more years of development so, more programmers involved on this who can help you. All of the ESRI's website functions can be programmed into Openlayers, unlike Leaflet.

3.3.1. Handling Openlayers issues. HTML's Head

One php file has been modified for starting the migration. The php is settlement_velkepole.php. Once all of the ESRI functions have been replaced by Openlayers functions, the structure will be repeated for the rest of the php involved in the original website.

Most of the php inside this Project works similarly. A viewer is shown and the user can add or remove layers into the map on the left section. Just below, the user is able to search any existent building on the place depicted in the viewer. In this example, will be Vel'ké Pole village.

After clicking on the search button, a table from the SQL data base will be shown. The user is able to click on any building of this table and the function will show automatically the same building on the map viewer. This action of showing the selected buildings on the map will be programmed for running with Openlayers.

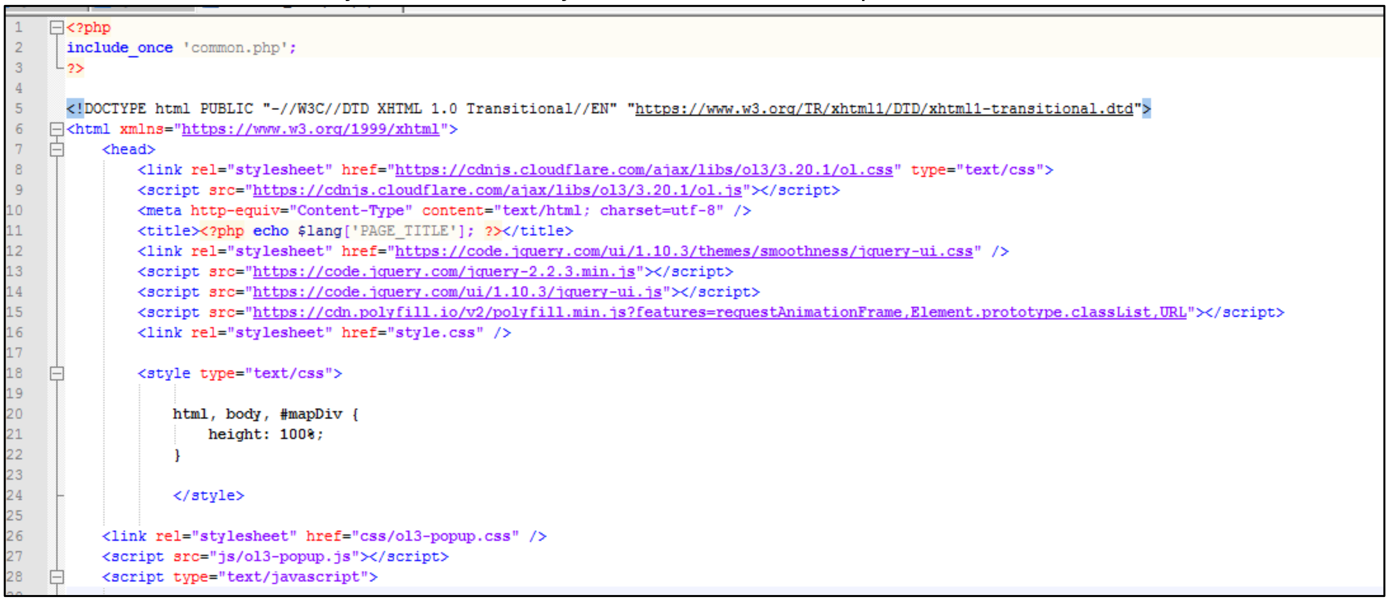
Within the website, there are quite many tags involved with other php. Those tags will be kept in the new version. One of these tags is for translating the keywords within the website content. These keywords are translated automatically when the website's language is changed by the users. Here is one example:

<title><?php echo $lang['PAGE_TITLE']; ?></title>

As one can see, the jquery function "$lang" is used in this line code, and when one language is chosen, "PAGE_TITLE" will be changed into that language. All the php files involved in all tags are within the website folder uploaded into the Apache server. Those files are pure php code, no ESRI functions are involved, so the same files will work on the new website version.

First of all, the head. Openlayers doesn't need as many scripts as ESRI does. There are two parts inside the head, one part is for these scripts, and the other part, is made of functions for loading and managing the data called from SQL data base. The code involved is shown in the next figures.

At the end of this figure, the call for the Openlayers popup library is done. This library shows the information for one object, when this object is clicked on the map.



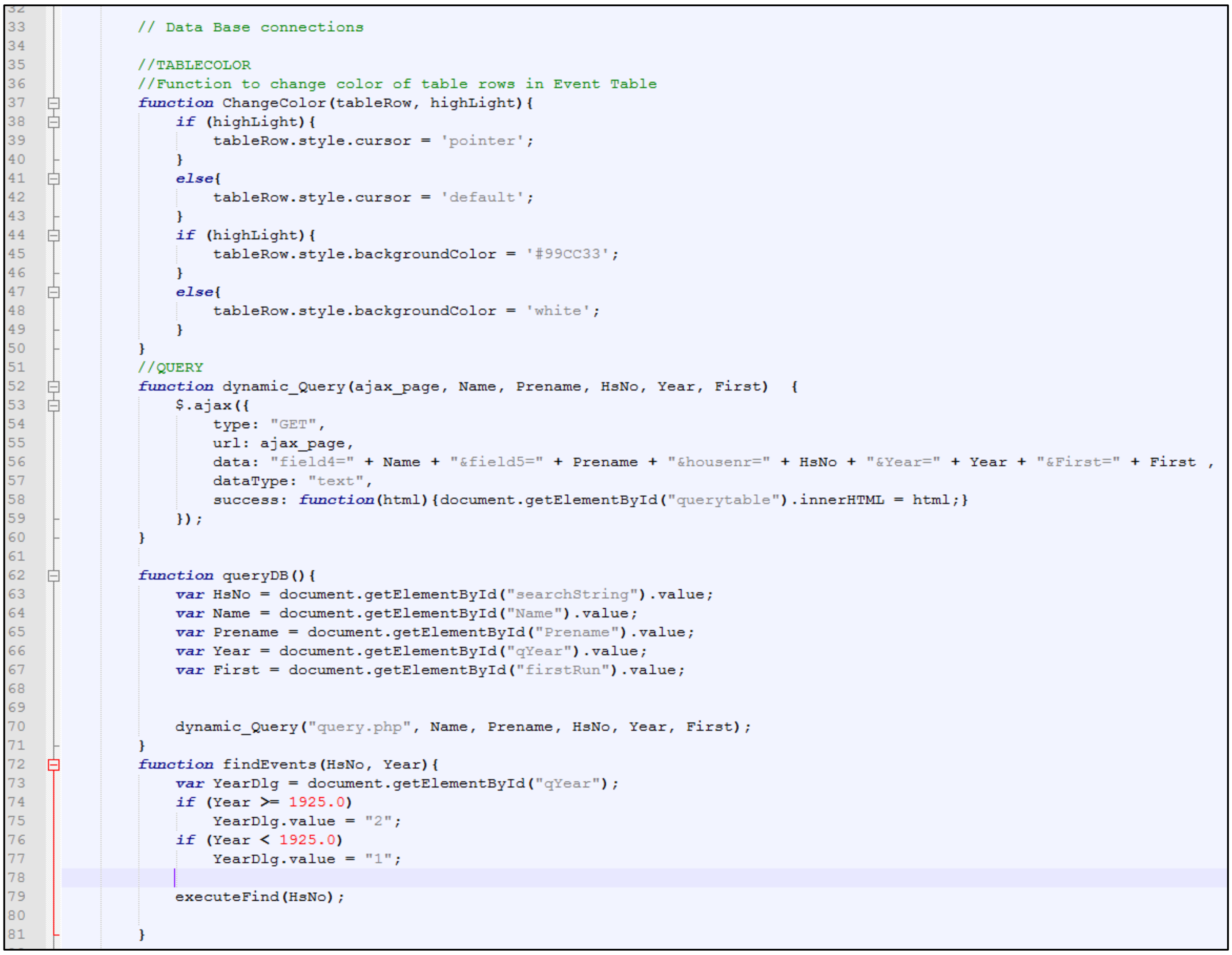
```
<?php
include_once 'common.php';
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "https://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="https://www.w3.org/1999/xhtml">
    <head>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/ol3/3.20.1/ol.css" type="text/css">
        <script src="https://cdnjs.cloudflare.com/ajax/libs/ol3/3.20.1/ol.js"></script>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title><?php echo $lang['PAGE_TITLE']; ?></title>
        <link rel="stylesheet" href="https://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
        <script src="https://code.jquery.com/jquery-2.2.3.min.js"></script>
        <script src="https://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
        <script src="https://cdn.polyfill.io/v2/polyfill.min.js?features=requestAnimationFrame,Element.prototype.classList,URL"></script>
        <link rel="stylesheet" href="style.css" />

        <style type="text/css">

            html, body, #mapDiv {
                height: 100%;
            }

            </style>

    <link rel="stylesheet" href="css/ol3-popup.css" />
    <script src="js/ol3-popup.js"></script>
    <script type="text/javascript">
```

Figure 3.39: Scripts involved in the head

The last script in the header has to be explained by parts, because this would be the most difficult issue along the php programming.

The first of these parts is based on the style for the query table (chaning colour when the mouse is over each result on the query table), calling variables from SQL data base by the query.php file (dynamic_Query and queryDB function) and findEvents function (from the year 1925, quite a lot of buildings changed his cadastral number). You can check the code involved in the next figure, which is quite similar than the original php.

On line 79 there is a function called “executeFind(HsNo)”, that will be explained later.



```
33          // Data Base connections
34
35          //TABLECOLOR
36          //Function to change color of table rows in Event Table
37          function ChangeColor(tableRow, highLight){
38              if (highLight){
39                  tableRow.style.cursor = 'pointer';
40              }
41              else{
42                  tableRow.style.cursor = 'default';
43              }
44              if (highLight){
45                  tableRow.style.backgroundColor = '#99CC33';
46              }
47              else{
48                  tableRow.style.backgroundColor = 'white';
49              }
50          }
51          //QUERY
52          function dynamic_Query(ajax_page, Name, Prename, HsNo, Year, First)  {
53              $.ajax({
54                  type: "GET",
55                  url: ajax_page,
56                  data: "field4=" + Name + "&field5=" + Prename + "&housenr=" + HsNo + "&Year=" + Year + "&First=" + First ,
57                  dataType: "text",
58                  success: function(html){document.getElementById("querytable").innerHTML = html;}
59              });
60          }
61
62          function queryDB(){
63              var HsNo = document.getElementById("searchString").value;
64              var Name = document.getElementById("Name").value;
65              var Prename = document.getElementById("Prename").value;
66              var Year = document.getElementById("qYear").value;
67              var First = document.getElementById("firstRun").value;
68
69
70              dynamic_Query("query.php", Name, Prename, HsNo, Year, First);
71          }
72          function findEvents(HsNo, Year){
73              var YearDlg = document.getElementById("qYear");
74              if (Year >= 1925.0)
75                  YearDlg.value = "2";
76              if (Year < 1925.0)
77                  YearDlg.value = "1";
78
79              executeFind(HsNo);
80
81          }
```

Figure 3.40: Calling and managing SQL queries

Before continuing explaining the php structure, other changes in one php file have to be explained, the query.php file.

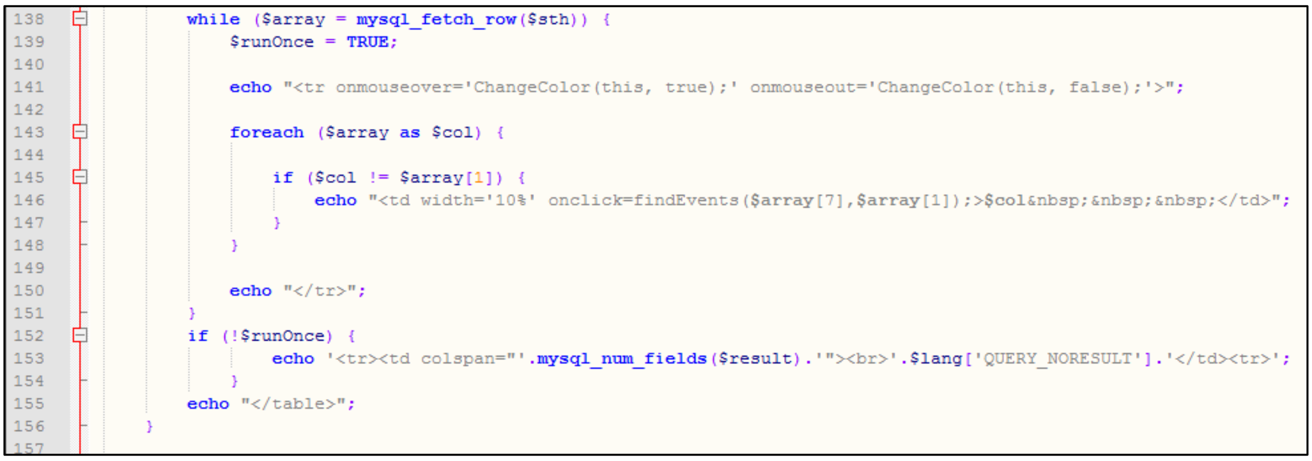
This file is programmed in MySQL php so, it’s different than HTML php, and one needs to be aware about this. The new version will be quite similar than the original, changing the username, password and database names defined in MySQL database. Also, the connection link has to be changed within the data base, because it is deprecated for the new php versions.

$link = @mysql_connect('localhost', 'root','', 'slowakei2014');

Adding the ‘at’ symbol, the deprecated version will be adapted for the new php versions. The rest of the query code will be kept.

From the line 138 of the following figure, there is the code involved on searching the variables “year” ($array [1]) and “house number” ($array [7]) and sending them into the findEvents function (line 146) with a click action (“onclick”). This function is on the last part of the last HTML figure. As it was explained before, this function works depending of the year, because on 1925 there were several changes on cadastral numbers. Now the executeFind function will be explained.

This function manages the values of “qYear” from findEvents function for loading new features into the map. These features have been added when the user clicks on one element from the query table.

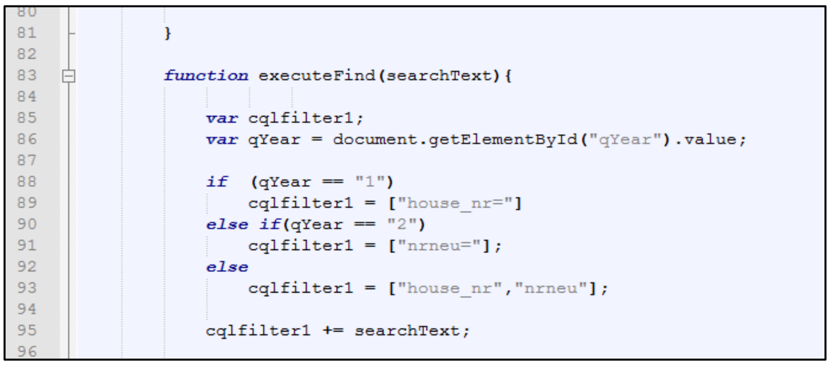


```
        while ($array = mysql_fetch_row($sth)) {
            $runOnce = TRUE;

            echo "<tr onmouseover='ChangeColor(this, true);' onmouseout='ChangeColor(this, false);'>";

            foreach ($array as $col) {

                if ($col != $array[1]) {
                    echo "<td width='10%' onclick=findEvents($array[7],$array[1]);>$col&nbsp;&nbsp;&nbsp;</td>";
                }
            }

            echo "</tr>";
        }
        if (!$runOnce) {
                echo '<tr><td colspan="'.mysql_num_fields($result).'"><br>'.$lang['QUERY_NORESULT'].'</td><tr>';
            }
        echo "</table>";
    }
```

Figure 3.41: Query code involved on HTML functions

The code has two parts. The first part gets the value depending on if is after o before 1925. If it's before 1925, house_nr will be returned as a string and if else, nrneu will be returned. These strings are added for doing a filter, CQLFILTER:

Cqlfilter1 contains the string house_nr or nrneu, with the number taken from MySQL, and afterwards, this variable will be used to look for the same building into the Geoserver database and later, add it into the map viewer.



```
        }

        function executeFind(searchText){

            var cqlfilter1;
            var qYear = document.getElementById("qYear").value;

            if  (qYear == "1")
                cqlfilter1 = ["house_nr="]
            else if(qYear == "2")
                cqlfilter1 = ["nrneu="];
            else
                cqlfilter1 = ["house_nr","nrneu"];

            cqlfilter1 += searchText;
```

Figure 3.42: executeFind function. Part 1

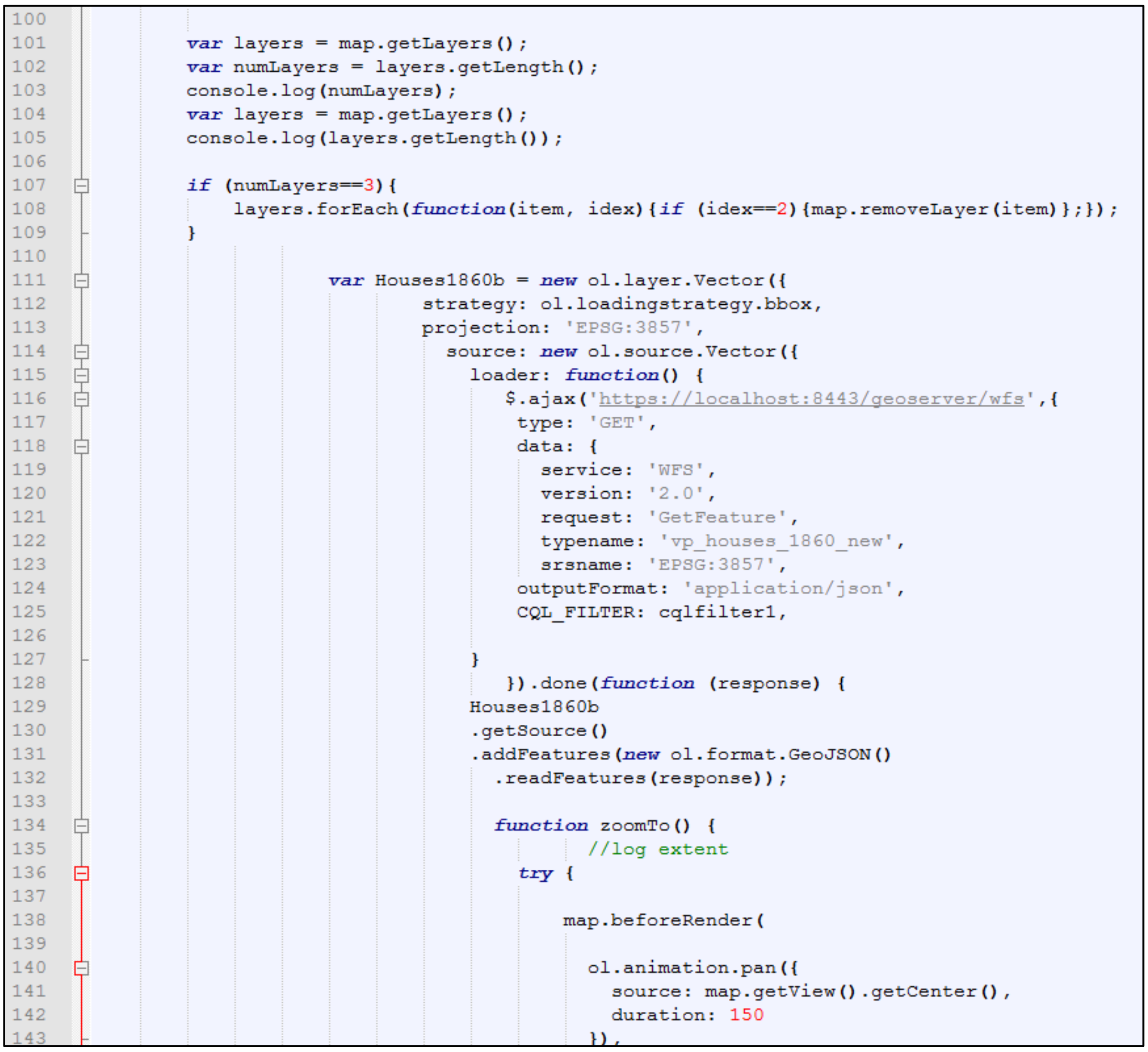
The second part is a bit tricky. It is about adding the same building from the Postgres database (by Geoserver) than the building is selected on the query table.

From the line 111 of the next figure, the WFS feature is called. On the line 125 the cqlfilter variable is shown. This is a filter which is applied to the response from the server. By this way, only one feature will be added to the map viewer and the variable which is defined on executeFind function.

When the user clicks on other buildings, they will be added to this layer. It will be a problem if the user selects two buildings which are together, because both of them will be highlighted so, it's difficult to know which is the last selection. From line 101 to 109, it is defined one 'if' loop for removing the last selected building then, always only one building will be shown in this layer.

From the line 134 another function is showed, "zoomTo". This function is for showing the building selected in the map viewer, this function is inside the ajax jquery code because if this function is outside the ajax jquery, it will be initialized before the WFS layer call so, "zoomTo" won't find any feature and will return an error. That happens because ajax query is asynchronous.

It is also important to put the 'strategy' (line 112) and 'projection' (line 113) options from the vector layer options at the first place. Normally these vector layer options are at the end of the vector layer options, but in this case, with these options at the end of the vector layer options, there is a problem with the ‘zoomTo’ function, because it’s loading the extension layer twice. Cutting these lines of code from the end and adding them to the beginning of this vector layer options will fix this problem.



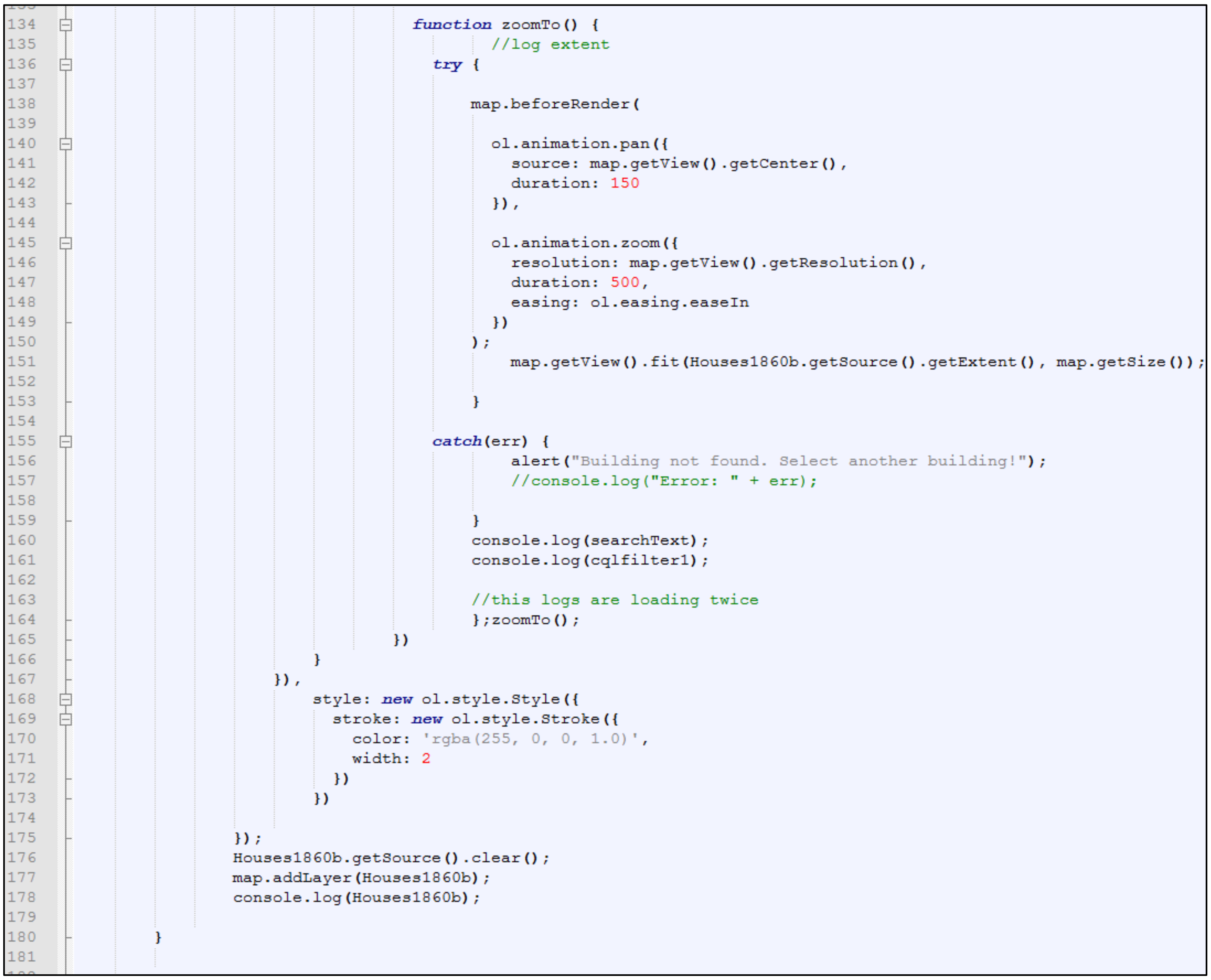
```

        var layers = map.getLayers();
        var numLayers = layers.getLength();
        console.log(numLayers);
        var layers = map.getLayers();
        console.log(layers.getLength());

        if (numLayers==3){
            layers.forEach(function(item, idex){if (idex==2){map.removeLayer(item)};});
        }

                    var Houses1860b = new ol.layer.Vector({
                            strategy: ol.loadingstrategy.bbox,
                            projection: 'EPSG:3857',
                              source: new ol.source.Vector({
                                loader: function() {
                                   $.ajax('https://localhost:8443/geoserver/wfs',{
                                    type: 'GET',
                                    data: {
                                      service: 'WFS',
                                      version: '2.0',
                                      request: 'GetFeature',
                                      typename: 'vp_houses_1860_new',
                                      srsname: 'EPSG:3857',
                                    outputFormat: 'application/json',
                                    CQL_FILTER: cqlfilter1,

                                }
                                   }).done(function (response) {
                                Houses1860b
                                .getSource()
                                .addFeatures(new ol.format.GeoJSON()
                                  .readFeatures(response));

                                  function zoomTo() {
                                          //log extent
                                    try {

                                        map.beforeRender(

                                          ol.animation.pan({
                                            source: map.getView().getCenter(),
                                            duration: 150
                                          }),
```

Figure 3.43: executeFind function. Part 2

This function does a pan and a zoom action into the feature’s extent. The complete code will be shown in the next figure.

The best statement that would be used for this case is the ‘try catch’ javascript’s statement, because it’s possible to define one alert when any building is not found, just like in the original website. Then, the style for this layer is defined with only a red colour. Then, the polygons will be shown with a red border.

At the end inside the head, there is another function, but it is the same than the original php so, it won’t be explained.



```
                          function zoomTo() {
                                  //log extent
                            try {

                                map.beforeRender(

                                  ol.animation.pan({
                                    source: map.getView().getCenter(),
                                    duration: 150
                                  }),

                                  ol.animation.zoom({
                                    resolution: map.getView().getResolution(),
                                    duration: 500,
                                    easing: ol.easing.easeIn
                                  })
                                );
                                    map.getView().fit(Houses1860b.getSource().getExtent(), map.getSize());

                                }

                            catch(err) {
                                    alert("Building not found. Select another building!");
                                    //console.log("Error: " + err);

                                }
                                console.log(searchText);
                                console.log(cqlfilter1);

                                //this logs are loading twice
                                };zoomTo();
                        })
                }
            }),
                style: new ol.style.Style({
                  stroke: new ol.style.Stroke({
                    color: 'rgba(255, 0, 0, 1.0)',
                    width: 2
                  })
                })

        });
        Houses1860b.getSource().clear();
        map.addLayer(Houses1860b);
        console.log(Houses1860b);

}
```

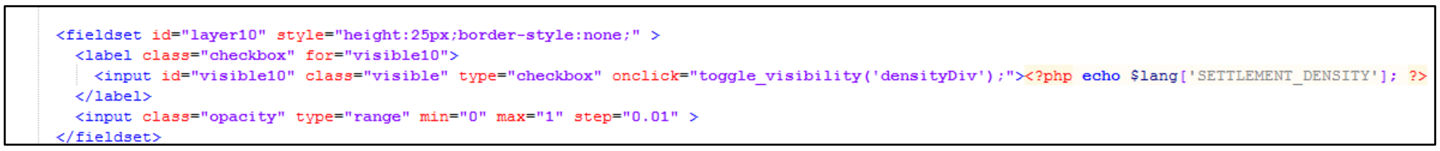
Figure 3.44: executeFind function. Part 3

3.3.2. Handling Openlayers issues. HTML's Body

The first part of the body includes the original header.php and menu.php files and the Openlayer's layer control, which is explained on Geoserver example, page 29, section 2.2.1.

But for the new version it will be different. There will be the tags for the name of each layer contained into this layer group control. Those tags are referring to language.php file, which translates those names into the selected language.

In the last figure, the fieldset definition for one layer of Openlayers layer control is shown. The same structure will be done for all layers within the layer control. There is one 'onclick' function, related to hiding or showing the legend depending on if the layer is being shown or not, and the php tag for translating the name, that's showed on layer control section.



```
<fieldset id="layer10" style="height:25px;border-style:none;" >
  <label class="checkbox" for="visible10">
    <input id="visible10" class="visible" type="checkbox" onclick="toggle_visibility('densityDiv');"><?php echo $lang['SETTLEMENT_DENSITY']; ?>
  </label>
  <input class="opacity" type="range" min="0" max="1" step="0.01" >
</fieldset>
```

Figure 3.45: One example of filedset from layer control definition

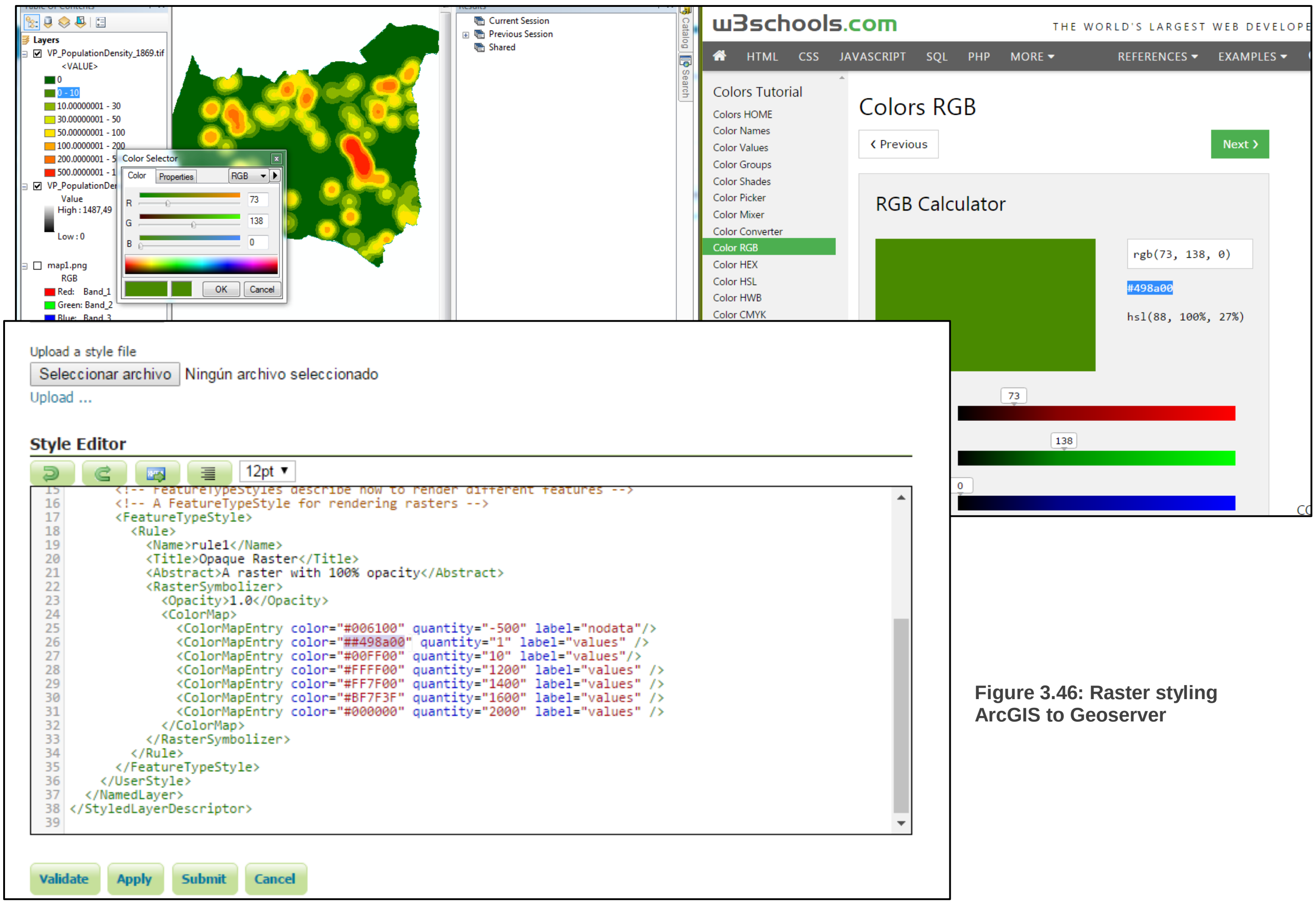
The legend tags section will be kept and then, the map script is defined. There are WMS calls to the server. Those layers have symbology that is difficult to set up as WFS Geojson callbacks, normally raster files or layers with few different categories within.

In those calls, the symbology will be defined on the Geoserver symbology editor. Here is an example of a density map raster file.

In raster styling, the old ArcServer can load the LYR file from ArcMap and apply the raster style within it but, Geoserver doesn't read the type of this file. SLD file is a styling xml file for Geoserver features, and SLD also supports raster files. the same LYR features have to be applied to this SLD file. For achieving this, it is necessary to view the accurate grade colour styling on ArcMap, and bring it to Geoserver. In the next figure, it is shown how this step was done.

For each colour on the ArcGIS table of content it is possible to check its RGB colour composition which is being depicted on the raster. RGB Calculator from w3schools.com can convert this RGB composition into a RGB composition for XML files. After converting this value, it has to be changed in the SLD editor from Geoserver.

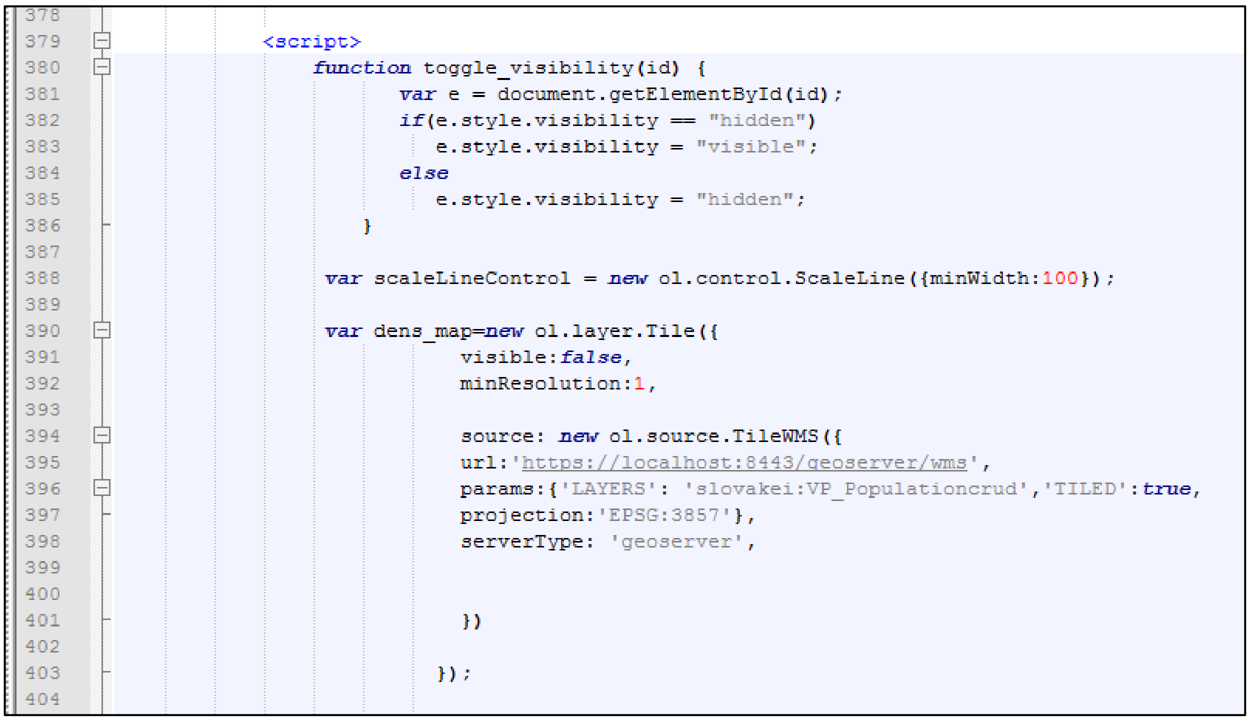
The following figure shows how the symbology style editor of Geoserver works, but this structure is only for raster files. Symbology for buildings has also been made, when those layers have more than one category. On GeoJson WFS connections, it is not easy to define one style for each category.



```
    <!-- FeatureTypeStyles describe how to render different features -->
    <!-- A FeatureTypeStyle for rendering rasters -->
    <FeatureTypeStyle>
      <Rule>
        <Name>rule1</Name>
        <Title>Opaque Raster</Title>
        <Abstract>A raster with 100% opacity</Abstract>
        <RasterSymbolizer>
          <Opacity>1.0</Opacity>
          <ColorMap>
            <ColorMapEntry color="#006100" quantity="-500" label="nodata"/>
            <ColorMapEntry color="##498a00" quantity="1" label="values" />
            <ColorMapEntry color="#00FF00" quantity="10" label="values"/>
            <ColorMapEntry color="#FFFF00" quantity="1200" label="values" />
            <ColorMapEntry color="#FF7F00" quantity="1400" label="values" />
            <ColorMapEntry color="#BF7F3F" quantity="1600" label="values" />
            <ColorMapEntry color="#000000" quantity="2000" label="values" />
          </ColorMap>
        </RasterSymbolizer>
      </Rule>
    </FeatureTypeStyle>
  </UserStyle>
 </NamedLayer>
</StyledLayerDescriptor>
```

Figure 3.46: Raster styling ArcGIS to Geoserver

Regarding the script's code structure:

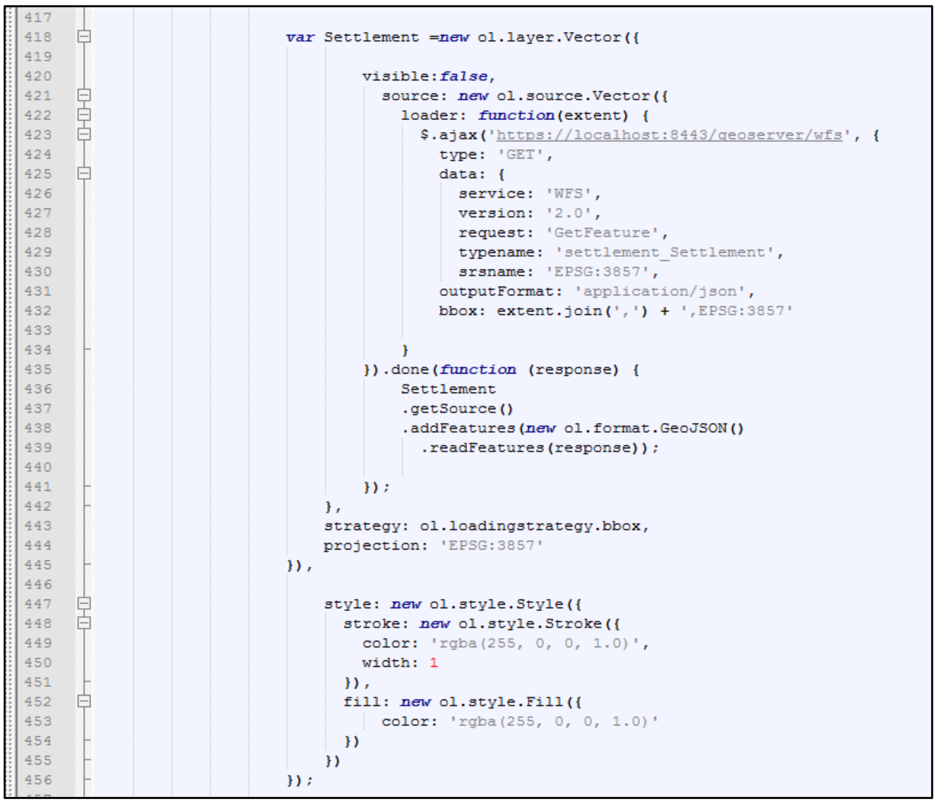


```
            <script>
                function toggle_visibility(id) {
                    var e = document.getElementById(id);
                    if(e.style.visibility == "hidden")
                      e.style.visibility = "visible";
                    else
                      e.style.visibility = "hidden";
                }

                 var scaleLineControl = new ol.control.ScaleLine({minWidth:100});

                 var dens_map=new ol.layer.Tile({
                          visible:false,
                          minResolution:1,

                          source: new ol.source.TileWMS({
                          url:'https://localhost:8443/geoserver/wms',
                          params:{'LAYERS': 'slovakei:VP_Populationcrud','TILED':true,
                          projection:'EPSG:3857'},
                          serverType: 'geoserver',


                          })

                        });
```

Figure 3.47: WMS call example from Velké Pole Settlement website

The map script will be explained with an example of one single layer, because the code is long and it's not necessary to show it at all. In the last figure, there is an example for WMS call, this layer is about the population density, whose style definition was explained before. The first function shown is for hiding or showing the density map legend depending if its layer is shown or not.

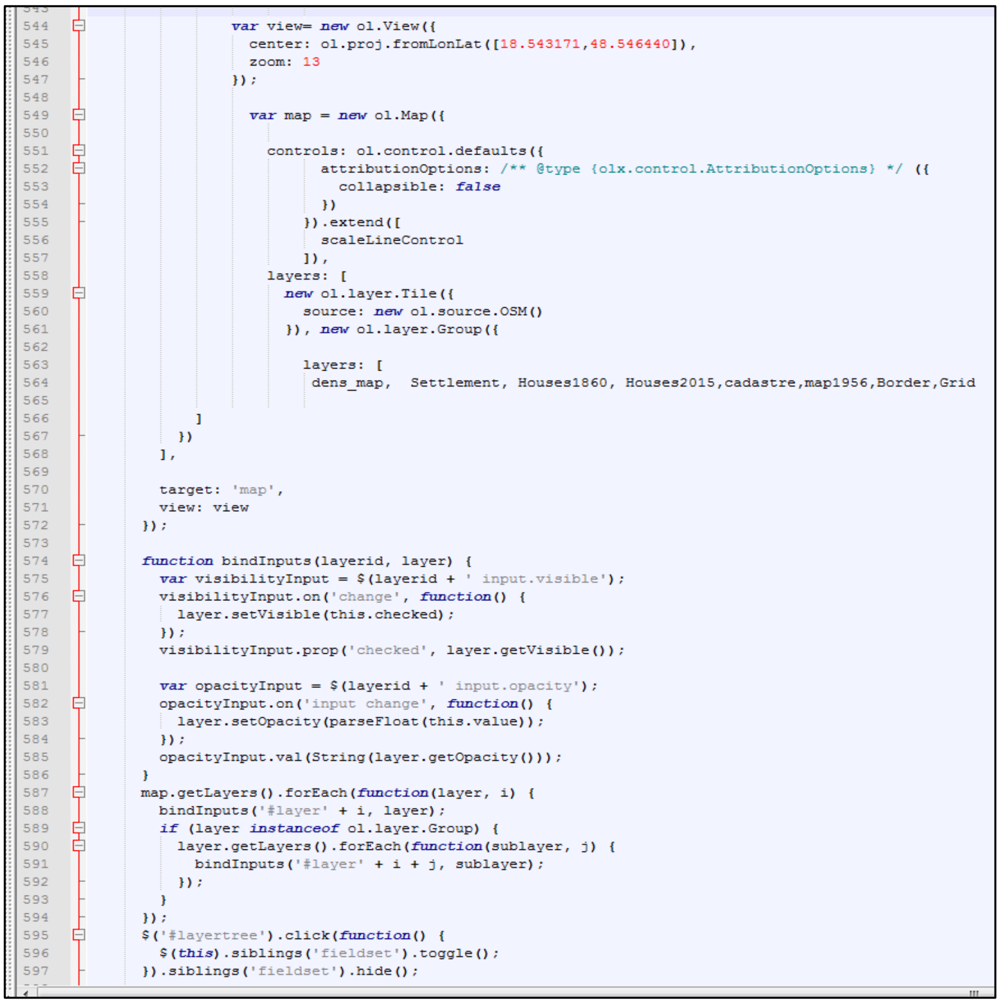
This WFS connection shows how a WFS call is defined, in this case for the settlement layer. At the end, the style is defined. Here, this layer only has one category. WFS connections draws the layers depending on the stroke (polygon's border) and fill (which colour this object is filled with), in this case is red for stroke and fill options.



```
var Settlement =new ol.layer.Vector({

        visible:false,
          source: new ol.source.Vector({
            loader: function(extent) {
              $.ajax('https://localhost:8443/geoserver/wfs', {
                type: 'GET',
                data: {
                  service: 'WFS',
                  version: '2.0',
                  request: 'GetFeature',
                  typename: 'settlement_Settlement',
                  srsname: 'EPSG:3857',
                outputFormat: 'application/json',
                bbox: extent.join(',') + ',EPSG:3857'

            }
        }).done(function (response) {
            Settlement
            .getSource()
            .addFeatures(new ol.format.GeoJSON()
              .readFeatures(response));

        });
    },
    strategy: ol.loadingstrategy.bbox,
    projection: 'EPSG:3857'
}),

    style: new ol.style.Style({
      stroke: new ol.style.Stroke({
        color: 'rgba(255, 0, 0, 1.0)',
        width: 1
      }),
      fill: new ol.style.Fill({
          color: 'rgba(255, 0, 0, 1.0)'
      })
    })
});
```

Figure 3.48: WFS connection example

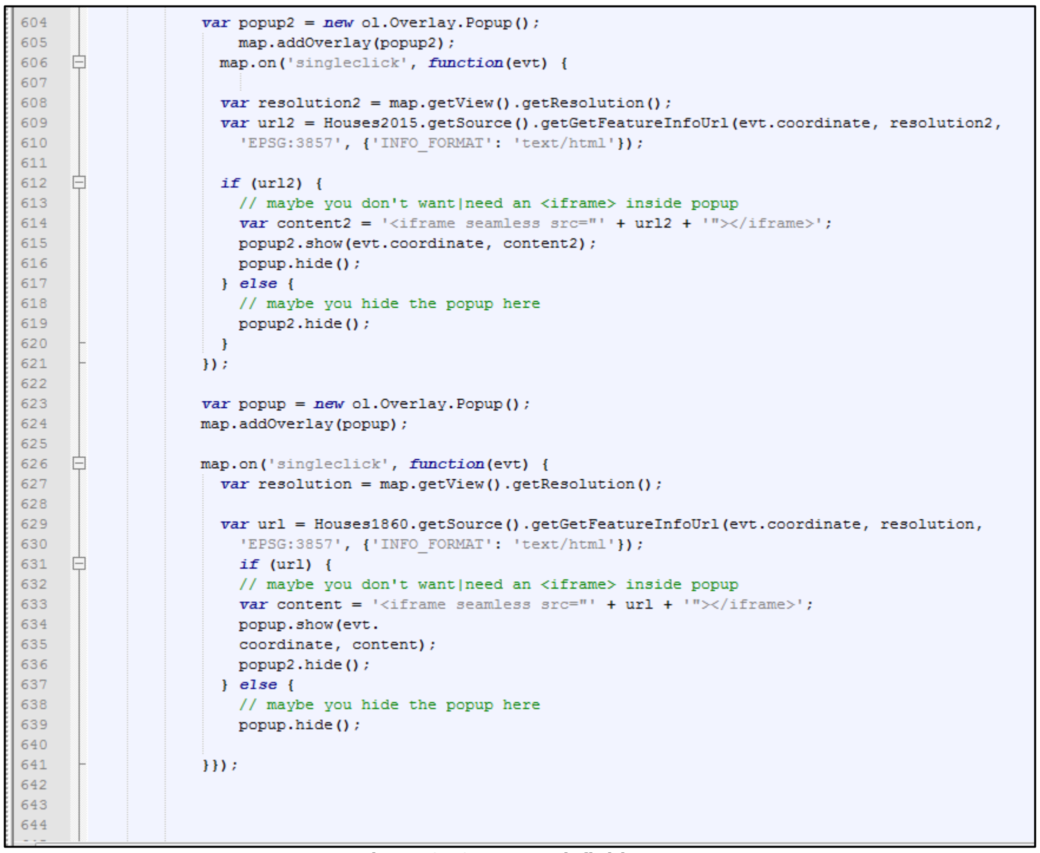
On line 564, the layer order is stablished for saving it into the Openlayers Layer Control function, this function is shown at the end of the same figure.



```
544                 var view= new ol.View({
545                   center: ol.proj.fromLonLat([18.543171,48.546440]),
546                   zoom: 13
547                 });
548
549                   var map = new ol.Map({
550
551                     controls: ol.control.defaults({
552                           attributionOptions: /** @type {olx.control.AttributionOptions} */ ({
553                             collapsible: false
554                           })
555                         }).extend([
556                           scaleLineControl
557                         ]),
558                     layers: [
559                       new ol.layer.Tile({
560                         source: new ol.source.OSM()
561                       }), new ol.layer.Group({
562
563                         layers: [
564                          dens_map,  Settlement, Houses1860, Houses2015,cadastre,map1956,Border,Grid
565
566               ]
567             })
568           ],
569
570           target: 'map',
571           view: view
572         });
573
574         function bindInputs(layerid, layer) {
575           var visibilityInput = $(layerid + ' input.visible');
576           visibilityInput.on('change', function() {
577             layer.setVisible(this.checked);
578           });
579           visibilityInput.prop('checked', layer.getVisible());
580
581           var opacityInput = $(layerid + ' input.opacity');
582           opacityInput.on('input change', function() {
583             layer.setOpacity(parseFloat(this.value));
584           });
585           opacityInput.val(String(layer.getOpacity()));
586         }
587         map.getLayers().forEach(function(layer, i) {
588           bindInputs('#layer' + i, layer);
589           if (layer instanceof ol.layer.Group) {
590             layer.getLayers().forEach(function(sublayer, j) {
591               bindInputs('#layer' + i + j, sublayer);
592             });
593           }
594         });
595         $('#layertree').click(function() {
596           $(this).siblings('fieldset').toggle();
597         }).siblings('fieldset').hide();
```

Figure 3.49: Map definition

Here, the popup is defined for the buildings so, when you click on one building, one popup is opened with its information from Postgres data base.



```
          var popup2 = new ol.Overlay.Popup();
              map.addOverlay(popup2);
            map.on('singleclick', function(evt) {

              var resolution2 = map.getView().getResolution();
              var url2 = Houses2015.getSource().getGetFeatureInfoUrl(evt.coordinate, resolution2,
                'EPSG:3857', {'INFO_FORMAT': 'text/html'});

              if (url2) {
                // maybe you don't want|need an <iframe> inside popup
                var content2 = '<iframe seamless src="' + url2 + '"></iframe>';
                popup2.show(evt.coordinate, content2);
                popup.hide();
              } else {
                // maybe you hide the popup here
                popup2.hide();
              }
          });

          var popup = new ol.Overlay.Popup();
          map.addOverlay(popup);

          map.on('singleclick', function(evt) {
            var resolution = map.getView().getResolution();

            var url = Houses1860.getSource().getGetFeatureInfoUrl(evt.coordinate, resolution,
              'EPSG:3857', {'INFO_FORMAT': 'text/html'});
                if (url)  {
                // maybe you don't want|need an <iframe> inside popup
                var content = '<iframe seamless src="' + url + '"></iframe>';
                popup.show(evt.
                coordinate, content);
                popup2.hide();
              } else {
                // maybe you hide the popup here
                popup.hide();

          }});
```

Figure 3.50: Popup definition

4. Uploading the website into HsKA server

When the project is finished, the last step is uploading all the files to the HsKA server.

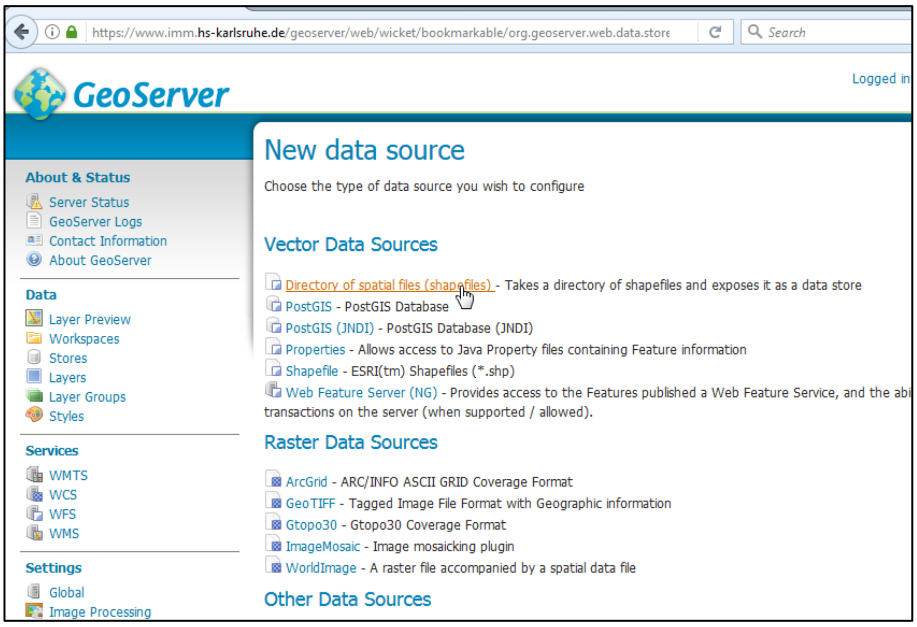
All the url from all php files must be changed, because until now, they are being called from localhost server to get all the layers uploaded into Geoserver.

A couple of HsKA network folders can be accessed now for uploading the files:

\\imm-gisevent\data (where the Geoserver files will be added) and \\imm-webserver\slowakei (where the website files will be located)

There are a couple of problems with Geodata, as Geoserver on HsKA server doesn't recognize the workspace of localhost Geoserver. That means that all of the project layers must be uploaded again to HsKA Geoserver. Besides, there is no Postgres database on HsKA server, so the shapefiles will be loaded directly from HsKA server folder.

Then, the shapefiles will be loaded as 'Directory of spatial files' on Data/ Stores/ New Store on Geoserver, instead a PostGIS package.



New data source

Choose the type of data source you wish to configure

Vector Data Sources

- Directory of spatial files (shapefiles) - Takes a directory of shapefiles and exposes it as a data store
- PostGIS - PostGIS Database
- PostGIS (JNDI) - PostGIS Database (JNDI)
- Properties - Allows access to Java Property files containing Feature information
- Shapefile - ESRI(tm) Shapefiles (*.shp)
- Web Feature Server (NG) - Provides access to the Features published a Web Feature Service, and the abi transactions on the server (when supported / allowed).

Raster Data Sources

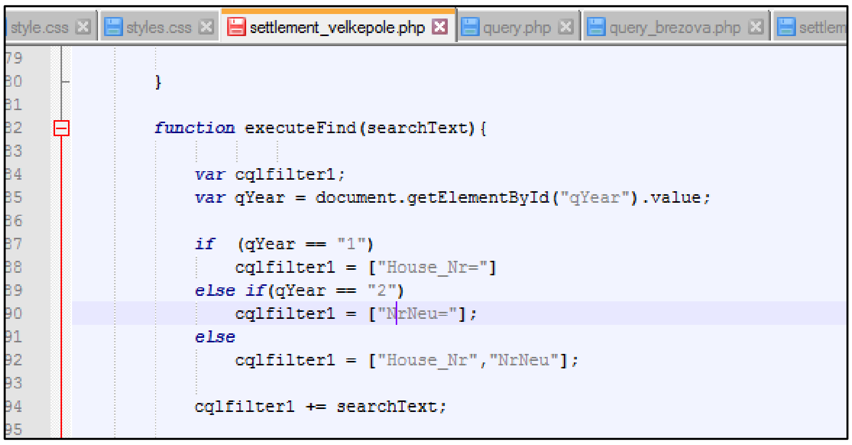
- ArcGrid - ARC/INFO ASCII GRID Coverage Format
- GeoTIFF - Tagged Image File Format with Geographic information
- Gtopo30 - Gtopo30 Coverage Format
- ImageMosaic - Image mosaicking plugin
- WorldImage - A raster file accompanied by a spatial data file

Other Data Sources

Figure 4.51: Loading the shapefiles into HsKA Geoserver

The website on localhost works with Postgres. When a layer is loaded into Postgres database, it changes all the field names of the layers into minor letters. Before uploading the website to the server, those layers are called by minor letters in the php file. The new website on HsKA server doesn't works with Postgres thus, it's necessary to change those field names from minor letter to capital letters, like the fields of the original layers has.

Especially the CQLFILTER function, because is searching for fields with ‘house_nr’ and must be changed to ‘House_Nr’ as the original field’s layer name.



```
        }

        function executeFind(searchText){

            var cqlfilter1;
            var qYear = document.getElementById("qYear").value;

            if  (qYear == "1")
                cqlfilter1 = ["House_Nr="]
            else if(qYear == "2")
                cqlfilter1 = ["NrNeu="];
            else
                cqlfilter1 = ["House_Nr","NrNeu"];

            cqlfilter1 += searchText;
```

Figure 4.52: Changing some functions. Changing Postgres Data base issues

The SLD file has to be changed because all the field names referred from the layers have also capital letters.

After uploading all the layers to Geoserver and checking everything is alright, it’s possible to compare this new website with the ESRI website, both on HsKA server. That comparison will be analysed on the next point.

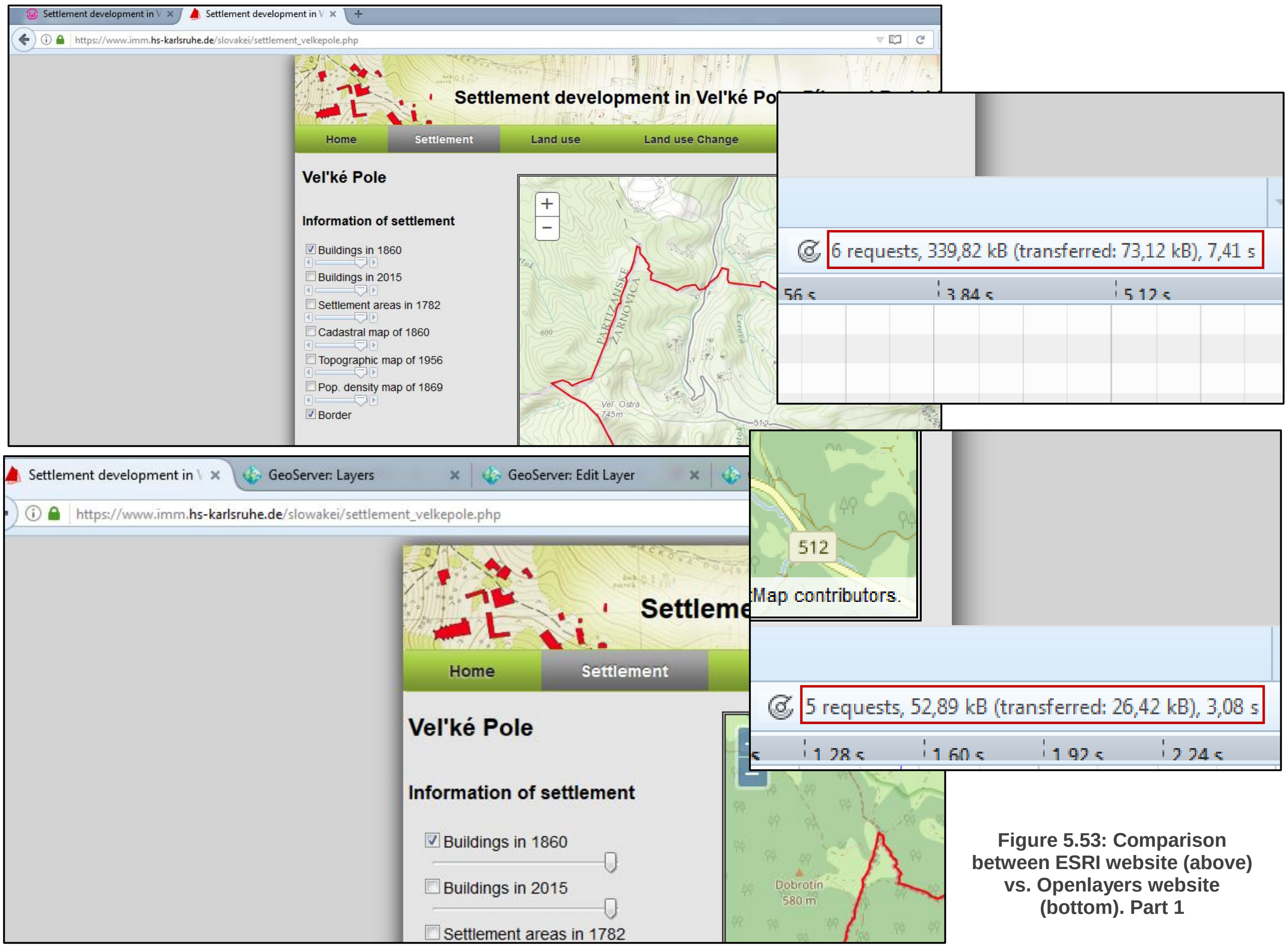
5. Performance Geoserver and Openlayers vs. ESRI

For testing the performance between the website with ESRI software against with Open Source software, a comparison with Firefox inspector has been made. On the network section, the time of web loading is showed on the right side of the Firefox' inspector. The test has been made outside the HsKA network.

These captures show how the new website works as good as the old website. Actually, the new website with Geoserver and Openlayers works better than the ESRI website because, ESRI website is calling for more requests and for more data than Openlayers. In the following captures, a couple of examples will be shown.

Both websites are from Hochschule Karlsruhe server but, the first image is with the old ESRI software and the captures below are with the new Open Source software. With Open Source Software, this website is more than twice faster, 3,08 seconds against 7,41 seconds. Also, the data transferred from the server to the HTML is also shown, and on ESRI website is getting 6 requests and 339,82 kB, meanwhile Openlayers is getting 5 requests and 52,89 kB instead. The difference is because of the style package from ESRI. These style files are larger than the Openlayers style files.

The php tested in both cases is about Vel'ké Pole settlement information. This php shows only two layers In Openlayers version, which are being loaded by WFS connections, instead of the ESRI version which loads only WMS connections. Maybe this php loads also the functions involved on the query table but, these functions are not being used yet on this test.


https://www.imm.hs-karlsruhe.de/slovakei/settlement_velkepole.php

6 requests, 339,82 kB (transferred: 73,12 kB), 7,41 s

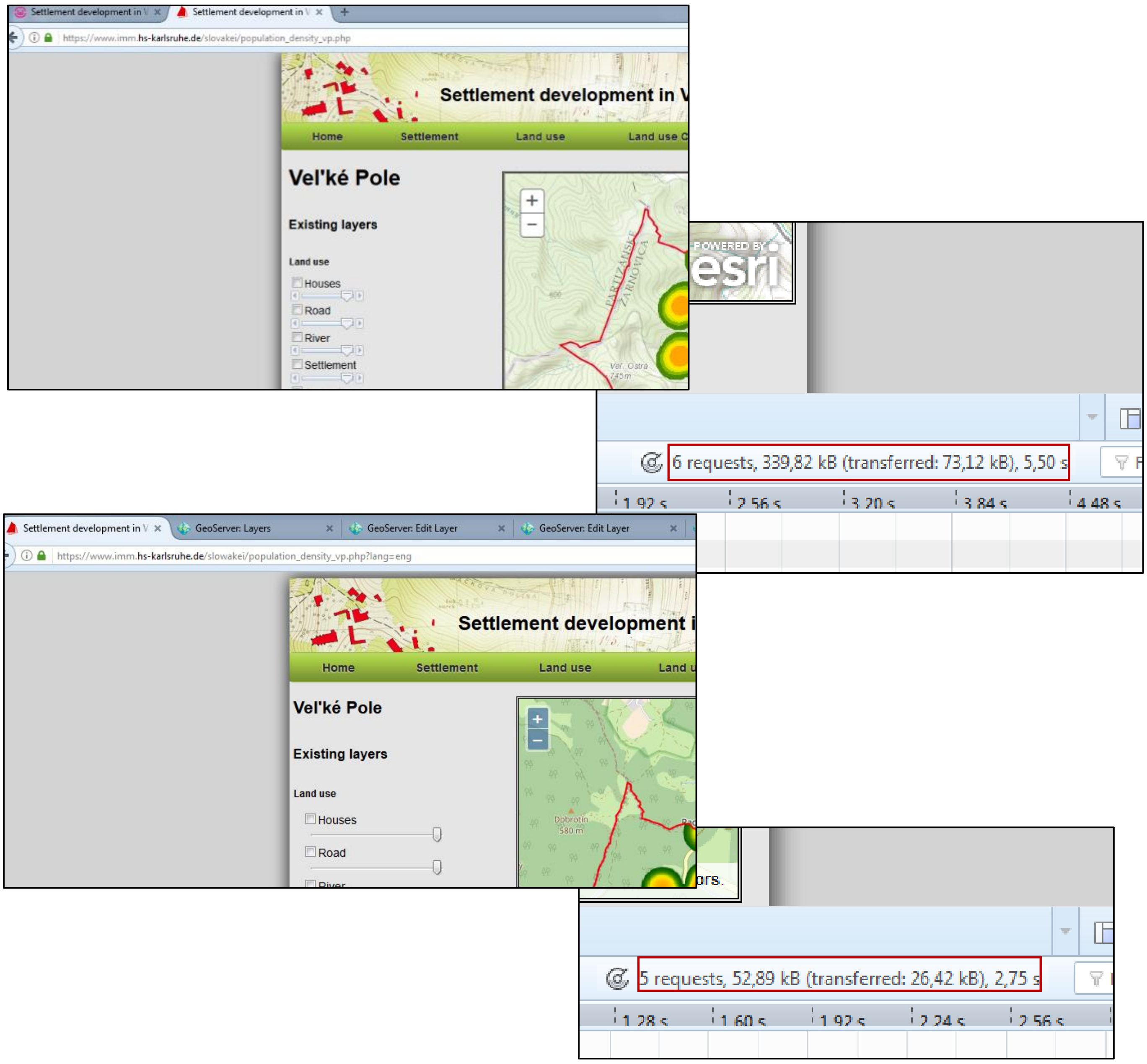
https://www.imm.hs-karlsruhe.de/slowakei/settlement_velkepole.php

5 requests, 52,89 kB (transferred: 26,42 kB), 3,08 s

Figure 5.53: Comparison between ESRI website (above) vs. Openlayers website (bottom). Part 1

The Vel'ké Pole population density php has also been tested. This php shows a graded colour population density raster map.

The data transfer is the same for both php and, again, the website with Openlayers is twice faster.


Settlement development in V
https://www.imm.hs-karlsruhe.de/slovakei/population_density_vp.php
Settlement development in V
Home
Settlement
Land use
Vel'ké Pole
Existing layers
Land use
Houses
Road
River
Settlement
POWERED BY esri
GeoServer: Layers
GeoServer: Edit Layer
https://www.imm.hs-karlsruhe.de/slowakei/population_density_vp.php?lang=eng
Dobrotin 580 m

6 requests, 339,82 kB (transferred: 73,12 kB), 5,50 s

5 requests, 52,89 kB (transferred: 26,42 kB), 2,75 s

Figure 5.54 Comparison between ESRI website vs. Openlayers website. Part 2

Then, it can be asserted that **Geoserver and Openlayers work more efficiently and faster than ESRI.**

6. Conclusions

Regarding the website and its content, the database is not so big, and there are also a few web maps. The important map info on this website is shown in large scales, with more or less 5 zoom levels.

The website shows all its data in large scales so, the advantage of Mapnik and MapServer doesn't matter. In fact, Mapnik deals with a re-projection using proj4 library, which makes Mapnik slower than the other map servers when working on large scales. Furthermore, Mapnik and MapServer are good at dealing with huge data because of its RAM management. In this part, Geoserver is the worst because it needs so much RAM for processing tiles although there are not so many layers and base maps in this website so, it's no necessary to care about RAM processing.

For this reason, the website doesn't have to be programmed with those open source servers. Because when working with a small database and few objects, the difference in performance between all the different map servers is not very big.

Geoserver feels like the best choice for the purposes of this website, also because it has a very friendly interface for managing and publishing WMS and WFS.

Anyway, MapServer is more difficult to manage than Geoserver because of its lack of GUI, but it has better performance than Geoserver with tile processing and handling several requests to the servers.

Once the server has been chosen, the javascript library for the map viewer has to be chosen. The options are Openlayers and Leaflet. Leaflet is the most used javascript library nowadays, but it needs a test for checking if the original functions works on Leaflet. Unfortunately, Leaflet has not opacity switcher in its layer control, and it's not much faster than Geoserver, thus Geoserver has been chosen.

The only function that had to be programmed was made using CQLFILTERs. This attribute is for filtering the Geojson response depending of its definition. The function gets the name from the Mysql query and add it to this filter. This Geojson gets the data from Postgres database and shows the same building than is being searched into MySQL data base.

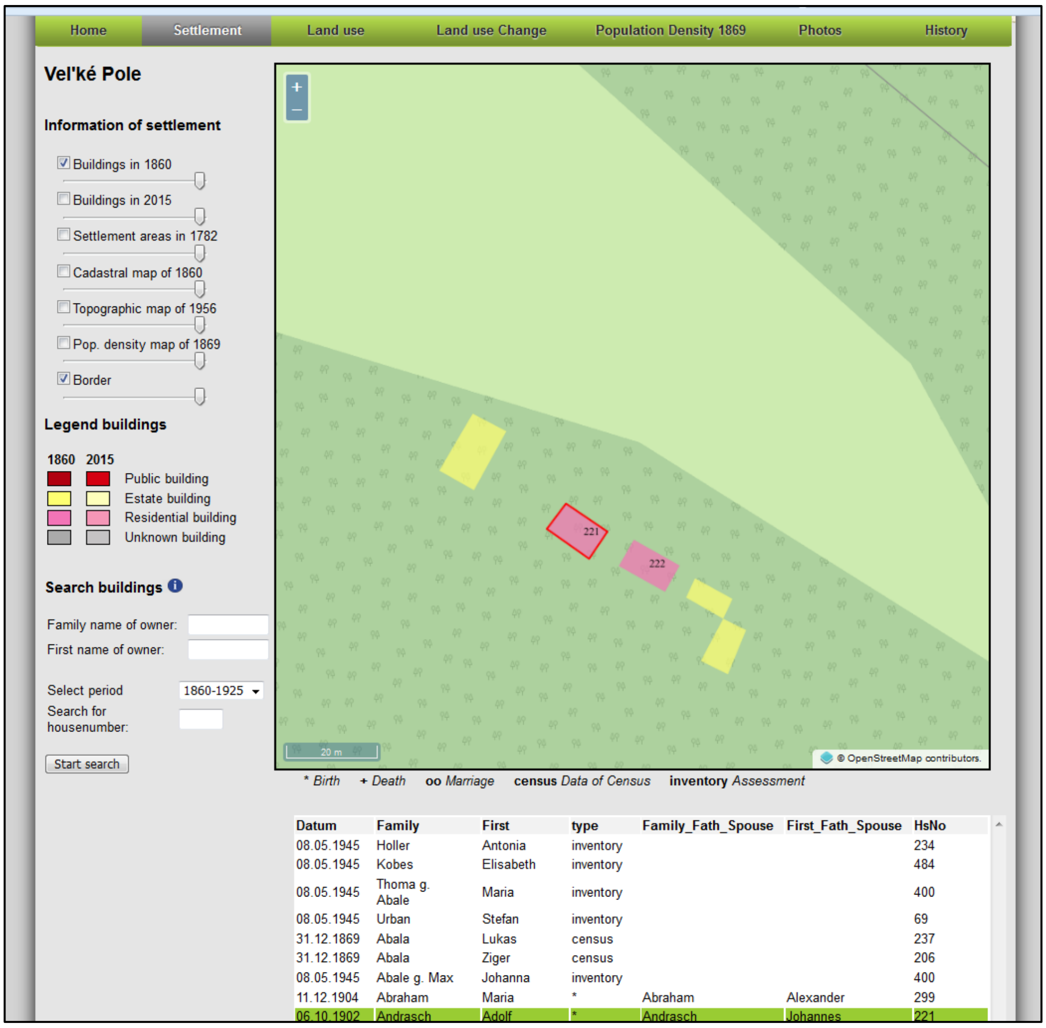
After having done these functions, it was easy to convert the old website into the new Apache Tomcat server, with Geoserver and Openlayers.

Finally, both websites have been tested, the website with ESRI software and the other with Open Source software. This test has been made outside the HsKA network and the results are conclusive: **The website with Open Source works better than the other with ESRI software.** With Geoserver and Openlayers, the website is twice faster and needs less libraries than with ESRI software.

This experience is great for learning how a SIG web map works, with get and post requests, and getting data from MySQL data base and Postgres at the same time. Maybe some companies would like to change their premium servers for an open source server in the same way, and this is a good chance for learning how to do it. Besides, **is a good chance to prove that the Open Source alternative is the best choice for building a website with web map viewers**. The commercial products maybe are more comfortable for managing, but **with Open Source, the website can be twice faster than commercial products.**

Later, some captures will show the final result of the new website.

This capture is showing how all functions involved on the search query are working and the functions for showing the building which is being clicked on the table.



Vel'ké Pole

Information of settlement

☑ Buildings in 1860

☐ Buildings in 2015

☐ Settlement areas in 1782

☐ Cadastral map of 1860

☐ Topographic map of 1956

☐ Pop. density map of 1869

☑ Border

Legend buildings

1860	2015	
🟥	🟥	Public building
🟨	🟨	Estate building
🟪	🟪	Residential building
⬜	⬜	Unknown building

Search buildings

Family name of owner:

First name of owner:

Select period 1860-1925

Search for housenumber:

Start search

* *Birth* + *Death* **oo** *Marriage* **census** *Data of Census* **inventory** *Assessment*

Datum	Family	First	type	Family_Fath_Spouse	First_Fath_Spouse	HsNo
08.05.1945	Holler	Antonia	inventory			234
08.05.1945	Kobes	Elisabeth	inventory			484
08.05.1945	Thoma g. Abale	Maria	inventory			400
08.05.1945	Urban	Stefan	inventory			69
31.12.1869	Abala	Lukas	census			237
31.12.1869	Abala	Ziger	census			206
08.05.1945	Abale g. Max	Johanna	inventory			400
11.12.1904	Abraham	Maria	*	Abraham	Alexander	299
06.10.1902	Andrasch	Adolf	*	Andrasch	Johannes	221

Figure 6.55: A capture with the website of the final version.

The photo's php capture is showing how the old functions are working well in the new server.

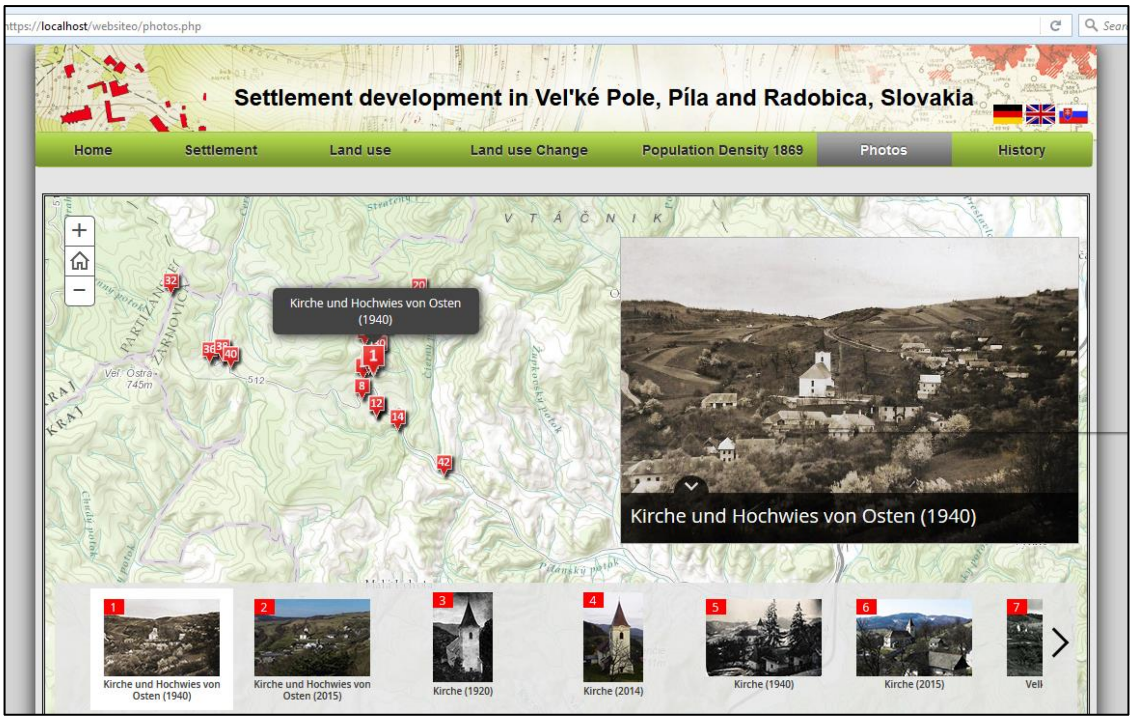
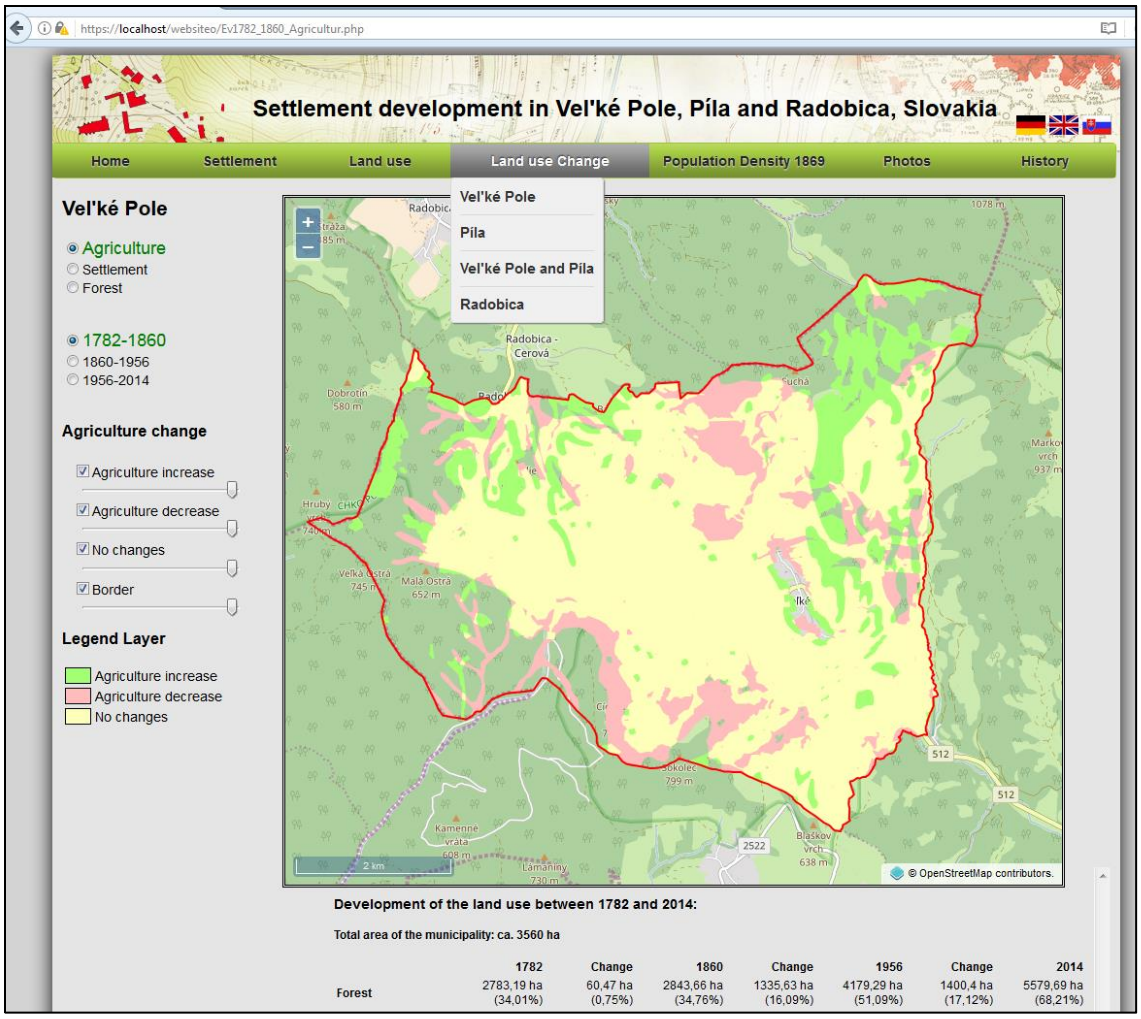

https://localhost/websiteo/photos.php
Settlement development in Vel'ké Pole, Píla and Radobica, Slovakia
Home
Settlement
Land use
Land use Change
Population Density 1869
Photos
History
Kirche und Hochwies von Osten (1940)
Kirche und Hochwies von Osten (1940)
Kirche und Hochwies von Osten (2015)
Kirche (1920)
Kirche (2014)
Kirche (1940)
Kirche (2015)

Figure 6.56: A capture with the website of the final version.

And the right's capture shows how all the legends and the different website parts are well organized with the new code.



Development of the land use between 1782 and 2014:

Total area of the municipality: ca. 3560 ha

	1782	Change	1860	Change	1956	Change	2014
Forest	2783,19 ha (34,01%)	60,47 ha (0,75%)	2843,66 ha (34,76%)	1335,63 ha (16,09%)	4179,29 ha (51,09%)	1400,4 ha (17,12%)	5579,69 ha (68,21%)

Figure 6.57: Three captures with the final version of the new website.

And here is the url: https://www.imm.hs-karlsruhe.de/slowakei/index.php

7. References

Documentation about QGIS Server, *QGIS Server,* n.d. Available from: http://docs.qgis.org/1.8/en/docs/user_manual/working_with_ogc/ogc_server_support.html

Gerd Katzenbeisser, GitHub Mapnik, *installing Mapnik in Windows* Available from: https://github.com/mapnik/mapnik/wiki/WindowsInstallation [5 October 2015].

Genealogy: Slovakia., 2000 n.d., *General Information* Available from: http://www.genealogy.net/reg/ESE/slovak.html [1 July 2000].

Settlement development in Vel'ké Pole, Pila ad Radobica, Slovakia 2016 n.d. *History of German Migration to and from Brezova, Slovakia* Available from:

https://www.imm.hs-karlsruhe.de/slovakei/history_brezova.php [June 2016].

Settlement development in Vel'ké Pole, Pila ad Radobica, Slovakia 2016 n.d. *History of German Migration to and from Vel'ké Pole, Slovakia* Available from:

https://www.imm.hs-karlsruhe.de/slovakei/history_velkepole.php [June 2016].

Settlement development in Vel'ké Pole, Pila ad Radobica, Slovakia 2016 n.d. *History of German Migration to and from Píla, Slovakia* Available from:

https://www.imm.hs-karlsruhe.de/slovakei/history_pila.php [June 2016].

Settlement development in Vel'ké Pole, Pila ad Radobica, Slovakia 2016 n.d. *Radobica* Available from:

https://www.imm.hs-karlsruhe.de/slovakei/history_radobica.php [June 2016].

Maxim Rylov, 2008-2017, MapSurfer.NET, *Benchmarking Mapping Toolkits in Tile Seeding*, Available from: http://mapsurfernet.com/blog/benchmarking-mapping-toolkits-in-tile-seeding 23 June 2015.

OSGeoLive, *QGIS Server quickstart*., n.d., Available from: https://live.osgeo.org/en/quickstart/qgis_MapServer_quickstart.html.

Wiki GIS website about Geoserver, 2011, n.d., *Geoserver* Available from: http://wiki.gis.com/wiki/index.php/GeoServer [9 May 2011].

Jeff McKenna, 2017, MapServer website, *An introduction to MapServer* Available from: http://MapServer.org/introduction.html#introduction [8 February 2017].

OSGEO Live, n.d., *MapServer Quickstart* Available from: https://live.osgeo.org/en/quickstart/MapServer_quickstart.html.

OSGEO wiki, *MapServer*. Available from: https://wiki.osgeo.org/wiki/MapServer [8 November 2010]

Openlayers, *Openlayers examples, n.d.* Available from: http://openlayers.org/en/latest/examples/index.html.

Openlayers, *Layer Groups*, n.d., Available from: http://openlayers.org/en/latest/examples/layer-group.html.

Wiki Openstreetmap website about Mapnik 2016 n.d., *Mapnik*. Available from: http://wiki.openstreetmap.org/wiki/Mapnik [31 May 2017].