Universitat Oberta de Catalunya

Mapping of forest fires through satellite imagery

Carlos Sarille Cadenas
Telecommunications Technologies Degree

Supervisor: **Anna Muñoz Bollas**

05/06/2018

Para María, por todas as noites roubadas

Final Project Data Sheet

Title:	*Mapping of forest fires through satellite imagery*
Author:	*Carlos Sarille Cadenas*
Supervisor:	*Anna Muñoz Bollas*
Date (mm/yyyy):	*06/2018*
Department:	*SIG i geotelemàtica*
Degree:	*Telecommunications Technologies Degree*

Abstract

Wild fires are a major problem in the northwest of the Iberian Peninsula. Only in Galicia during 2006 96 000 hectares were burned.

There is no official cartography of those catastrophic events, and some of them are not even registered.

This project develops an automated downloading and processing system for European Union's Sentinel satellites, in order to map burned land efficiently and with a high frequency.

NBR and NBR-difference indexes are calculated from Sentinel imagery for the Galician area, using different imagery processing libraries.

Later on, the data is transformed into vector files and uploaded to a geographical database to be show on an interactive web map.

Resumen

Los incendios forestales son un problema de primer orden en el noroeste de la península ibérica. Sólo en Galicia en 2006 ardieron más de 96 000 hectáreas.

No existe una cartografía oficial de estos incendios, y algunos ni siquiera constan como tales en los registros.

Este trabajo desarrolla un sistema automatizado de descarga y procesado de imágenes de satélite provenientes de los satélites Sentinel de la Unión Europea, con el fin de proporcionar una cartografía de la superficie quemada de forma rápida y con una periodicidad alta.

Se calculan a partir de estos datos los índices NBR y NBR-diferencia para el área correspondiente a Galicia, mediante el uso de diferentes librerías de procesado de imágenes geográficas.

Posteriormente estos datos se vectorializan y se vuelcan a una base de datos geográficos para ser mostrados en un mapa interactivo.

Keywords

Map Server, GIS, Sentinel, Satellite Imagery, Wild Fires, Forest Fires

Index

Index v
List of Figures vii
1. Introduction 1
  1.1 Justification 1
  1.2 Problem Formulation 1
  1.3 Main Objectives 1
  1.4 Outcoming 2
  1.5 Brief summary of the developed products 2
  1.6 Organization 3
    PAC 1 3
    PAC 2 3
    PAC 3 4
    PAC 4 5
2. Milestones 6
3. Risks 7
4. Materials 8
  Hardware 8
  Software 9
5. State of the Art. Concepts 10
  Satellite Imagery: An overview 10
  Satellite Imagery: Copernicus 11
  Satellite Imagery: Landsat and private companies 13
  Land Classification: Spectrum analysis 13
    Colour Composites 13
    Indexes 14
  Land Classification: Object Based Image Analysis (OBIA) 16
6. State of the art. Technologies 17
  Web-based Maps: Introduction 17
  Open Source Geographical databases 17
    PostGis 17
  Web Map servers 17
    GeoServer 18
  Web Map libraries 18
  Web servers: Apache 18
  Java Applet Containers 19
    Apache Tomcat 19
  Python 19
    GDAL 19
    SentinelSat 20
  Cloud Detection 21
    Fmask 21
7. Design Stage 24
  System Overview 24
  Backend Tier 25
    Admin Script 25
    Download Tool 26
    Unzip and Order Tool 26
    Processing tool 26

Folder System ... 27
Middle Tier ... 27
Presentation Tier ... 28
Webmap ... 28
8. Implementation Stage ... 29
Backend Tier: Dependencies ... 29
Numpy ... 29
GDAL ... 29
Sentinel Sat ... 29
Winrar ... 29
Shape and Geojson ... 30
Backend Tier: Implementation ... 31
Admin Script ... 31
Download Script ... 32
Unzip script ... 34
Processing script ... 36
Middle Tier ... 39
PostGis ... 39
Presentation Tier ... 40
Apache ... 40
Apache Tomcat ... 40
GeoServer ... 41
Leaflet ... 42
Web map ... 43
Production Stage ... 44
9. Results ... 45
Backend Tier ... 45
1-Download Imagery ... 46
Order and unzip downloaded files ... 47
Purge imagery content ... 47
Show folder content ... 47
Clip rasters ... 47
Calculate NBR Index ... 48
Calculate difference between NBR indexes ... 48
Vectorize NBR and NBR difference ... 49
Upload files to PostGIS ... 49
Batch Processing ... 49
Products Obtained ... 49
Presentation Tier ... 50
10. Conclusions ... 53
Results ... 53
Learned Lessons ... 53
Problems Found ... 53
Future work lines ... 54
11. Annexes ... 55
SLD style for NBR difference layer ... 55
SLD style for NBR difference layer ... 57
Windows 10 Installation ... 60
12. Glossary ... 69
13. Bibliography ... 70

List of Figures

1 - Original PAC2 work plan 4
2 - Original PAC3 work plan 4
3 - Sun-synchronous Polar Orbit 11
4 - Sentinel-2 spectral bands........ 12
5 - Different Colour Composites 14
6 - NDWI and NBR indexes........ 14
7 - As Neves district, before and after the 2017 October wildfires seen in RGB colour composite of bands SWIR/NIR/Red. Burnt areas are highlighted.......... 15
8 - As Neves district. NBR index (left) and BAI index (right) after the 2017 October wildfires........ 16
9 - Segmentation and classification Example........ 16
11 - Geojson file covering area of interest........ 20
12 - Sentinel Tiles........ 21
13 - Sentinel-2 B08 band cloudy scene........ 22
14 - Superimposed cloud mask........ 22
15 - Superimposed Fmask detection. Cluds and shadows........ 23
16 - System Architecture 24
17 - Backend Architecture 25
18 - Folder Structure 27
19 - Presentation Tier Structure 28
20 - Original Shapefile........ 30
21 - Modified Shapefile........ 30
22 - Database detail 39
23 - Apache default web page........ 40
24 - Tomcat default web page........ 41
25 - Geoserver default web page 41
26 - Geoserver Time Domain 42
27 - Admin Script Main Menu 46
28 - Download Imagery Option........ 46
29 - Show Folder Content Option........ 47
30 - Calculate Difference Option 48
31 - Batch Processing Option........ 49
32 - Products Obtained 50
33 - Web Map main view........ 50
34 - Web Map Base Layer Options 51
35 - NBR difference Layer........ 511
36 - Cadastral Parcels and NBR Index........52

1. Introduction

This chapter will briefly describe the project objectives and justification.

1.1 Justification

Wildfires are a major problem in the northwest area of the Iberian Peninsula. During the summer season of 2006, only in the region of Galicia, around 96,000 hectares were burnt (1) in what was one of the worst years in terms of wildfires. Last year, although there aren't statistics yet, it is estimated that around 61,000 hectares were burnt (2).
There are several causes for this problem, among them the depopulation of rural areas, lack of forest care, climate change and a forest sector that despite of its size in economic terms, number of people employed and importance for the region (3), hasn't been able to deal with the periodical wildfire crisis.
In order to perform a good analysis of the roots of this problem, keep track of the affected surfaces and allow land owners, investigators and government to have access to this data, this proposal focuses on the use of GIS technologies (Geographical Information Systems) and satellite imagery to map the areas affected by forest fires.

1.2 Problem Formulation

Nowadays it isn't known the real extension and severity of the wildfires, and there isn't cartography available of those burnt surface that could help to study and prevent those fires. The extension and severity of the fires is measured on terrain in the cases of big fires, and estimated in other cases. In some cases when people extinguish the fires with no help from firefighters, those are not even registered.
Although the previously mentioned cartography is a legal obligation of the Galician autonomic government (4), these data hasn't been collected and made public yet.
This proposal tries to fill that gap, providing an easy and reliable tool to map and consult the areas affected by forest fires. Furthermore, this tool can easily be expanded to show other useful data as cadastral parcels, national parks, etc. integrating a complete "wildfires composed viewer".

1.3 Main Objectives

- Understanding of the different GIS technologies, methodologies and structures.
- Creation of an integrated system to automatically download, process, analyse and cartography satellite images with the objective of detecting wildfires.

- Creation of a web map service, allowing users to view and interact with the data gathered and generated.
- Use of Open Source software through all the stages of the process when available.
- Contribute to a better management of forest areas and environment protection.

1.4 Outcoming

Although the main goal of this project is focused on wildfires, many other environment related projects could be done using the same technologies and processes with small modifications. Monetization could be possible, as land owners and governments could be interested on this kind of data.
Moreover, specific applications regarding construction or agriculture industry, which require more precision could be achieved with more detailed imagery, which could be bought from private satellite companies.
Overall, this kind of technology is becoming more and more popular and useful, and almost every industry is potentially interested on it, as long as there is something that can be located on a map. The big challenge is to generate the data shown on it.

1.5 Brief summary of the developed products

- Interactive Web map: Web map written in JavaScript using the Leaflet library.
- Geographical Database and Maps Server: Configured and updated Database and Maps server, using PostGIS, Rasdaman and/or Geoserver.
- Coordination software: Provides an interface with the system administrator and that executes and manages the whole automated map generation system.
- Image acquisition Script: It downloads the required layers from the Sentinel HUB.
- Image processing Script: It clips the downloaded images according to the designated area and the clouds mask provided by the Sentinel HUB[1]. Generates the raster files corresponding to the burnt areas.

[1] Sentinel Hub web page: https://www.sentinel-hub.com/

- Change Control: Compares two images from different dates performing the detection of new burnt areas.
- Layer Generation Script: Converts the raster files to vector files and uploads them to the Map server.

1.6 Organization

The project activities have been grouped as it follows:

PAC1	Work plan elaboration and general planning.
PAC2	Hardware and software set up. Basic Web map implementation. Coordination Script phase 1 and Image acquisition Script. Report writing.
PAC3	Image processing Script, Layer Generation and Change Detection. Coordination Script Phase2. Final Web map. Report Writing.
PAC4	Geobia. Report Writing. Presentation Elaboration

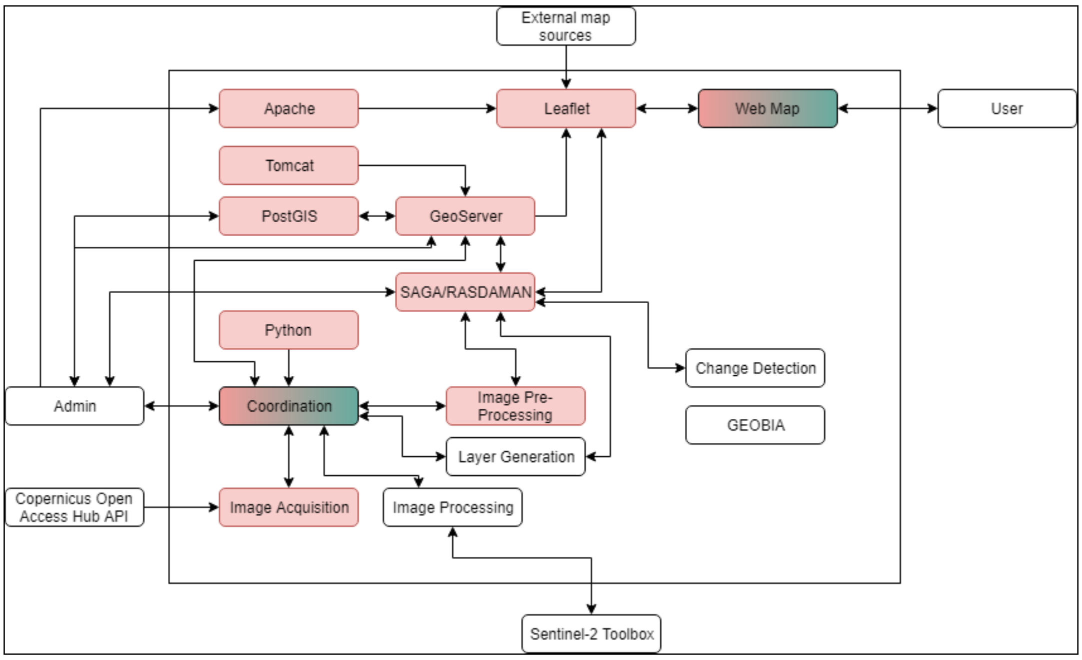
PAC 1

The first PAC consists on the elaboration of the workplan, which includes defining project objectives and scope, milestones, schedules, and risk analysis.

PAC 2

The second PAC comprehends system dependencies installation and configuration, as well as the first steps of coordination and acquisition software implementation.

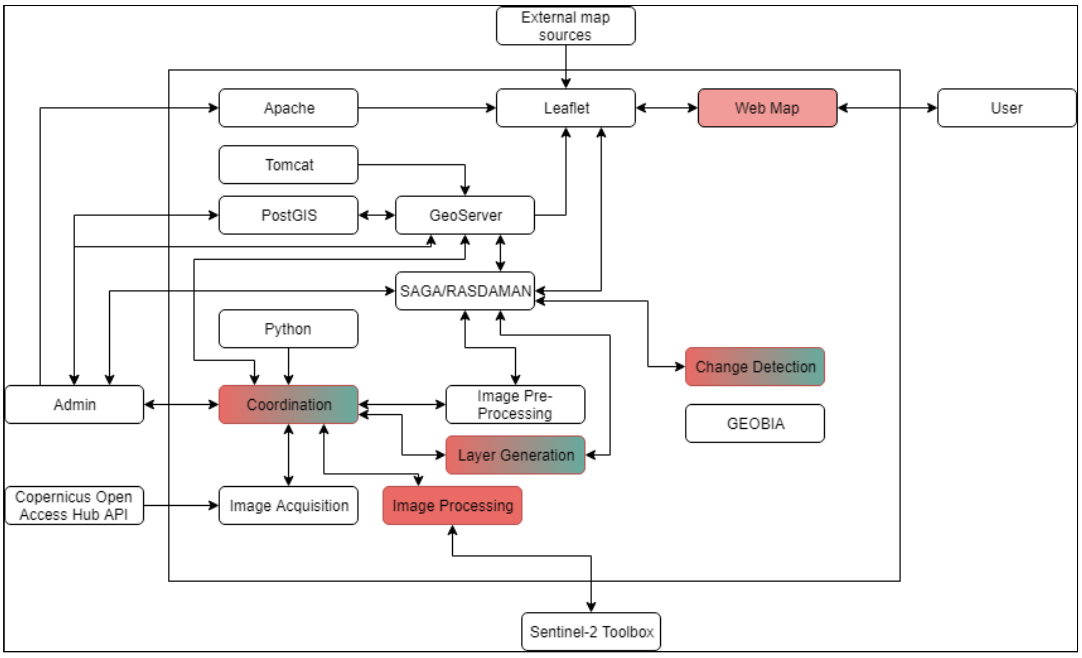
In the image 1 can be seen in red for complete stages and red/green for parcial stages, the original work plan as presented in PAC2.



1 - Original PAC2 work plan

PAC 3

PAC 3 includes finishing web map implementation, as well as image processing and layer generation software. Furthermore, most documentation must be done within this PAC. In the image 2, with the same colour code than the previous one, the PAC 3 original work plan is presented.



2 - Original PAC3 work plan

PAC 4

PAC 4 is the final phase. Layer generation implementation must be finished, as well as change detection software. If time allows, GEOBIA research and implementation can be done. Final memoir and video presentation are to be done in this phase.

2. Milestones

The main milestones of this project are:

Date	**Milestone**
20/02/2018	Course Start
07/03/2018	PAC 1 Draft Submit
09/03/2018	PAC 1 Submit
30/03/2018	PAC 2 Draft Submit
01/04/2018	PAC 2 Submit
06/05/2018	PAC 3 Draft Submit
11/05/2018	PAC 3 Submit
31/05/2018	PAC 4 Draft Submit
04/06/2018	PAC 4 Submit
19/06/2018	Virtual Debate Starts
21/06/2018	End of the Project

3. Risks

Regarding this project, the following risks have been identified:

Risk	Lack of time
Description	Problems finding time to get the project done
Impact	Project tasks duration
Probability	High
Action	Adapt the calendar trying to anticipate those situations.

Risk	Hardware Failures
Description	Failure of the computers involved
Impact	Project tasks duration
Probability	Low
Action	Repairing the machine, get a new one or migrate the whole system to another platform such as a VPS.

Risk	Planning mistakes
Description	Errors due to the lack of experience on the GIS field
Impact	Project Tasks Duration
Probability	Mid
Action	Spend more time implementing the tasks

Risk	Processing Complexity
Description	The computer isn't able to carry out the computation required.
Impact	Impossibility of getting the project results
Probability	Middle
Action	Split the processing queues into smaller ones. Use a second machine to distribute the computation.

Risk	Too Big Scope
Description	The Scope of the project is excessive
Impact	Project Features
Probability	Middle-High
Action	Remove features from the project

4. Materials

This chapter describes which Hardware and software will be used to develop the project.

Hardware

There will be several machines involved in this project, running different OS.

Most of the development will be done using a HP Omen 15 laptop. Main specifications are:

- Windows 10 Student Edition
- Intel core i7-7700HQ
- 16 Gb RAM
- NVIDIA GTX 1050 4GB GDDR5 graphics
- 1 TB HDD + 256 GB SSD

This machine will be used too to perform image processing for the web map demo.

The second machine is a Raspberry[2] Pi 3 Model B acting as a headless server. This is a cheap and reasonably powerful minicomputer that can be easily set up to work as a server.

The operative System is Raspbian, a derivative of Debian, which makes this system extremely easy to migrate to a production environment (i.e. a Virtual Private Server, a Hosting Service, etc.).

Raspberry Pi 3 main specifications are (5):

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- Micro SD storage

The Raspberry will be used to host the database during development, and to perform slow image processing once the system is up and running, as it will have around 3 days between each satellite imagery series.

There is a spare computer, with the following specifications:

- AMD X2 Dual core 64 bit processor
- 4 GB RAM
- 1 TB HDD

Furthermore, a web map demo, as well as a Geoserver instance and a Postgis database will be uploaded to a Linode[3] VPS server, in order to provide an easy way to see the project results. The characteristics are:

- 1 GB RAM

[2] Raspberry Pi webpage: http://raspberrypi.org/
[3] Linode webpage: https://www.linode.com/

- 1 CPU Core
- 25 GB SSD Storage
- 1 TB Monthly Transfer

Software

The following Open Source Software will be used in the system:

- Apache Web Server[4]
- Apache Tomcat[5]
- PostGIS/PostgrSQL[6]
- GeoServer[7]
- Leaflet[8]
- GDAL/OGR[9]
- Python[10]

Furthermore, during the development and documentation stages, the following software will be used:

- QGIS[11]
- LightTable[12]
- Microsoft Office Suite[13]
- Draw.io[14]

For Windows machines, Winrar[15] is used to unpack compressed files.

[4] Apache webpage: http://apache.org/
[5] Tomcat webpage: http://tomcat.apache.org/
[6] PostGis webpage: https://postgis.net/
[7] Geoserver webpage: http://geoserver.org/
[8] Leaflet webpage: http://leafletjs.com/
[9] GDAL/OGR webpage: http://www.gdal.org
[10] Python webpage: https://www.python.org/
[11] QGIS webpage: https://www.qgis.org/
[12] LightTable webpage: http://lighttable.com/
[13] Microsoft office webpage: https://www.office.com/
[14] Draw.io webpage: https://www.draw.io/
[15] Winrar webpage: https://www.winrar.es/

5. State of the Art. Concepts

This chapter briefly describe the main concepts used in this memoir.

Satellite Imagery: An overview

Artificial satellites surround our world, and they can be classified depending on their use, the distance from the surface or their type of orbit.

Attending to their use, several types of satellites can be found, such as communications satellites (telephony, television, internet…), military satellites (surveillance, military communications…), scientific satellites (climate monitoring, astronomic telescopes…), geolocation (GPS, Glonass, Galileo…), educational (small satellites called “CubeSats” from universities…) and many others.

Depending on their distance from the surface of the earth, we can classify them in three groups:

-Low Earth Orbit (LEO): Satellites located between 160km and 2000km from earth. They are located relatively close to earth. Because of that, their orbital periods (time they took to go around the globe) are short, typically between one and two hours. In this orbit are located the International Space Station (ISS), the Hubble Telescope and most of the CubeSats sent to orbit.

-Mid Earth Orbit (MEO): Region of space located between 2000 km and 36000 km of altitude. The main use for satellites this orbit is for navigation and communications, and it’s where GPS satellites are located. Orbital Periods range from 2 to almost 24 hours.

-High Earth Orbit (HEO): Region of space located above the geosynchronous orbits, this is around 36000 km. In this type of orbits, satellites apparently move backwards because their orbital period is greater than 24 hours.

-Geostationary orbit. At 35786 km above the equator, the orbital period is exactly 24 hours. This means that from earth perspective, a satellite will stay in a fixed position in the sky. It is a very useful orbit for communication satellites, as they can constantly provide service to the region underneath them.

-Geosynchronous orbit. The orbital period is 24 hours but the inclination is different to 0º. Thus they typically perform an eight figure pattern in the sky.

The differences on the periods of each orbit are due to the relation that describes the distance and speed for an orbiting object:

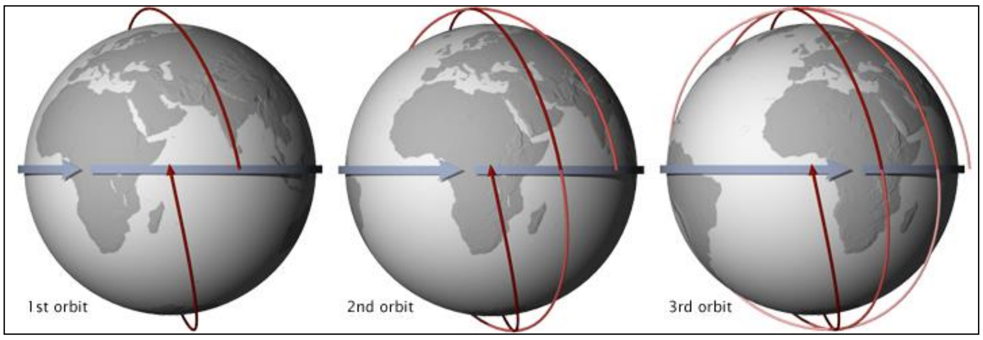
$$v = \sqrt{\frac{GM}{r}}$$

Where v is the speed of the orbiting object, G is the gravity acceleration of orbited object, M is the mass of the orbited object, and r is the radius of the orbit, this is the distance from the centre of the orbited body.

As it can be seen, the bigger the distance is, the speed of the object is slower. That is why LEO satellites orbit the earth every hour and a half, geostationary satellites stay apparently at the same point in the sky, and HEO satellites seem to move “backwards” in the sky.

Aside from the speed and altitude, the orbits of satellites can vary regarding the inclination they have to the equator. For example, geostationary satellites are 0º to the equator, and the ISS is 51.64º to the equator. This means that the maximum and minimum latitudes the ISS can reach are 51.64º north and 51.64º south over the earth.

There is a type of orbit used by some satellites called “polar orbit”, shown on image 3. Polar orbits are orbits with really big inclinations, close to 90º, so they orbit the earth from pole to pole. The advantage of this kind of satellites is that they can orbit over the whole globe, which is very useful for earth survey and recognition.



3 - Sun-synchronous Polar Orbit[16]

Satellite Imagery: Copernicus

Copernicus[17] is a European Union program for observing and monitoring the earth and its environments. As explained in the brochure Copernicus: Europe’s eyes on Earth, “the Copernicus programme places a world of insight about our

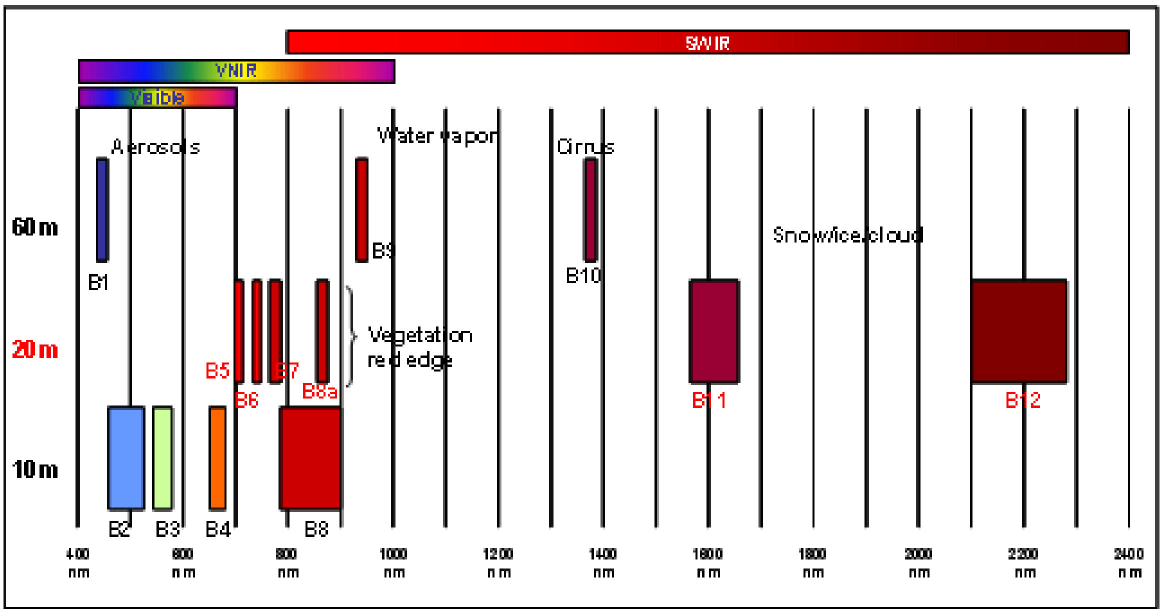
[16] Source: https://earthobservatory.nasa.gov/Features/OrbitsCatalog/page2.php
[17] Copernicus webpage: http://copernicus.eu/

planet at the disposal of citizens, public authorities and policy makers, scientists, entrepreneurs and businesses on a full, free and open basis" (5).
The Copernicus Program has six thematic focuses: Atmosphere, Marine, Land, Climate, Emergency, and Security (6).

The Copernicus Program operates using several satellites for earth observation and an on-site sensors network. Furthermore, it has an infrastructure network composed by ground stations, data centres and others, which allow the processing and distribution of the data gathered.

The satellites are called Sentinel, and there are several types of them. Sentinel 1 satellites provide radar imaging of the earth regardless of the weather. There are two Sentinel 1 satellites which operate on a polar orbit.

Sentinel 2 satellites provide optical imagery of the earth. There are two of them, operating on a polar orbit shifted by 180º.
The Sentinel 2 main instrument, Multispectral Instrument (MSI), allows them to get images of the earth on 13 spectral bands (7):



4 - **Sentinel-2 spectral bands**[18]

Sentinel 2 satellites have different spatial resolutions depending on the light band, and they are between 10 meters per pixel and 60 meter per pixel (7). The revisit frequency for Sentinel 2 satellites is 5 days, which means that every 5 days there will be new images for a given place.

It is important to mention that the Sentinel 2 instrument is sensitive to weather, and depending on the atmospheric conditions the image validity can vary. Sentinel 2 imagery is focused on "land management, agriculture and forestry, disaster control, humanitarian relief operations, risk mapping and security concerns" (8).

[18] Source: http://www.cesbio.ups-tlse.fr/us/index_sentinel2.html

Sentinel 3 will be a constellation of three satellites used for marine observation, and they observe sea-surface topography, sea temperatures and ocean colours. Only one satellite is operative nowadays.

Satellite Imagery: Landsat and private companies

Besides the European Copernicus Program, there are several other enterprises doing similar things. The most iconic one probably is the Landsat Program[19] executed by the National Aeronautics and Space Administration (NASA) from 1972.

Landsat are a series of satellites designated to multispectral Earth observation, in a similar way to the Sentinel-2 constellation. There has been eight Landsat missions, the Landsat-1 was launched on 1972, and the last one, Landsat-8, launched on 2013. Landsat-8 provides 11 spectral bands, and its data is available freely 24 hours after the acquisition.

Furthermore, several private companies have their own satellite constellations and sell the images that they capture, such as Planet Labs[20]. Planet Labs has a constellation of more than 175 small satellites in LEO, and they are able to provide images with a resolution up to 0.8 meters per pixel revisited daily (9).

There are more private companies such as Boeing[21] or Digital Globe[22], among others.

Land Classification: Spectrum analysis

When it comes to analyse and classify the type of terrain that can be seen in the satellite images, the easiest approach is Spectrum Analysis.
This technique performs a pixel-by-pixel analysis, by measuring the content of each spectrum band on each pixel, and creating a "terrain signature" which can be referenced to a type of land defined by the user.
For example, vegetation is highly reflective for the Near Infrared Band, so it will be high in their signature, while water surfaces are quite the opposite.
This technique works well when dealing with medium or low spatial resolutions (more than 10 or 15 meters per pixel). However, for better spatial resolution, the Object Based Image Analysis gives more accurate results (10).

By combining different spectral bands, different land properties will be highlighted. Those combinations can be colour composites or indexes.

Colour Composites

In satellite imagery, each pixel from each band contains monochrome information. In order to obtain a RGB visualization, those bands are assigned to

[19] LandSat webpage: https://landsat.usgs.gov/
[20] Planet Labs: https://www.planet.com/
[21] Boeing Space webpage: https://www.boeing.com/space/
[22] Digital Globe webpage: https://www.digitalglobe.com/

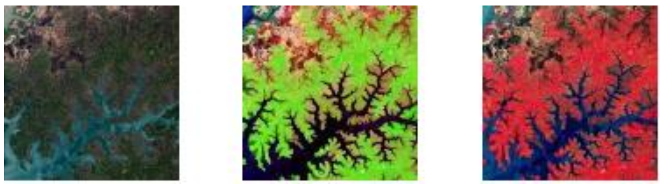
one of the three RGB channels. Deppending on the bands selected, the colour composite will highlight different properties of the selected area.

In the image 5, three different colour composite images of the same area can be seen.

The left one is a True Colour image, where bands 04, 03 and 02 are assigned to red, green and blue channels respectively, providing a colour configuration that looks as if seen by human eyes.

The image on the center is a Short-wave infrared composite, where bands B12,B08 and B04 are assigned to RGB channels. It highlights vegetation and land, as well as alterations as burnt areas or fires.

The image on the right hand side has a composition called False Colour, which is B08,B04 and B03 assigned to RGB channels and highlights in red vegetations areas. Those three are just a small sample of the colour compositions that can be done with satellite imagery.



5 - Different Colour Composites[23]

Indexes

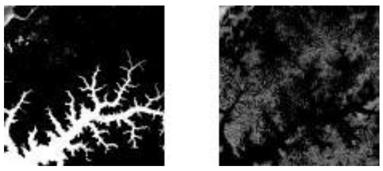
On the other hand, indexes are monochrome images obtained from performing algebra with different spectral bands in order to obtain a measurable trace of any feature. In the image 6 two indexes can be seen: Normalized Difference Water Index (NDWI) on the left hand side, and Normalized Burn Ratio (NBR) on the right hand side.

NDWI highlights water areas with brighter pixels, and it is calculated as:

$$NDWI = \frac{B03 - B08}{B03 + B08}$$

NBR highlights burnt areas with darker pixels, and it's calculated as:

$$NBR = \frac{B08 - B12}{B08 + B12}$$



6 - NDWI and NBR indexes[24]

[23] Source: https://www.sentinel-hub.com/develop/documentation/eo_products/Sentinel2EOproducts

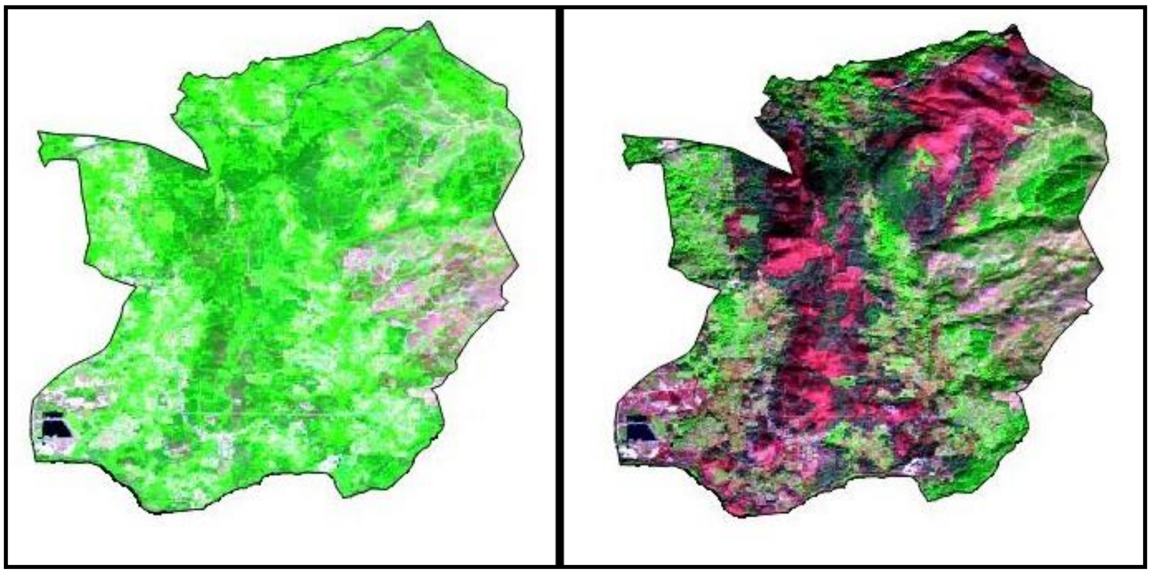
Another useful index for this project is the Burn Area Index (BAI), which highlights with brighter pixels areas covered by charcoal after fire events.

By subtracting the indexes of two different dates, a differential index can be created, where positive values will be new occurrences of the measured property (e.g. burned areas), while negative values will mean fading of the measured property (e.g. vegetation growing).
For the NBR difference index, values are characterised as follows[25]:

$$\Delta NBR = prefireNBR - postfireNBR$$

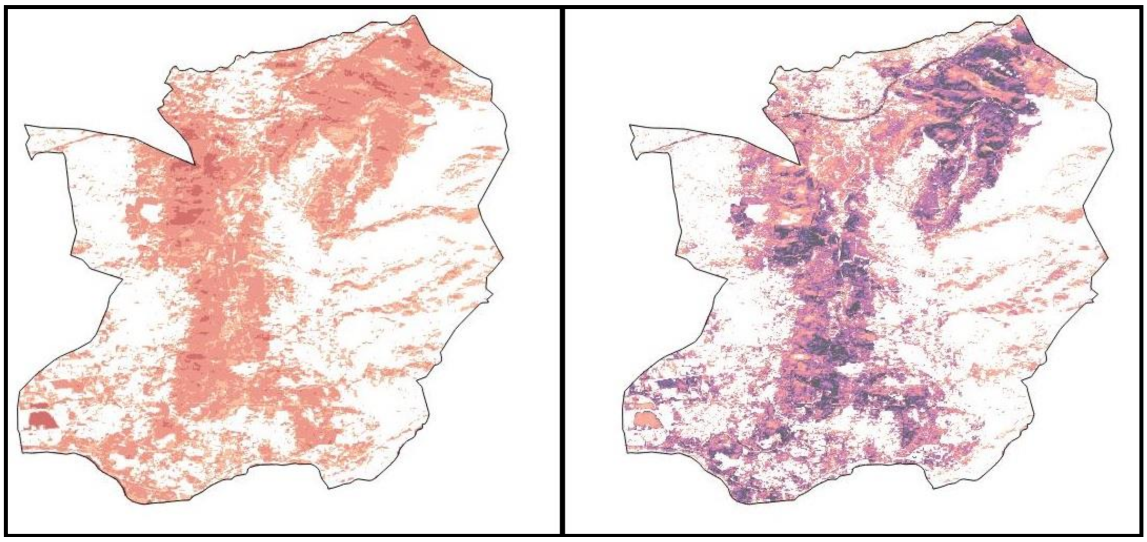
ΔNBR	Burn Severity
< -0.25	High post-fire regrowth
-0.25 to -0.1	Low post-fire regrowth
-0.1 to +0.1	Unburned
0.1 to 0.27	Low-severity burn
0.27 to 0.44	Moderate-low severity burn
0.44 to 0.66	Moderate-high severity burn
> 0.66	High-severity burn

In the images 7 and 8, a colour composite pre and post fire can be seen, as well as the same burned area in NBR index and BAI index.



7 - **As Neves district, before and after the 2017 October wildfires seen in RGB colour composite of bands SWIR/NIR/Red. Burnt areas are highlighted.**

[24] Source: https://www.sentinel-hub.com/develop/documentation/eo_products/Sentinel2EOproducts
[25] Land Scape toolbox: http://wiki.landscapetoolbox.org/doku.php/remote_sensing_methods:normalized_burn_ratio



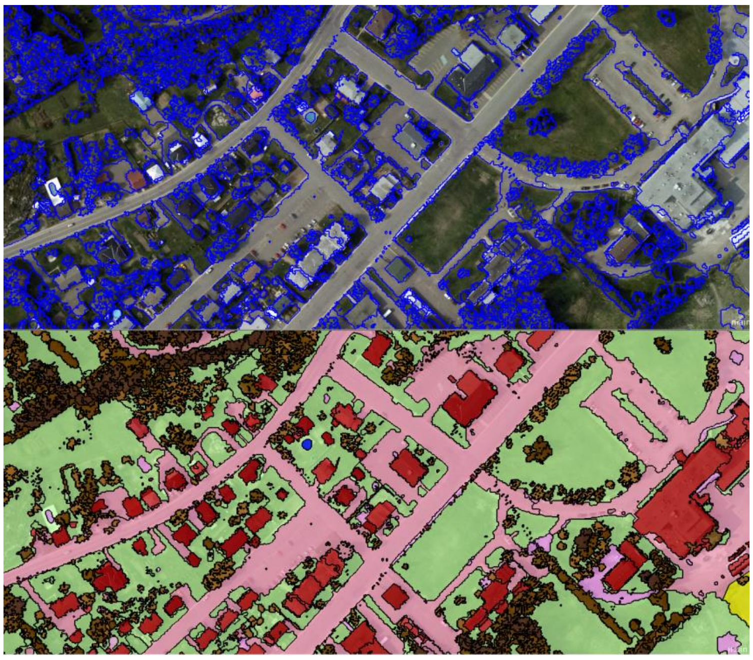
8 **- As Neves district. NBR index (left) and BAI index (right) after the 2017 October wildfires.**

Land Classification: Object Based Image Analysis (OBIA)

When the satellite images have enough resolution, i.e. the pixel size is smaller than the objects measured; Object Based Image Analysis (OBIA or GEOBIA) provides an accurate land classification.

GEOBIA works by segmenting images grouping similar neighbour pixels into objects, and then it looks for common patterns regarding their spectral and geometrical characteristics, performing the classification. Those algorithms can be trained in order to improve their segmentation and classification precision (11).

An example is shown on image 9[26].



9 - Segmentation and classification Example

[26] Source: https://gisgeography.com/obia-object-based-image-analysis-geobia/

6. State of the art. Technologies

This chapter describes the technologies used during the project implementation.

Web-based Maps: Introduction

There are several ways to present an interactive map in a webpage. Companies like Google, Apple or Microsoft have their own Map engines called Google Maps, Apple Maps and Bing Maps respectively. They not only provide a standard web map, but also allow developers to use their engine to build customized maps.

On the Open Source Side, the Open Street Maps (OSM) aims to provide an open source world web-map, elaborated collaboratively by the users.

Also, there are some pieces of open source geographical software that can be stacked to provide a complete Web-map engine, composed of the following parts: a geographical database, a map server and a web to display the maps. On top of that geographical processing tools can be added. This third option is the one chosen for this project.

Open Source Geographical databases

Geographical databases are simply databases that allow storing geo-referenced objects, such as points, lines and polygons. The two more relevant open source geographical databases are SpatiaLite and PostGIS.

SpatiaLite is a spatial extension for SQLite, an open-source relational database that doesn’t use the client-server paradigm; which allows geospatial queries.

PostGIS is a spatial extension for PostgreSQL, an open-source object-relational database, allowing geospatial queries.

PostGis

In this project PostGIS[27] will be used as database. PostGIS is a geographical information database based on PostgreSQL[28]. It is open source and it is available for Windows and Linux systems. PostGIS was selected because it is widely used, it has a powerful user community and it is easy to develop.

Web Map servers

In order to analyse the data contained in the database and render it for the final user, a web map server is needed. They can work with both vectorial and raster files (images and geometric objects), and provide an image of the map

[27] PostGIS webpage: http://postgis.net/
[28] PostgreSQL webpage: https://www.postgresql.org/

according to the style assigned for each layer, and the zoom level and position required. The more relevant open source web map servers are MapServer[29], Mapnik[30] and Geoserver.

MapServer was developed by the University of Minnesota. It is described as “an Open Source platform for publishing spatial data and interactive mapping applications to the web” (12).

Mapnik is a mapping toolkit which works with almost every geographical data formats, and it is widely used by institutions like OSM and companies as MapQuest or MapBox.

As Mapnick, Geoserver can work with most geographical data format and it is fairly popular. It is written on Java, so it needs to run over a Java Applet Container such as Apache Tomcat. It provides integrated tile caching and a web based control panel.

GeoServer

For this project, GeoServer is the chosen server. GeoServer[31] is an open source map server. It runs over Apache Tomcat.

Web Map libraries

In order to manage the map request and allow user interaction, a Web Map library must be used. Open Layers and Leaflet are the most popular ones.

Open Layers is an open-source, JavaScript library which allows the creation of complex interactive maps, supporting almost every geographical data format. The current version is Open Layers 4, and it has a vast community behind it.

Leaflet, as Open Layers, is an open-source JavaScript library. Although it is not as profuse as Open Layers, among its advantages are its simplicity, small size and easiness of use. Complex functions can be added through a considerable collection of plugins.

For this project, Leaflet is the chosen web map library,

Web servers: Apache

Apache[32] is a web server created by The Apache Software Foundation[33], which is open source and runs in different platforms. It is nowadays a *de facto* standard for http servers and it is used in this project to host the web map.

[29] Map Server webpage: http://mapserver.org/
[30] Mapnik web page: http://mapnik.org/
[31] Geoserver web page: http://geoserver.org/
[32] https://httpd.apache.org/
[33] https://www.apache.org/

Java Applet Containers

To execute Java code on web services, a Java applet container is needed. As Geoserver is partially written on Java, a Java Applet Container is needed. The more popular Java Applet Containers are Jetty[34] and Apache Tomcat.

Jetty is a web server with an integrated Java Applet Container, and it is maintained by the Eclipse Foundation.

Although the last versions of Geoserver are shipped with Jetty, for this project Apache Tomcat is the preferred Java Applet Container. The reason is that the combination of Tomcat+Geoserver is better tested and documented.

Apache Tomcat

Apache Tomcat[35] is a Java Applets container that runs on top of Apache. It is needed to execute Geoserver.

Python

Python[36] is an open source interpreted programming language.

There are two versions of this language, Python 2.7 and Python 3.6. The former is the legacy version of the language, and the latter is the present and the future of it[37].

In this project Python 3.6 is used as main language for the backend.

There is an immense collection of libraries to many different purposes, which enormously facilitate the implementation. For this project the following ones will be used:

GDAL

GDAL[38] "is a translator library for raster and vector geospatial data formats". GDAL provides a wide variety of tools to manipulate and calculate both raster and vector files. It is open source and it has a python API.
In this project, the following tools are used:

> -Gdal_calc.py[39]: provides a command line tool for raster calculations. It relies on numpy syntax, as the whole GDAL library is based on numpy[40], a python package for scientific calculations.

[34] Jetty web page: https://www.eclipse.org/jetty/
[35] http://tomcat.apache.org/
[36] https://www.python.org/
[37] https://wiki.python.org/moin/Python2orPython3
[38] http://www.gdal.org/
[39] http://www.gdal.org/gdal_calc.html

-Gdal_warp[41]: command line tool to perform reprojections and warp operations such as clips.

-Ogr2ogr[42]: command line tool that allows multiple operations on vector files.

-Gdal_polygonize[43]: command line tool to create vector files from rasters.

-Ogrinfo[44]: command line tool that provides several functions related to vectorial layers management.

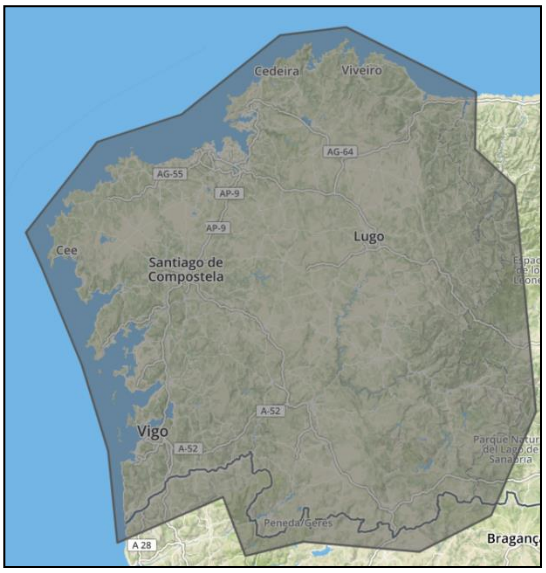
-Gdal

SentinelSat

“Sentinelsat makes searching, downloading and retrieving the metadata of Sentinel satellite images from the Copernicus Open Access Hub easy. It offers an easy-to-use command line interface and a powerful Python API.”[45]

Sentinelsat is currently maintained by Kersten Fernerkundung[46] and Martin Valgur[47]. It comes with a broad documentation[48] and it is reliable and multiplatform.

Sentinelsat is able to filter Sentinel products regarding cloud coverage and date. Also, a geojson file can be provided to indicate which area should be downloaded. The file created for this project is shown on image 11, and was created using the web tool geojson.io[49].

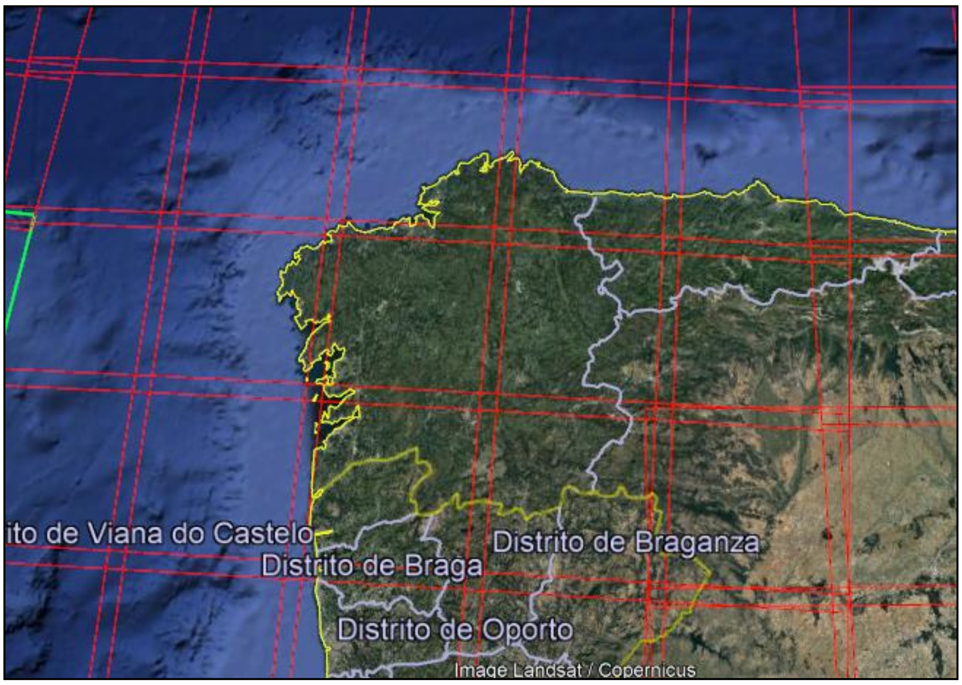


10 - Geojson file covering area of interest

Sentinel files are organised in tiles, each one covering a rectangle of approximately 110 square kilometres. These rectangles are named following a convention that takes into account the UTM zone, latitude and other

[40] http://www.numpy.org/
[41] http://www.gdal.org/gdalwarp.html
[42] http://www.gdal.org/ogr2ogr.html
[43] http://www.gdal.org/gdal_polygonize.html
[44] http://www.gdal.org/ogrinfo.html
[45] https://github.com/sentinelsat/sentinelsat
[46] https://github.com/Fernerkundung
[47] https://github.com/valgur
[48] http://sentinelsat.readthedocs.io/en/stable/index.html
[49] http://geojson.io/

parameters[50]. There is a .kml file provided by the European Space agency with the vectorial data of this grid[51], shown on image 12. Sentinelsat calculates which tiles have to be downloaded.



11 - Sentinel Tiles

Sentinelsat also needs a user name and password to access the SciHUB services. The account is free and only two concurrent download per user are allowed.

Cloud Detection

Cloud detection is fundamental for satellite imagery processing. Although with cirrus type clouds image correction can be performed, when it comes to opaque clouds, there is no way to avoid them.

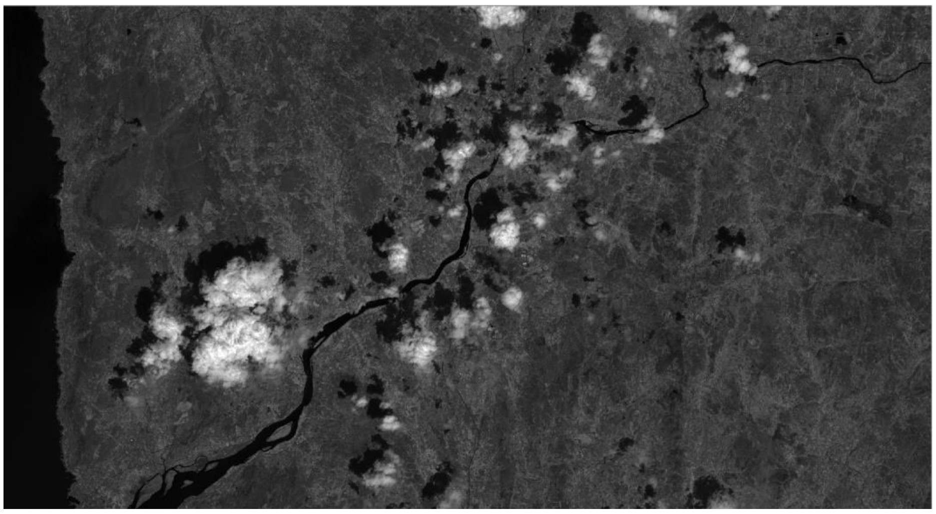
Sentinel-2 data comes with a cloud mask, which is not very accurate. There are many different cloud detection algorithms, from which Fmask stands out.

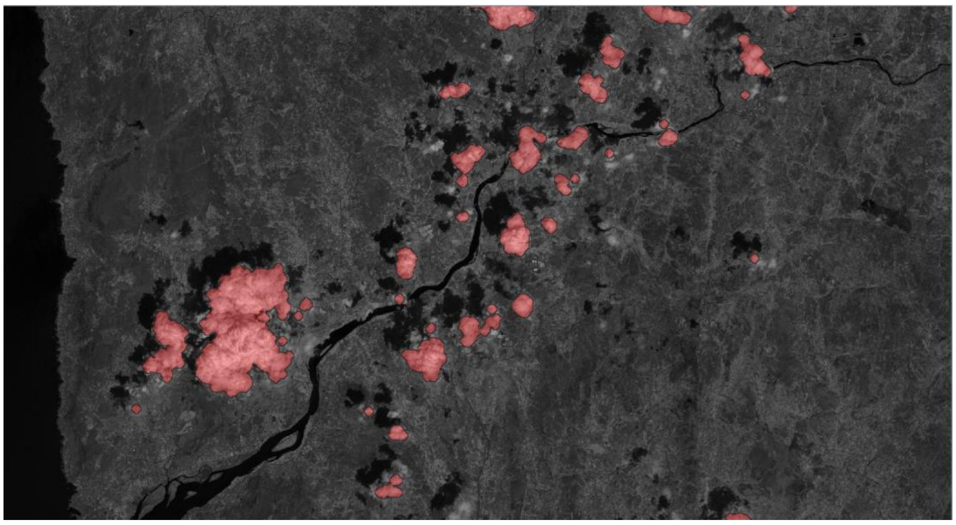
Fmask

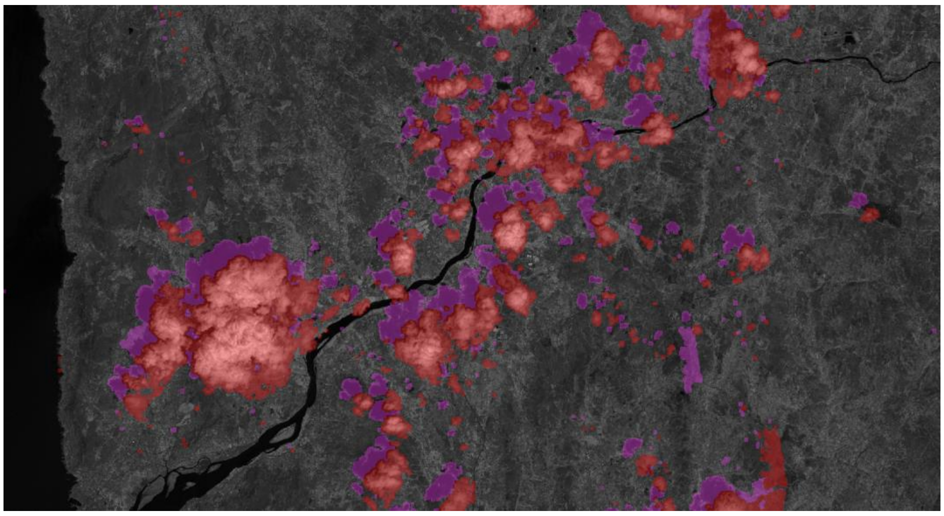
Fmask is an algorithm for cloud, cloud shadows, water and snow detection (13). It was developed for Landsat imagery but it was adapted to Sentinel-2 imagery. There are several implementations of this algorithm such as Pythonfmask[52] or matlab based Fmask[53]

[50] http://www.cesbio.ups-tlse.fr/multitemp/?p=5610
[51]https://sentinel.esa.int/documents/247904/1955685/S2A_OPER_GIP_TILPAR_MPC__20151209T095117_V20150622T000000_21000101T000000_B00.kml
[52] Pythonfmask web page: http://pythonfmask.org
[53] Fmask github page: https://github.com/prs021/fmask

The results of Fmask are far better than the cloud mask provided with sentinel imagery, as can be seen in the images below. Image 13 is a cloudy scene in the B08 band. Image 14 shows the Sentinel cloud mask superimposed to it. Image 15 shows Fmask detection for clouds and shadows.


12 - Sentinel-2 B08 band cloudy scene


13 - Superimposed cloud mask


14 - Superimposed Fmask detection. Cluds and shadows

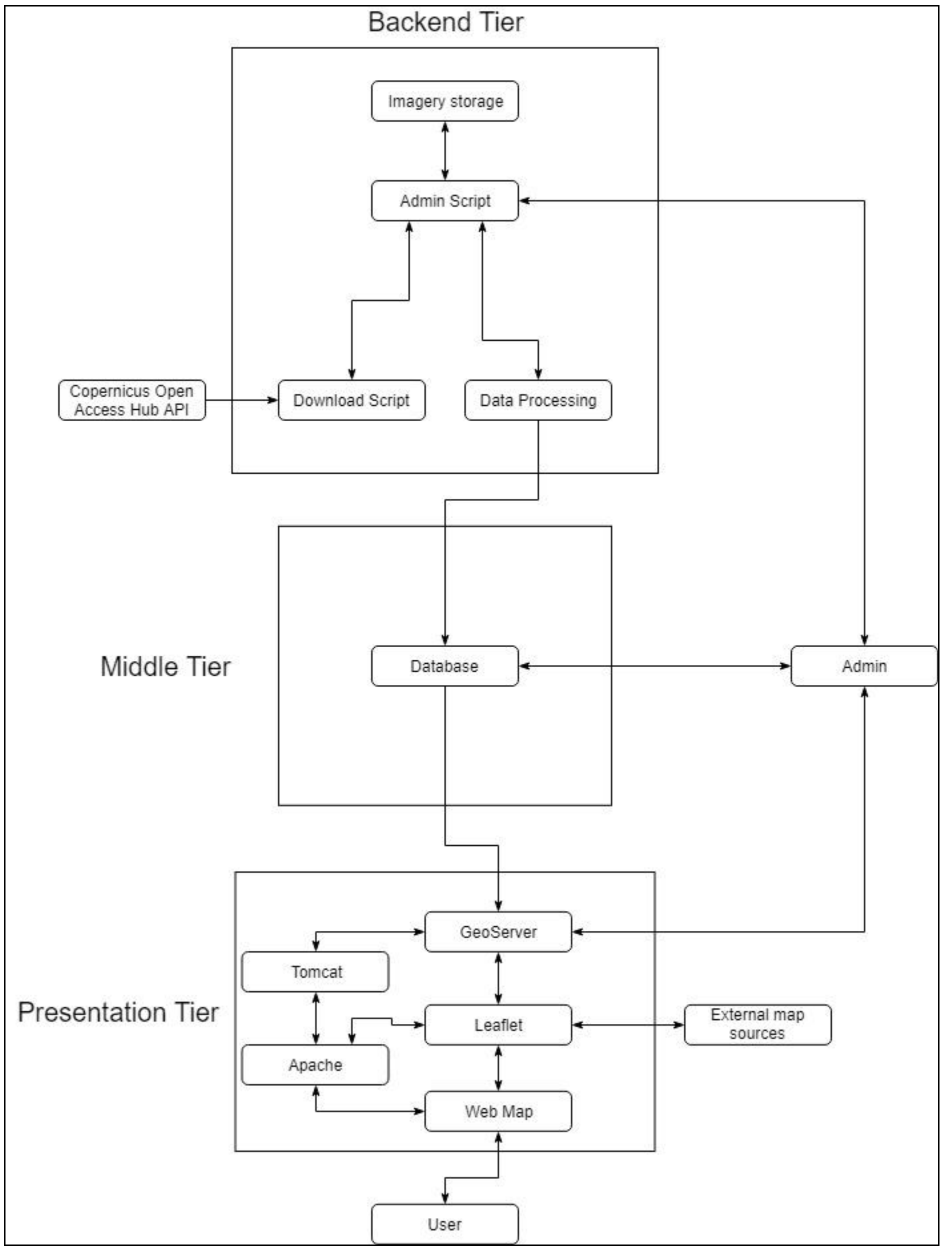
Although Fmask is more powerful, it requires a lot of time and processing power to perform the cloud detection, and the existing implementations are not as stable and reliable as they should be to perform unsupervised tasks. For those reason, Fmask is not used in this project. Instead, the cloud mask provided is preferred although it can introduce false positives.

7. Design Stage

This chapter describer the general structure of the system proposed.

System Overview

This project is structured in three levels. There is a Backend Tier, which performs duties as downloading imagery, processing it and upload it to the database. There is a Middle Tier, a PostGIS database where the backend places the processed data, and from where the Presentation Tier takes the data. Figure 16 shows the general architecture of the system.



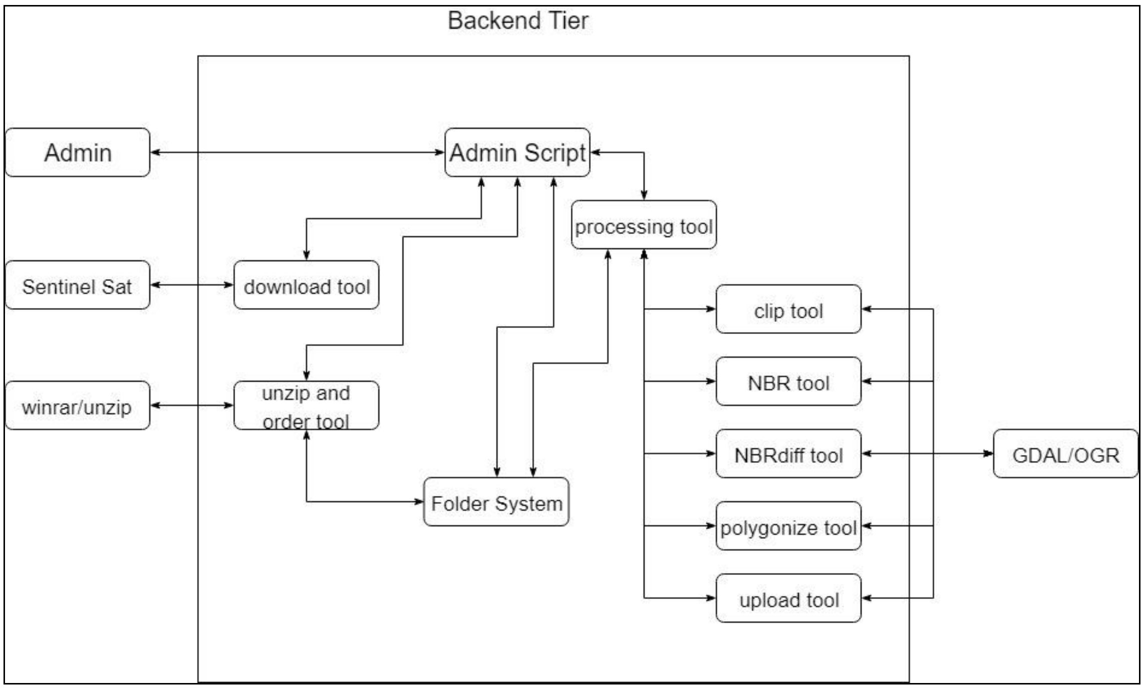
15 - System Architecture

It is worth to mention that the three different parts of the system can work on different but coordinated machines.

Backend Tier

The Backend level provides a complete imagery download and processing tool. It is made from different python scripts coordinated by an administration tool.

The architecture of the backend tier is structured as shown on figure 17.



16 - Backend Architecture

Admin Script

The admin script provides a management system that can be accessed via python interpreter. It allows the user to manage the backend tier and interact with it, requesting downloads under certain parameters, or ordering different processes such as image processing, file creation, etc. It also provides system information to the user.

Furthermore, this tool checks if the data provided by the user is correct (e.g. non existing dates, wrong folders, etc.), and provides information about the ongoing processes.

It allows to program batch processing, in order to perform unsupervised operations.

Download Tool

The download tool is a python script that can perform imagery downloading. It works as an interface between the admin script and the Sentinel Sat software, adapting date formats, cloud coverage and other options, and calling the function. It also checks data integrity (e.g no future dates, etc.).

Unzip and Order Tool

This tool provides two functions: one for ordering the downloaded files, i.e. move them to the folder system; and another one to unzip the files into named folders, select the bands of interest, and remove the rest of the file.

It also provides tools to view the content of the folder system, and to purge empty folders.

Processing tool

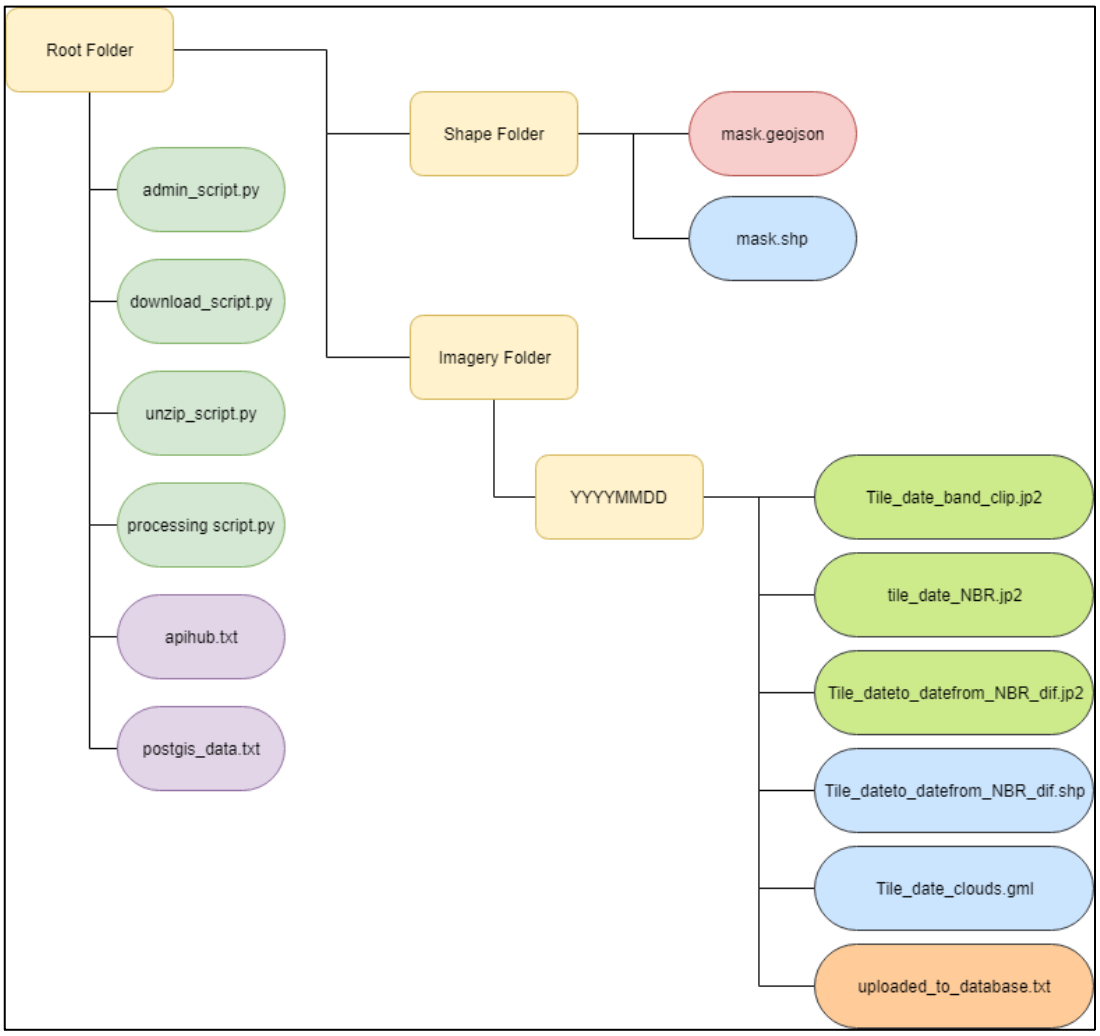
the Processing tool provides several methods to manipulate raster data, polygonize, clip and upload to postGIS image data.

It contains the following functions:

- Clip Tool: Clips the original raster data with a shape of the interest area, in order to avoid processing unnecessary data.
- NBR Tool: Calculates the NBR index for a given date and area.
- NBR difference Tool: Calculates the difference between two given date’s NBR index.
- Polygonize Tool: Transforms NBR indexes and NBR difference indexes from raster to vector files, categorizing fire severity in four different grades, and dismissing other data. This method saves up plenty of storage.
- Uploading Tool: Connects to a given database and uploads the vector data.

Folder System

Instead of a database, the backend works by directly manipulating files. The folder scheme has the structure indicated on figure 18 once the files are processed.



17 - Folder Structure

Middle Tier

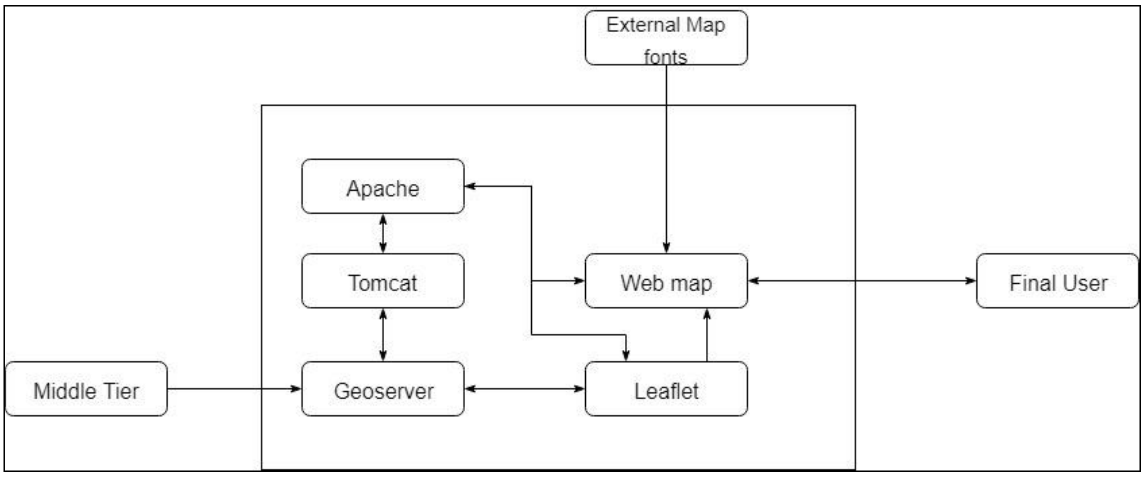
The Middle Tier consists on a PostGIS Geographical database, which will accept remote connections from the Backend Tier to update data, and from the Map Server on the Presentation Tier to retrieve data.

Presentation Tier

This tier manages data visualization on a web map.

It is made with a stack of a web server, a Java applet container, a maps server and, a web map library, and a web map.

It is structured as indicated on figure 19.



18 - Presentation Tier Structure

Webmap

To display the data, a webmap was created using Leaflet[54] and Geoserver[55].

The webmap has a main display with a Bing satellite base layer. Optional base layers are Open Street Maps[56] and Toner, a OSM based layer optimized for printers.

There is a mini-map with OSM base layer for orientation purposes. Furthermore, a cadastral layer can be displayed over the main layer. A time slider bar is located at the bottom. The interval time is 1 day, and it requests wms images from Geoserver, containing NBR difference and NBR data for each date. Additionally, an opacity control for the NBR difference layer is located under the layer control.

54 https://leafletjs.com/
55 http://geoserver.org/
56 https://www.openstreetmap.org/

8. Implementation Stage

This chapter describes how the project was implemented.

Backend Tier: Dependencies

The Backend tier has many dependencies. They will be discussed in this section.

Numpy

Numpy is a scientific package for python, on which rely GDAL and OGR. Installation is done with pip, the python package manager:

```
pip3 install numpy
```

GDAL

GDAL is in the core of the backend system.

GDAL installation depends on the operative system, and it is well documented for windows[57] .

It consists on downloading an installer and, after installation, add the path to the windows system path.

Alternatively, it can be installed by downloading a pre-compiled version and developing it with pip[58].

In linux systems is as easy as:

```
Sudo apt-get install python-gdal
```

GDAL comes integrated in some other packages as OSGeo[59], but it was avoided in this project due to potential compatibility problems.

Sentinel Sat

Sentinelsat installation is done via pip with the following command:

```
pip3 install sentinelsat
```

Winrar

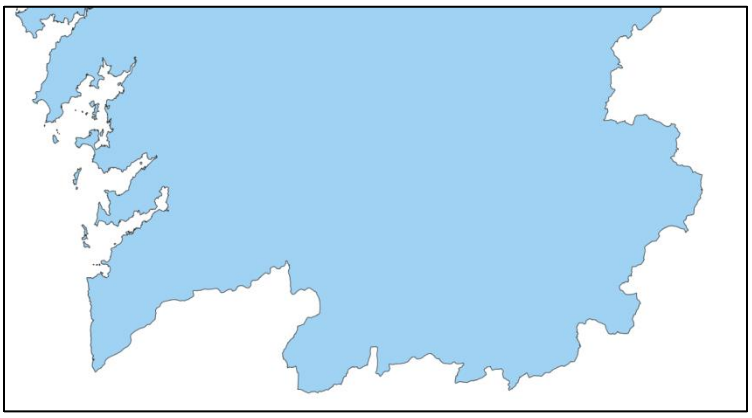
For windows systems, winrar path must be added to the system environment variables[60]. Linux systems already have unzip installed.

[57] https://viewer.nationalmap.gov/tools/rasterconversion/gdal-installation-and-setup-guide.html
[58] https://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal
[59] https://www.osgeo.org/projects/gdal/
[60] https://cects.com/using-the-winrar-command-line-tools-in-windows/

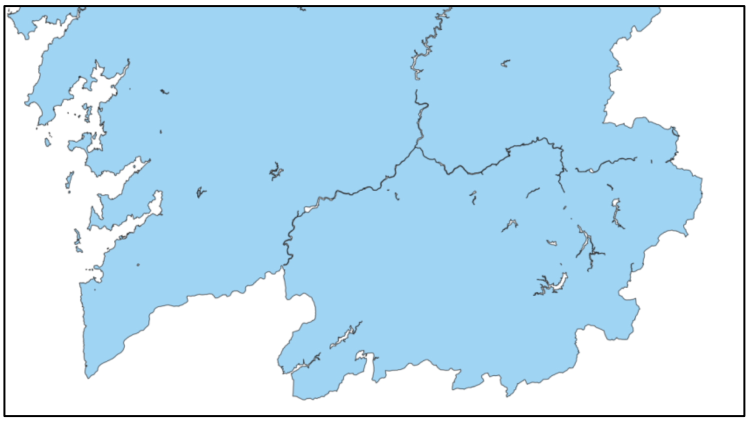
Shape and Geojson

A shapefile and a geojson file are included inside the folder “shape”. They are used by the system to determine which area must be downloaded (geojson file) and which area must be clipped from the rasters to avoid processing unnecessary parts. Geojson file was already mentioned on chapter 6.

The shapefile for this project corresponds to the Galician Autonomous Community, and was made by merging the four provinces from the shapefile provided by CNIG[61]. Then, using the shape of water masses provided by Xunta de Galicia[62], those areas where subtracted, to avoid false positives. A comparison between the original shapefile and the shapefile modified can be seen in the images 20 and 21.



19 - Original Shapefile



20 - Modified Shapefile

61 http://centrodedescargas.cnig.es/CentroDescargas/equipamiento.do?method=mostrarEquipamiento

62 http://mapas.xunta.gal/produtos-cartograficos/capas-six/hidrografia

Backend Tier: Implementation

Admin Script

Implementation of Admin Script was made in python. It uses datetime and os packages. It clears the screen and prints a menu on the screen and waits for user input. Those are the clear screen and menu code:

```
def cls():
    os.system('cls' if os.name=='nt' else 'clear')

def main_menu():
    print ('1 - Download Imagery')
    print ('2 - Order and unzip downloaded files')
    print ('3 - Utils - Purge imagery folder')
    print ('4 - Utils - Show folders content')
    print ('5 - Processing - Clip rasters')
    print ('6 - Processing - Calculate NBR index')
    print ('7 - Processing - Calculate difference between NBR indexes')
    print ('8 - Processing - Vectorize NBR an NBR difference')
    print ('9 - Upload vector files to PostGIS')
    print ('')
    print ('10 - Batch Processing')
    print ('')
    print ('')
    print ('0 - Exit')
    print ('')
    print ('Select an option')
```

Note that depending on the operative system, the code is slightly different.

As an illustration of how it works, the code for option 5 is shown:

```
    if user_choice == '5':
        system_directory=os.getcwd()
        imagery_directory=os.path.join(system_directory, r'imagery')
        imagery_directory_content=os.listdir(imagery_directory)
        print ('Clip raster files')
        print ('Imagery folder content:')
        print ('')
        unzip_script.print_tree()
        print ('')
        print ('Choose folder. Enter 0 to process every folder not yet processed')
        folder_chosen=input()
        while True:
            if folder_chosen in imagery_directory_content:
                print('')
                print ('Starting clip. This can take up to 10 minutes per file. Be
patient...')
                print ('')
                processing_script.clip(folder_chosen)
                break
            if folder_chosen == '0':
                print('')
                print ('Starting clip. This can take up to 10 minutes per file. Be
patient...')
                print ('')
                for folders in imagery_directory_content:
                    processing_script.clip(folders)
                break
            else:
                print('')
                print('Unknown folder, please try again. Available folders:')
                print ('')
                unzip_script.print_tree()
                print ('')
                print ('Choose folder. Enter 0 to process every folder not yet
processed')
                folder_chosen=input()
```

As can be seen, it check the existence of the folder before calling the corresponding function. The same case happens when working with dates, as illustrated on the following code extract from option 1:

```
1        print ('Enter Year')
2        year_=input()
3        while True:
4            try:
5                year_=year_.zfill(4)
6                year_test=int(year_)
7               if 1900>year_test or int(datetime.datetime.today().year) < year_test:
8                    raise CustomError('Please insert a valid year')
9                break
10            except:
11                print ('Enter Year')
12                year_=input()
```

First, it forces four digits by filling with zeros if necessary (line 5). Then it checks if the given year is valid (i.e. no future dates and no dates before 1900).

If any of the conditions is not met, the user is required to input data again.

Called functions are in other files, included at the beginning of the code:

```
#Admin script

import os
import download_script
import unzip_script
import datetime
import processing_script
```

Download Script

The download_script.py is a script that gets as arguments the imagery date, the cloud covering percentage and other options, and calls the Sentinelsat script to perform the file searching for a given area.
It checks the integrity of the data entered by the user (e.g no future dates, etc.), and prepares the sentinelsat arguments.
User and data information is taken from the file "apihub.txt" in the main folder. Geojson area mask is taken from the file "mask.geojson" in the folder "shape".
All files are downloaded to the main directory.

```
#Download Script. Carlos Sarille 2018


import datetime
```

```
5  import os
6
7  system_directory=os.getcwd()
8  auth_file_path=os.path.join(system_directory, r'apihub.txt')
9  mask_folder_path=os.path.join(system_directory, r'shape')
10 mask_file_path=os.path.join(mask_folder_path, r'mask.geojson')
11
12
13 class CustomError(Exception):
14    pass
15
16
17
18 def download(dt, clouds, down_files):
19
20     try:
21         auth_file=open(auth_file_path, 'r')
22         username=auth_file.readline().strip('\n')
23         password=auth_file.readline().strip('\n')
24         auth_file.close()
25     except:
26           raise CustomError('No apihub.txt file in main folder or corrupted file.
Format: 1st line username, 2nd line password.')
27
28     datetime_string_format = '%Y%m%d'
29
30     dt_ant = dt - datetime.timedelta(days=1)
31     dt_ant_str = dt_ant.date().strftime(datetime_string_format)
32     dt_str=dt.date().strftime(datetime_string_format)
33
34
35    #check for clouds
36     cloud_index=int(clouds)
37     if (cloud_index < 0) or (cloud_index > 100):
38         raise CustomError('Cloud coverage index must be between 0 and 100')
39     if (cloud_index==100):
40         clouds_=""
41     else:
42         clouds_=" --cloud "+str(cloud_index).zfill(2)
43     if (down_files is False):
44         lista=""
45     else:
46         lista="-d "
```

On lines 21 to 27 the script reads the file with username and password. Lines from 28 to 33 are to adapt date format, and to ensure that only a 1 day period is downloaded each time. Lines 35 to 42 check validity of clouds percentage and adapt the format. Finally, if the user doesn't want to download files, command "-d" is appended on lines 43 to 46.

Then, SentinelSat is called with the data collected:

```
sentinel_command='sentinelsat  -u '+username+' -p '+password+' -g
  "'+mask_file_path+'" -s '+dt_ant_str+' -e '+dt_str+clouds_+' --sentinel 2 –
  producttype S2MSI1C '+lista
print ("Executing:", sentinel_command)
os.system(sentinel_command)

print ('')
print ('Download finished')
```

On line 1, sentinelSat call is prepared, and on line 3 the system is ordered to execute the sentinel call.

Unzip script

The unzip_script.py file provides four different functions, order(), unzip(), print_tree() and purge_folders().

Order()

Order() checks the main directory looking for.zip files, which are the type of files provided by the European Space Agency. If the imagery folder already exists, it moves all the files inside. Otherwise, it creates the folder and then moves the files in.
The aim of this script is to keep the main directory clean.

```
def order():
    print ('')
    print ('Ordering Directories')
    print ('')


    print('Current system directory: ', str(system_directory))
    current_directory_content=os.listdir(system_directory)

    #creates imagery directory if it doesn't exist
    if not os.path.exists(imagery_directory):
        print ('Directory "imagery" has been created')
        os.makedirs(imagery_directory)
    else:
        print('The folder',str(imagery_directory),' already exists')

    print('')
    print('The following files have been moved to /imagery:')
    for name in current_directory_content:
        name_chunks=name.split('.')
        if name_chunks[-1] == 'zip':
            print (name)
            current_file=os.path.join(system_directory,name)
            moved_file=os.path.join(imagery_directory,name)
            os.rename(current_file,moved_file)
```

Unzip()

Unzip() is a function that, for one compressed file at a time, creates a temporary folder to unzip the files, creates a folder for the file date, and moves the desired files from the temporary folder to the file date folder. Normally these files are the Sentinel Bands of interest (e.g B08 and B12 to calculate the NBR).
The data about the tile name, dates, instrument and others, is extracted from the file name, which follows a standard naming convention[63], making it easier to automate this part.
After extracting the files of interest, it deletes the temporary folder and the original.zip file and starts again with the next imagery compressed file.
When the script finishes, only the files of interest are in the system.

[63] https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/naming-convention

In linux systems, the function calls the unzip command, which is already included in the linux path.
In windows systems, the function is dependent on Winrar[64]. The computer needs to have it installed and it needs to be in the system path, as described in several places[65][66].

The following code extract shows how it uses the file name to detect zip files and create folders named with the correspondent date, and calls the external unzip software:

```
        for name in current_directory_content:
            name_format=name.split('.')
            if name_format[-1] == 'zip': #check if the file format is .zip

                name_parts=name_format[-2].split('_')
                name_date=name_parts[2].split('T')
                final_folder=os.path.join(imagery_directory,name_date[0])
                temp_folder=os.path.join(imagery_directory,r'temp')
                if not os.path.exists(final_folder):
                    print ('Directory',str(final_folder) ,'has been created')
                    os.makedirs(final_folder)
                if not os.path.exists(temp_folder):
                    print ('Directory',str(temp_folder) ,'has been created')
                    os.makedirs(temp_folder)

                zip_file=os.path.join(imagery_directory,name)
                unzip_path_linux=r"unzip "+str(zip_file)+" -d "+str(temp_folder)
                   unzip_path_windows=r'winrar  x  -ibck  -o+  "'+str(zip_file)+'" "'+str(temp_folder)+'" -INUL'
          os.system(unzip_path_windows if os.name=='nt' else unzip_path_linux)
```

In the following code fragment it can be seen how the system checks file names and select only the bands of interest and cloud masks:

```
          for path,subdir,files in os.walk(temp_folder):
              for name2 in files:
                  fileformat=name2.split('.')
                  if fileformat[-1] == 'jp2':
                      band_type=fileformat[-2].split('_')
                      if band_type[-1] == 'B08' or band_type[-1] == 'B12':
                          if os.path.isfile(os.path.join(final_folder,name2)):
                              os.remove(os.path.join(final_folder,name2))

              os.rename(os.path.join(path,name2),os.path.join(final_folder,name2))
                   if fileformat[-1] == 'gml':
                       band_type=fileformat[-2].split('_')
                       if band_type[-2] == 'CLOUDS':
                           new_name=str(name_parts[-2]+'_'+name_date[0]+'_'+name2)
                           if os.path.isfile(os.path.join(final_folder,new_name)):
                               os.remove(os.path.join(final_folder,new_name))
           os.rename(os.path.join(path,name2),os.path.join(final_folder,new_name))
```

[64] https://www.winrar.es/
[65] http://www.itprotoday.com/management-mobility/how-can-i-add-new-folder-my-system-path
[66] https://www.howtogeek.com/118594/how-to-edit-your-system-path-for-easy-command-line-access/

Purge_folders()

Purge_folders() provides a method to check if any folder inside the imagery folder is empty and delete it.

Print_tree()

Print_tree() provides a method to display the content of the imagery folder and its subfolders.

```
def print_tree():
    for root, dirs, files in os.walk(imagery_directory):
        level = root.replace(imagery_directory, '').count(os.sep)
        indent = ' ' * 4 * (level)
        print('{}{}/'.format(indent, os.path.basename(root)))
        subindent = ' ' * 4 * (level + 1)
        for f in files:
            print('{}{}'.format(subindent, f))
```

Processing script

Processing Script contains six different processing functions: clip(), get_tiles(), NBR(), difference_NBR(),vector() and load_postgis(). Most of them use GDAL tools.

Clip()

Clip() gets as an argument a date folder name (e.g. ‘20170810’). If the folder is not found inside the imagery folder, it throws an error message.
If the folder is found, for each sentinel .jp2 image, a clipped version is created, appending “_clip” to the file name. The original file is deleted.
The clip operation is performed using Gdal_warp. The output file is an unsigned Int16, and the no data value is 0.
The clip mask is taken from the folder “Shape” and it is called “mask.shp”.

The following code extract shows how GDAL is called:

```
if files_parts[-1]=='jp2' and files_status[-1]!='clip' and files_status[-1]!='NBR':
    input_file_path=os.path.join(date_folder_directory,str(files))
    output_file_name=str(files_parts[0])+"_clip.tif"
   output_file_path=os.path.join(date_folder_directory,output_file_name)
    print('Input file: '+input_file_path)
    print("Output_file: "+output_file_path)
    print('')
    gdal_call='gdalwarp -ot UInt16 -dstnodata 0 -of GTiff -tr 10.0 -10.0 -tap -cutline "'+str(mask_path)+'" "'+str(input_file_path)+'" "'+str(output_file_path)+'"'
    os.system(gdal_call)
    os.remove(input_file_path)
    print ("Clip done")
```

Get_tiles()

Get_tiles() gets as an argument a date folder name (e.g. ‘20170810’). If the folder is not found inside the imagery folder, it throws an error message.

If the folder is found, it scans every file inside and extracts their tile name. The function returns a list of tiles found in a given folder.
It is an auxiliary function used by other functions.

NBR()

NBR() gets as an argument a date folder name (e.g. ‘20170810’). If the folder is not found inside the imagery folder, it throws an error message.
If the folder is found, it searches the band 08 (Near infrared - 842nm) and band 12 (short-wavelength infrared – 2190 nm) files, and performs the calculous using Gdal_calc.py.

The operation to get the NBR[67] index is:

$$NBR = \frac{B08 - B12}{B08 + B12}$$

The output file is a geotiff Float32 type. Furthermore, if a cloud mask is found in the folder, the areas covered by clouds are removed from the output file using gdal_rasterize to give them a value of “no value”.

The following extract of code shows how the software searches the correct bands and calls gdal_calc and gdal_rasterize:

```
for files in date_folder_content:
    files_parts=files.split('.')
    files_status=files_parts[0].split('_')

    if files_status[-2]=='B08' and files_status[0]==tile_name and files_status[-1] ==
    'clip' and files_parts[-1]=='tif':
         B08=os.path.join(date_folder_directory, files)
         print ("B08 file: ",B08)
```

```
gdal_calc_call='gdal_calc.py --type="Float32" -A "'+str(B08)+'" -B "'+str(B12)+'" --
outfile="'+str(NBR_file_path)+'" --calc "(A.astype(float32)-B.astype(float32))
/(A.astype(float32)+B.astype(float32))"'
os.system(gdal_calc_call)
if CLOUDS!=None:
    gdal_call='gdal_rasterize -at -burn 3.40282e+38 -where "maskType=\'OPAQUE\'"
     "'+CLOUDS+'" "'+NBR_file_path+'"'
    print('Applying cloud mask.')
    os.system(gdal_call)
    print ('Output File: ',NBR_file_path)
```

Difference_NBR()

Difference_NBR() gets as arguments two date folder name (e.g. ‘20170810’). If the folders are not found inside the imagery folder, it throws an error message.
If both folders are specified, it gets a list of all the common tiles and calculates the NBR index difference[68] for those tiles.

[67] https://www.sentinel-hub.com/develop/documentation/eo_products/Sentinel2EOproducts
[68] http://gsp.humboldt.edu/olm_2015/Courses/GSP_216_Online/lesson5-1/NBR.html

If only the last folder is specified, and the first one has a value of 0, then the system searches the common tiles in the next more recent folder and calculates them. If there are still any tile not calculated, the system searches in the next more recent folder, over and over until a valid tile is found or all folders are checked.
The output file is a geotiff file, processed with gdal_calc.py.

Vector()

Vector() gets as argument a date folder name (e.g. ‘20170810’). If the folder is not found inside the imagery folder, it throws an error message.
This function transforms the NBR file and the NBR difference file in vectorial files.

The output NBR file is a shape file, with two columns:

-DN: integer type. Fire severity, from 1 to 4. Threshold can be adjusted in the script, currently a value between -0.15 and -0.2 in the NBR layer is 1 in the shapefile. -0.2 to -0.25 is 2, -0.25 to -0.3 is 3 and less than -0.3 is 4.
-Date: date type.

Values under the threshold are considered no data.

The output NBR difference file is a shape file, with three columns:

-DN: integer type. Fire severity, from 1 to 4. Threshold can be adjusted in the script, currently a value between 0.6 and 0.8 in the difference NBR layer is 1 in the shapefile. 0.8 to 1 is 2, 1 to 1.2 is 3 and more than 1.2 is 4. Also, cloud areas are assigned the value 200.
-Date from: date type. Oldest date in the NBR difference. Form
-Date to: date type. Newest date in the NBR difference.

Values under the threshold are considered no data.
Processing is done using gdal_polygonize.py, ogrinfo and gdal_calc.py

Load_postgis()

Load_postgis() uploads shapefiles to a postgis database. Database information (username, password, host and database name) is read from the file “postgis_data.txt”..

This function gets as argument a date folder name (e.g. ‘20170810’). If the folder is not found inside the imagery folder, it throws an error message.

This function leaves an empty file called “this_folder_was_uploaded.txt” as a beacon once a folder is uploaded to the database. If this file is found during the scan of the folders, the function won’t upload any data to the database.

The upload process is done using ogr2ogr, as can be seen in the following code extract:

```
if file_name_parts[-1]=='NBR' and file_name_parts[-2]=='DIF' and file_format=='shp':
            shape_file_path=os.path.join(date_folder_directory, files)
            print('')
            print('Current File: ',shape_file_path)
            ogr_call='ogr2ogr -append -f "PostgreSQL" PG:"host='+db_host+'
                 user='+username+' dbname='+db_name+' password='+password+'"
                 "'+shape_file_path+'" -nln NBR_DIFF'
            os.system(ogr_call)
            f= open(uploaded_file_path,"w+")  #leave beacon file
            f.close()
            print('Database uploaded')
```

Middle Tier

Middle Tier implementation was just installing and configuring PostGIS.

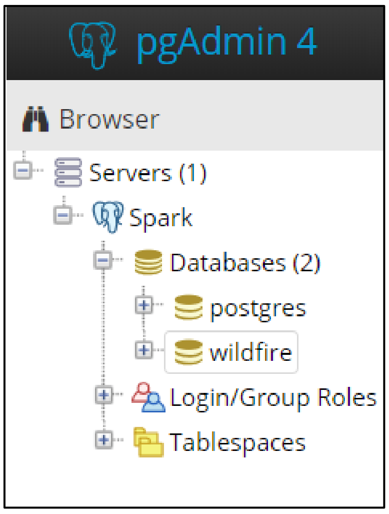
PostGis

PostGIS installation in linux is done by typing the following commands:

```
sudo apt-get install postgresql postgis
```

In windows systems it can be installed as a part of the OsGEO package[69].

Setup and configuration is easily done with the software pgAdmin[70]. Once PostGIS was running, a database called “wildfire” was created, as seen on image 22.



21 - Database detail

[69] https://postgis.net/windows_downloads/
[70] https://www.pgadmin.org/

PostGIS has to be configured to accept remote connections modifying the files postgresql.conf and pg_hba.conf files[71].

Also, on the database, the extension PostGIS must be enabled.

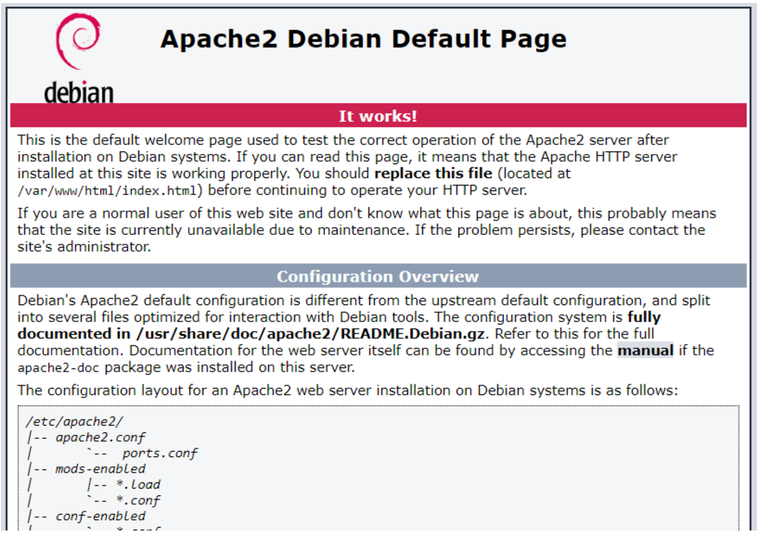
Presentation Tier

Apache

The installation from linux terminal is pretty straight forward, because it is included in the OS repositories, and it only involves the following commands[72]:

```
sudo apt-get install apache2 -y
```

If the installation was successful, the webpage shown on image 23 will appear when accessing to the server via web browser.



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
```

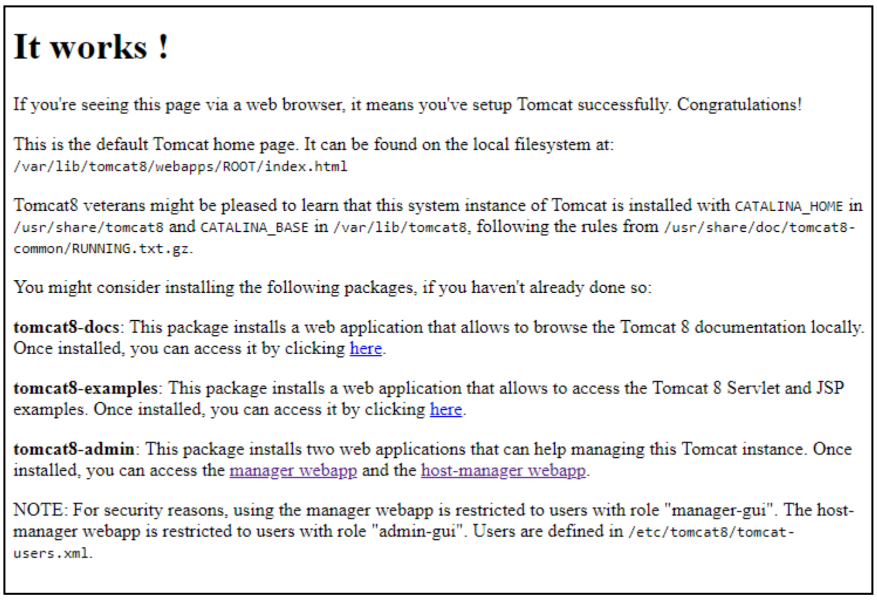
22 - Apache default web page

Apache Tomcat

As in the previous case, installation was done via terminal. As Tomcat relies on Java Runtime Environment[73], it needs to be installed as well. Depending on the desired versions, they are included in the repositories or not.

The installation process is not complicated[74], and once it is done, the webpage shown on image 24 will appear when accesing the server port 8080 via web browser:

[71] https://gis.stackexchange.com/questions/174004/remote-connection-to-postgis
[72] https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md
[73] http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
[74] https://wolfpaulus.com/java/tomcat-jessie/


It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat8/webapps/ROOT/index.html`

Tomcat8 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat8` and `CATALINA_BASE` in `/var/lib/tomcat8`, following the rules from `/usr/share/doc/tomcat8-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

tomcat8-docs: This package installs a web application that allows to browse the Tomcat 8 documentation locally. Once installed, you can access it by clicking here.

tomcat8-examples: This package installs a web application that allows to access the Tomcat 8 Servlet and JSP examples. Once installed, you can access it by clicking here.

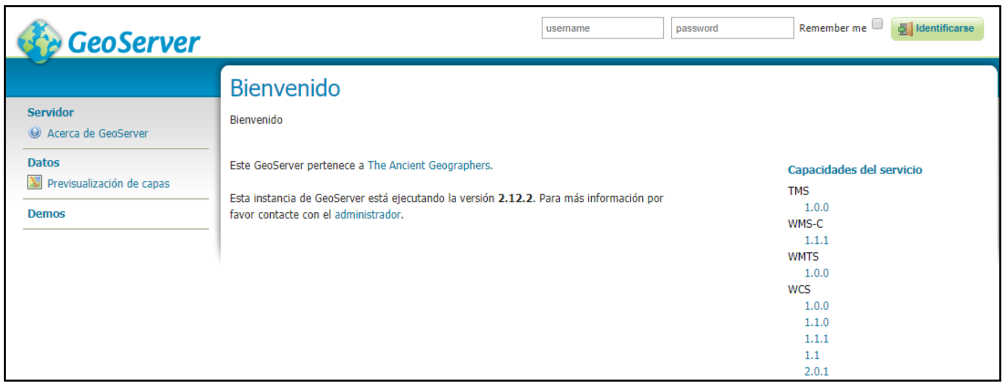
tomcat8-admin: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the manager webapp and the host-manager webapp.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat8/tomcat-users.xml`.

23 - Tomcat default web page

GeoServer

GeoServer runs on top of Apache Tomcat. Installation is done by downloading the .war file[75] and deploying it in the Tomcat webapps folder[76]. After reseting the Apache Tomcat service, if everything worked out, access to GeoServer control panel can be done via web browser in *hostaddres:8080/geoserver*, which will display something like image 25:



Bienvenido

Bienvenido

Este GeoServer pertenece a The Ancient Geographers.

Esta instancia de GeoServer está ejecutando la versión **2.12.2**. Para más información por favor contacte con el administrador.

Capacidades del servicio
TMS
1.0.0
WMS-C
1.1.1
WMTS
1.0.0
WCS
1.0.0
1.1.0
1.1.1
1.1
2.0.1

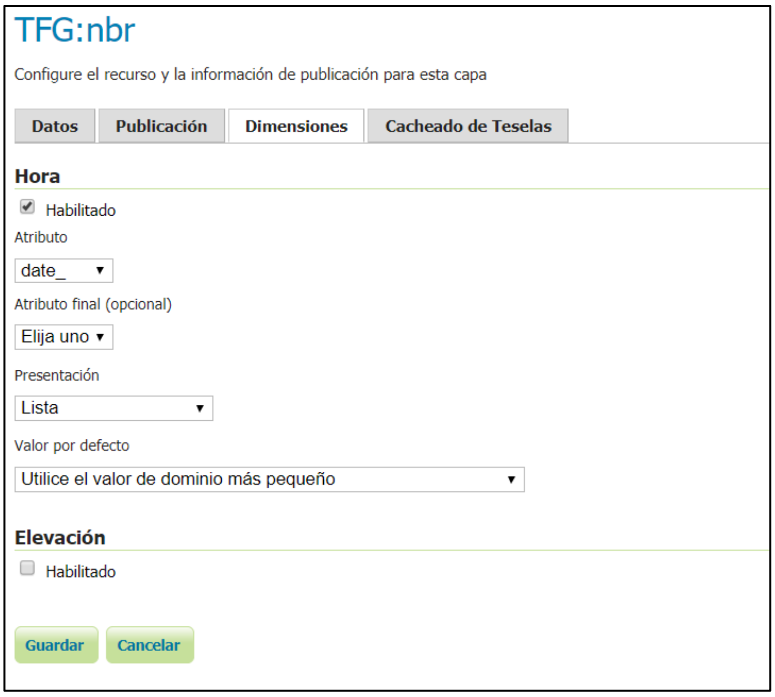
24 - Geoserver default web page

In order to allow external (i.e. non localhost connections), the CORS[77] header must be activated in Geoserver, modifying the file web.xml. The process is well documented in the Geoserver[78] manual as well as in several sites[79].

[75] http://geoserver.org/release/stable/
[76] https://geoserver.geo-solutions.it/edu/en/install_run/gs_install.html
[77] https://developer.mozilla.org/es/docs/Web/HTTP/Access_control_CORS
[78] http://docs.geoserver.org/latest/en/user/production/container.html
[79] https://gis.stackexchange.com/questions/210109/enabling-cors-in-geoserver-jetty

Once installed, a workspace must be created. In this case the name for the workspace was “TFG”, and WMS, WMTS must be enabled.

A new storage must be created, selecting PostGIS database and entering the data of our PostGIS database. Once it is connected, a new layer can be created. When creating the layer, time dimension must be enabled, as seen in the image 26.


TFG:nbr

Configure el recurso y la información de publicación para esta capa

Hora

[✓] Habilitado

Atributo: date_

Atributo final (opcional): Elija uno

Presentación: Lista

Valor por defecto: Utilice el valor de dominio más pequeño

Elevación

[ ] Habilitado

25 - Geoserver Time Domain

Also, a style must be given to each layer. Styles can be set up in QGIS, exported as SLD files and uploaded to Geoserver; or can be chosen from a list or typed directly on Geoserver. As an annex both styles for NBR and NBR difference layers can be seen.

Once the layer is published, it can be requested by web map services.

Leaflet

Leaflet[80] is a JavaScript library to create interactive web maps. Its installation is done by linking the .css and the .js files in the html map file.

```
<link rel="stylesheet" href="lib/leaflet.css" />
<script src="lib/leaflet.js"></script>
```

[80] http://leafletjs.com/

Web map

The web map is based on Leaflet, and it is made of a HTML file and a Javascript file, which contains the code for the interactive web map.

WMS layers are requested from Geoserver, as well as base layers from external sources like Bing maps.

Several Leflet plugins were used:

- Minimap[81], which allows showing a mini map.
- Leaflet Time Dimension[82], which enables Leaflet to work with temporal series of data
- Context Menu[83], which adds a customizable context menu to the map.
- Opacity Controls[84], which allows the user to select layer opacity.

The following code illustrates the map creation on the JS file.

```
//map creation
var map = L.map('map',{
         contextmenu: true,
      contextmenuWidth: 200,
         contextmenuItems: [{
           text: 'How to get here',
           callback: como_chegar
  }
       ,  '-',{
           text: 'Center here',
           callback: centerMap
       }, {
           text: 'Zoom in',
           callback: zoomIn
       }, {
           text: 'Zoom out',
           callback: zoomOut
       }, '-',{
      text: 'Restart Webmap',
      callback: reiniciar_visor
  }],
    timeDimension: true,
    timeDimensionOptions: {
        timeInterval: interval,
        period: "P1D"
    },
    timeDimensionControl: true,
}).setView([42.8 , -8] , 8)
```

The following code illustrates how a base layer is added to the map.

```
var options_bing = {
  bingMapsKey: 'ApWc1GvXN7a4LGPyvpolzQohUqHSDlbv4598PxCv_djXjfO5lb2YnJIWppQx1x75',
```

[81] https://github.com/Norkart/Leaflet-MiniMap
[82] https://github.com/socib/Leaflet.TimeDimension
[83] https://github.com/aratcliffe/Leaflet.contextmenu
[84] https://github.com/lizardtechblog/Leaflet.OpacityControls

```
  imagerySet: 'Aerial',
  culture: 'gl'
  }

var baselayer = L.tileLayer.bing(options_bing).addTo(map);
```

The following code illustrates how the minimap is added to the map.

```
//minimap
var osm2 = new L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 18
  });
var  miniMap  =  new  L.Control.MiniMap(osm2,  {  toggleDisplay:  true  ,  position: 
'bottomright' , zoomLevelOffset: -4 , width: 250 , strings: {
                             hideText: 'Hide map',
                             showText: 'Show map'
                      }}).addTo(map);
```

Production Stage

Linode is a VPS with a Debian system running. Configuration is similar to Linux configuration described before, but it has to be done through SSL terminal connection with a software like Putty[85].

[85] https://putty.org/

9. Results

This chapter presents the final results of the project.

Backend Tier

Currently, the backend tier provides ten different options:

- Download imagery
- Order and unzip downloaded files
- Purge imagery folder
- Show folders content
- Clip rasters
- Calculate NBR index
- Calculate difference between NBR indexes
- Vectorize NBR index and NBR difference
- Upload vector files to postgis.
- Batch processing

To execute the admin script, user must use Terminal on linux systems or CMD on windows systems. After navigating to the folder containing the script, a python instance must be called by typing

```
py admin_script.py
```

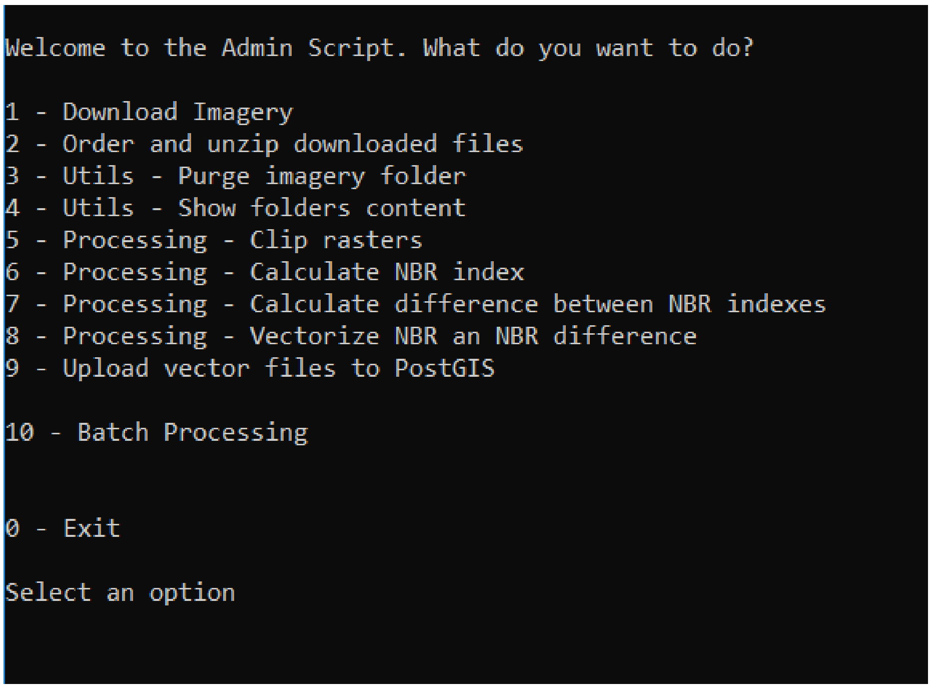
Or in some systems

```
python3 admin_script.py
```

Note that in order to execute this script in linux environments super user access must be provided:

```
sudo python3 admin_script.py
```

The image 27 shows the main menu.



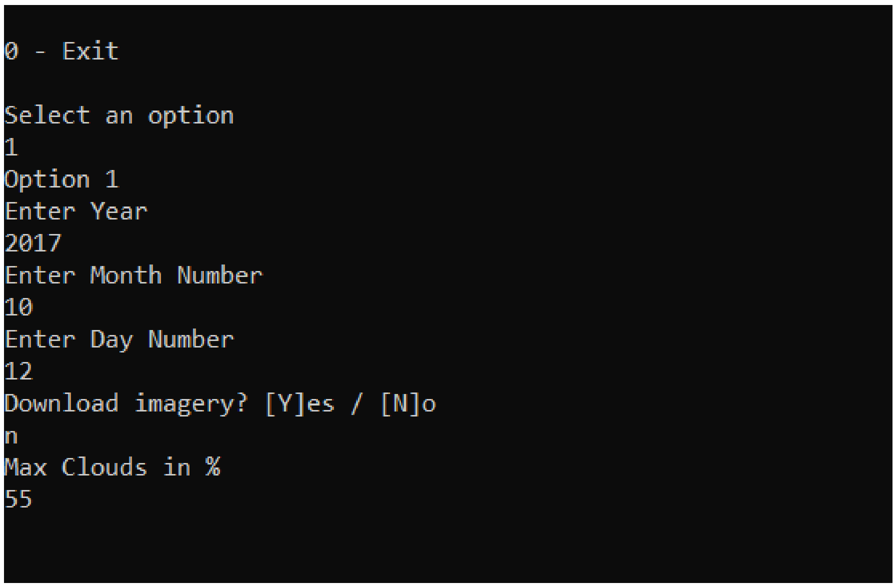
```
Welcome to the Admin Script. What do you want to do?

1 - Download Imagery
2 - Order and unzip downloaded files
3 - Utils - Purge imagery folder
4 - Utils - Show folders content
5 - Processing - Clip rasters
6 - Processing - Calculate NBR index
7 - Processing - Calculate difference between NBR indexes
8 - Processing - Vectorize NBR an NBR difference
9 - Upload vector files to PostGIS

10 - Batch Processing


0 - Exit

Select an option
```

26 - Admin Script Main Menu

1-Download Imagery

Download imagery requires the user to enter the desired date, cloud coverage percentage and a download yes/no option:

It performs checks to ensure the entered date is valid (i.e. non future or inexistent dates, no negative values, etc.) and calls the download function from the download_script.py file.



```
0 - Exit

Select an option
1
Option 1
Enter Year
2017
Enter Month Number
10
Enter Day Number
12
Download imagery? [Y]es / [N]o
n
Max Clouds in %
55
```

27 - Download Imagery Option

Order and unzip downloaded files

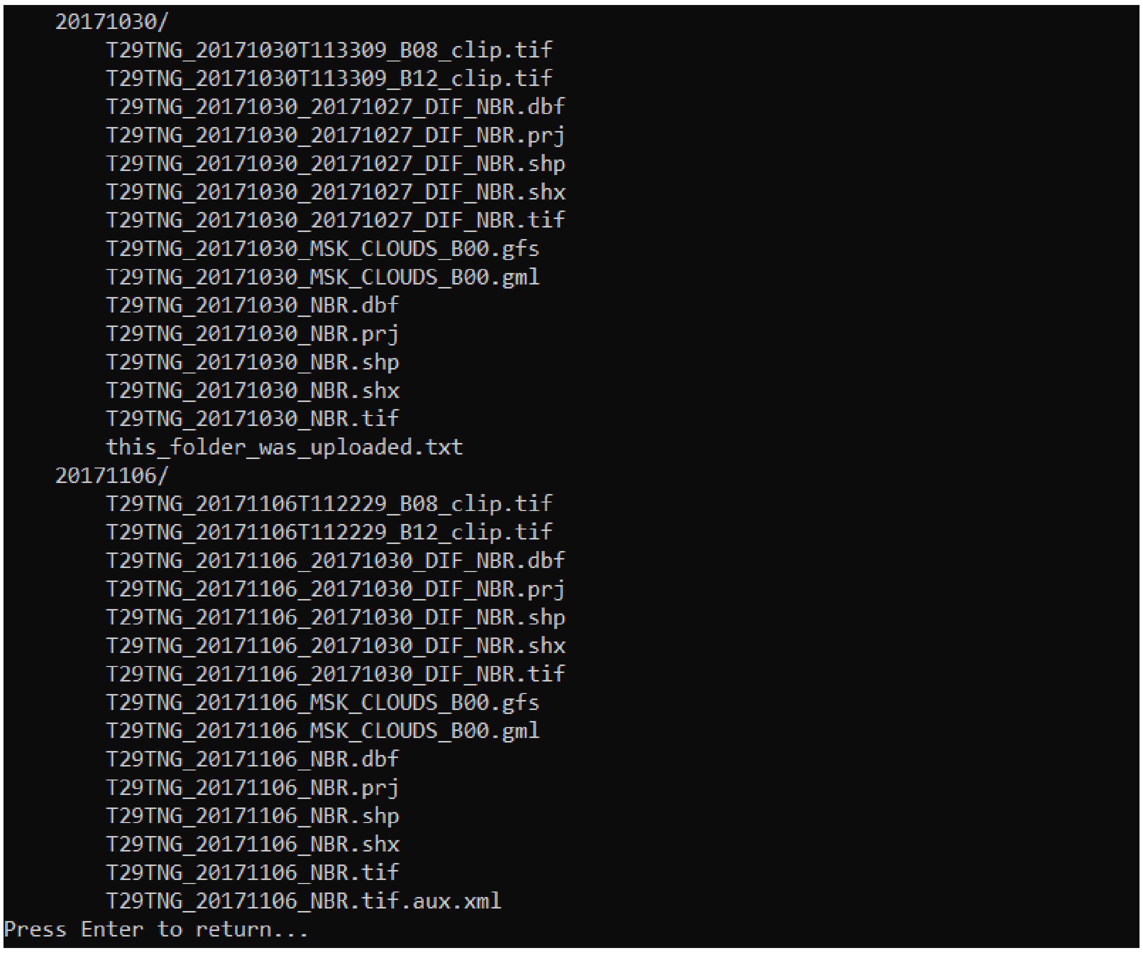
This option calls the functions order() and unzip() from the file unzip_script.py.

Purge imagery content

This option calls the function purge_folders() from the file unzip_script.py.

Show folder content

This option prints a tree with the folder structure, as can be seen on image 29.



```
    20171030/
        T29TNG_20171030T113309_B08_clip.tif
        T29TNG_20171030T113309_B12_clip.tif
        T29TNG_20171030_20171027_DIF_NBR.dbf
        T29TNG_20171030_20171027_DIF_NBR.prj
        T29TNG_20171030_20171027_DIF_NBR.shp
        T29TNG_20171030_20171027_DIF_NBR.shx
        T29TNG_20171030_20171027_DIF_NBR.tif
        T29TNG_20171030_MSK_CLOUDS_B00.gfs
        T29TNG_20171030_MSK_CLOUDS_B00.gml
        T29TNG_20171030_NBR.dbf
        T29TNG_20171030_NBR.prj
        T29TNG_20171030_NBR.shp
        T29TNG_20171030_NBR.shx
        T29TNG_20171030_NBR.tif
        this_folder_was_uploaded.txt
    20171106/
        T29TNG_20171106T112229_B08_clip.tif
        T29TNG_20171106T112229_B12_clip.tif
        T29TNG_20171106_20171030_DIF_NBR.dbf
        T29TNG_20171106_20171030_DIF_NBR.prj
        T29TNG_20171106_20171030_DIF_NBR.shp
        T29TNG_20171106_20171030_DIF_NBR.shx
        T29TNG_20171106_20171030_DIF_NBR.tif
        T29TNG_20171106_MSK_CLOUDS_B00.gfs
        T29TNG_20171106_MSK_CLOUDS_B00.gml
        T29TNG_20171106_NBR.dbf
        T29TNG_20171106_NBR.prj
        T29TNG_20171106_NBR.shp
        T29TNG_20171106_NBR.shx
        T29TNG_20171106_NBR.tif
        T29TNG_20171106_NBR.tif.aux.xml
Press Enter to return...
```

28 - Show Folder Content Option

Clip rasters

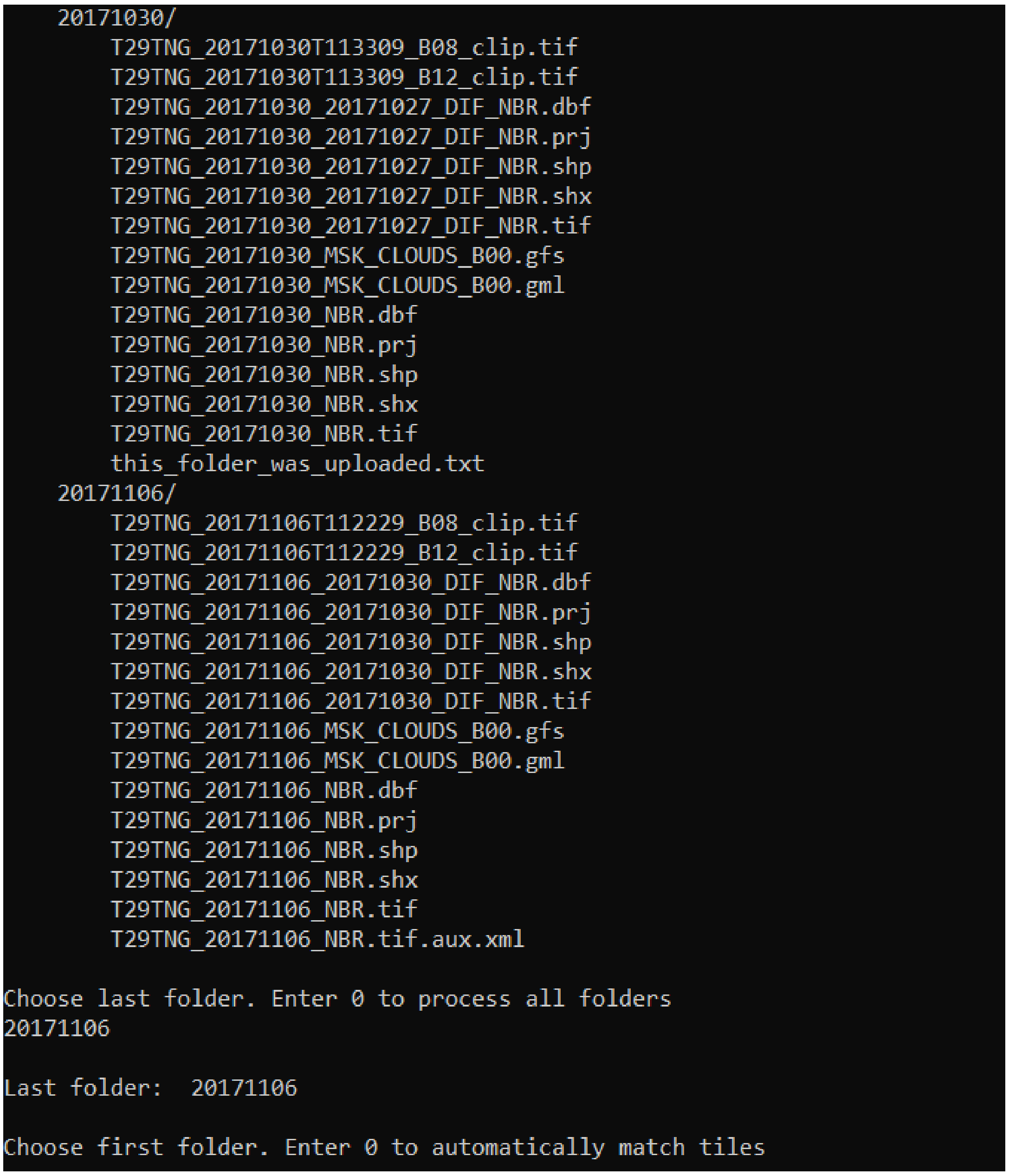
This option displays the content of the imagery folder and requires the user to enter a folder to process. Alternatively it can check which folders haven’t been clipped and clip them automatically.

Calculate NBR Index

This option displays the content of the imagery folder and requires the user to enter a folder to process. Alternatively it can check which folders don't have NBR files and calculate them automatically.

Calculate difference between NBR indexes

This option displays the content of the imagery folder and requires the user to enter two folders to calculate the difference between the NBR index of their files. Alternatively only the most recent folder can be provided by the user and the system will automatically search matching NBR tiles in past folders starting with the closest one.



```
    20171030/
        T29TNG_20171030T113309_B08_clip.tif
        T29TNG_20171030T113309_B12_clip.tif
        T29TNG_20171030_20171027_DIF_NBR.dbf
        T29TNG_20171030_20171027_DIF_NBR.prj
        T29TNG_20171030_20171027_DIF_NBR.shp
        T29TNG_20171030_20171027_DIF_NBR.shx
        T29TNG_20171030_20171027_DIF_NBR.tif
        T29TNG_20171030_MSK_CLOUDS_B00.gfs
        T29TNG_20171030_MSK_CLOUDS_B00.gml
        T29TNG_20171030_NBR.dbf
        T29TNG_20171030_NBR.prj
        T29TNG_20171030_NBR.shp
        T29TNG_20171030_NBR.shx
        T29TNG_20171030_NBR.tif
        this_folder_was_uploaded.txt
    20171106/
        T29TNG_20171106T112229_B08_clip.tif
        T29TNG_20171106T112229_B12_clip.tif
        T29TNG_20171106_20171030_DIF_NBR.dbf
        T29TNG_20171106_20171030_DIF_NBR.prj
        T29TNG_20171106_20171030_DIF_NBR.shp
        T29TNG_20171106_20171030_DIF_NBR.shx
        T29TNG_20171106_20171030_DIF_NBR.tif
        T29TNG_20171106_MSK_CLOUDS_B00.gfs
        T29TNG_20171106_MSK_CLOUDS_B00.gml
        T29TNG_20171106_NBR.dbf
        T29TNG_20171106_NBR.prj
        T29TNG_20171106_NBR.shp
        T29TNG_20171106_NBR.shx
        T29TNG_20171106_NBR.tif
        T29TNG_20171106_NBR.tif.aux.xml

Choose last folder. Enter 0 to process all folders
20171106

Last folder:  20171106

Choose first folder. Enter 0 to automatically match tiles
```

29 - Calculate Difference Option

Vectorize NBR and NBR difference

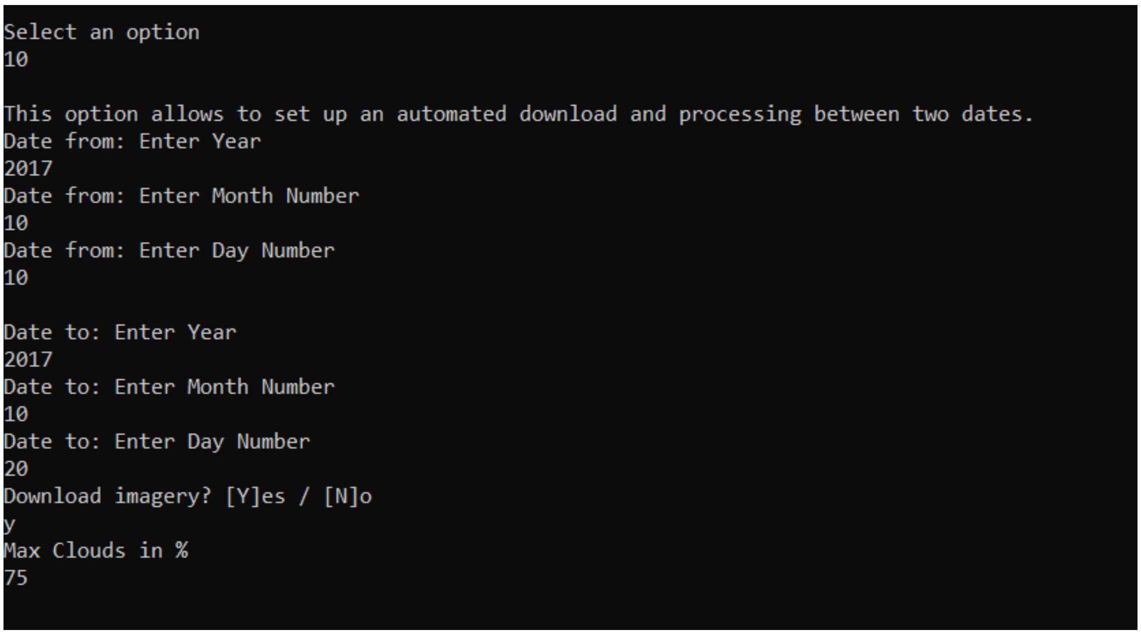
This option displays the content of the imagery folder and requires the user to enter a folder to process. Alternatively it can check which folders don't have ESRI shapefiles and calculate them automatically.

Upload files to PostGIS

This option displays the content of the imagery folder and requires the user to enter a folder to upload. Alternatively it can check which folders haven't been uploaded and upload them.

Batch Processing

This option allows the user to request all of the previous processes for a given period of time. It requires a start date and an end date, and parameters for downloading, as shown on image 31.



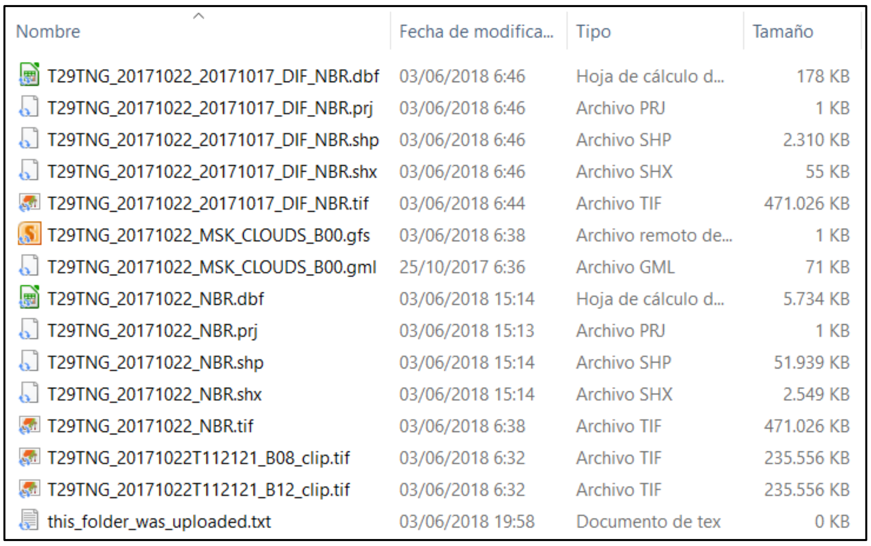
```
Select an option
10

This option allows to set up an automated download and processing between two dates.
Date from: Enter Year
2017
Date from: Enter Month Number
10
Date from: Enter Day Number
10

Date to: Enter Year
2017
Date to: Enter Month Number
10
Date to: Enter Day Number
20
Download imagery? [Y]es / [N]o
y
Max Clouds in %
75
```

30 - Batch Processing Option

Products Obtained

At the end of the process, aside from the files uploaded to the server, the following products are obtained for each tile, as can be seen on image 32:

- Clipped Sentinel Imagery: Bands B08 and B12
- Cloud Mask
- NBR Index raster file
- NBR Index vector file
- NBR difference raster file
- NBR difference vector file



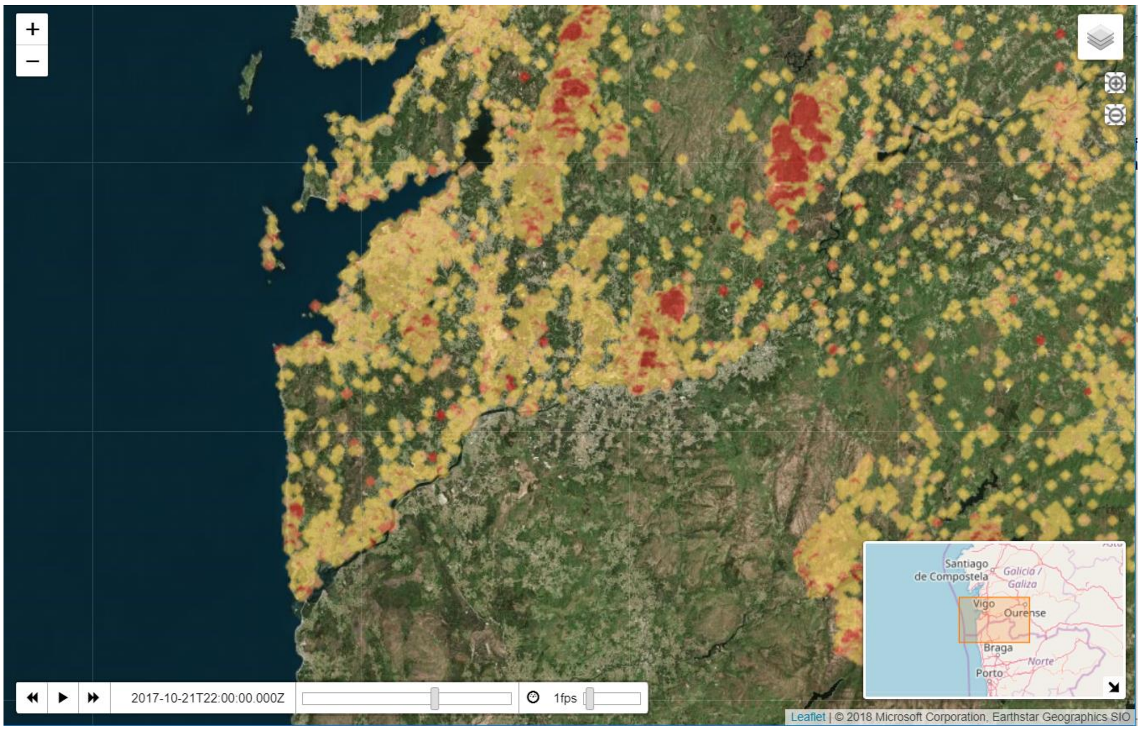
Nombre	Fecha de modifica...	Tipo	Tamaño
T29TNG_20171022_20171017_DIF_NBR.dbf	03/06/2018 6:46	Hoja de cálculo d...	178 KB
T29TNG_20171022_20171017_DIF_NBR.prj	03/06/2018 6:46	Archivo PRJ	1 KB
T29TNG_20171022_20171017_DIF_NBR.shp	03/06/2018 6:46	Archivo SHP	2.310 KB
T29TNG_20171022_20171017_DIF_NBR.shx	03/06/2018 6:46	Archivo SHX	55 KB
T29TNG_20171022_20171017_DIF_NBR.tif	03/06/2018 6:44	Archivo TIF	471.026 KB
T29TNG_20171022_MSK_CLOUDS_B00.gfs	03/06/2018 6:38	Archivo remoto de...	1 KB
T29TNG_20171022_MSK_CLOUDS_B00.gml	25/10/2017 6:36	Archivo GML	71 KB
T29TNG_20171022_NBR.dbf	03/06/2018 15:14	Hoja de cálculo d...	5.734 KB
T29TNG_20171022_NBR.prj	03/06/2018 15:13	Archivo PRJ	1 KB
T29TNG_20171022_NBR.shp	03/06/2018 15:14	Archivo SHP	51.939 KB
T29TNG_20171022_NBR.shx	03/06/2018 15:14	Archivo SHX	2.549 KB
T29TNG_20171022_NBR.tif	03/06/2018 6:38	Archivo TIF	471.026 KB
T29TNG_20171022T112121_B08_clip.tif	03/06/2018 6:32	Archivo TIF	235.556 KB
T29TNG_20171022T112121_B12_clip.tif	03/06/2018 6:32	Archivo TIF	235.556 KB
this_folder_was_uploaded.txt	03/06/2018 19:58	Documento de tex	0 KB

31 - Products Obtained

Presentation Tier

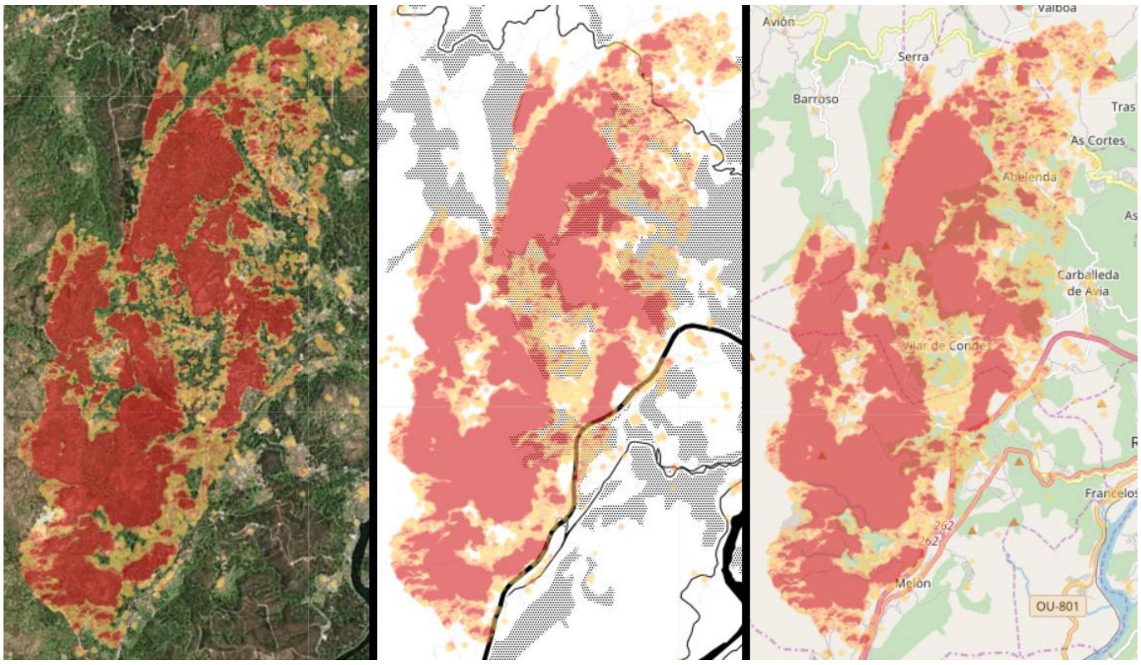
On the Presentation Tier, a web map is obtained, which can display three different base maps, and can overlay NBR index and NBR difference index layers, as well as a cadastral layer. Time navigation is done with a time slider.

It also has an opacity control for the NBR index layer, and a zoom control, as seen on image 33.



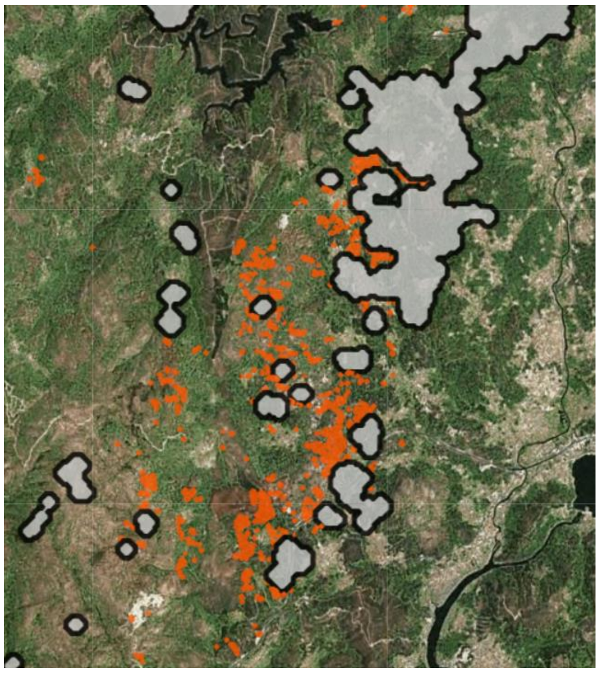
32 - Web Map main view

Image 34 shows a burned area with the NBR index superimposed and a satellite base layer on the left hand side, toner bas layer on the centre and map base layer on the right hand side, all three printed from the web map.



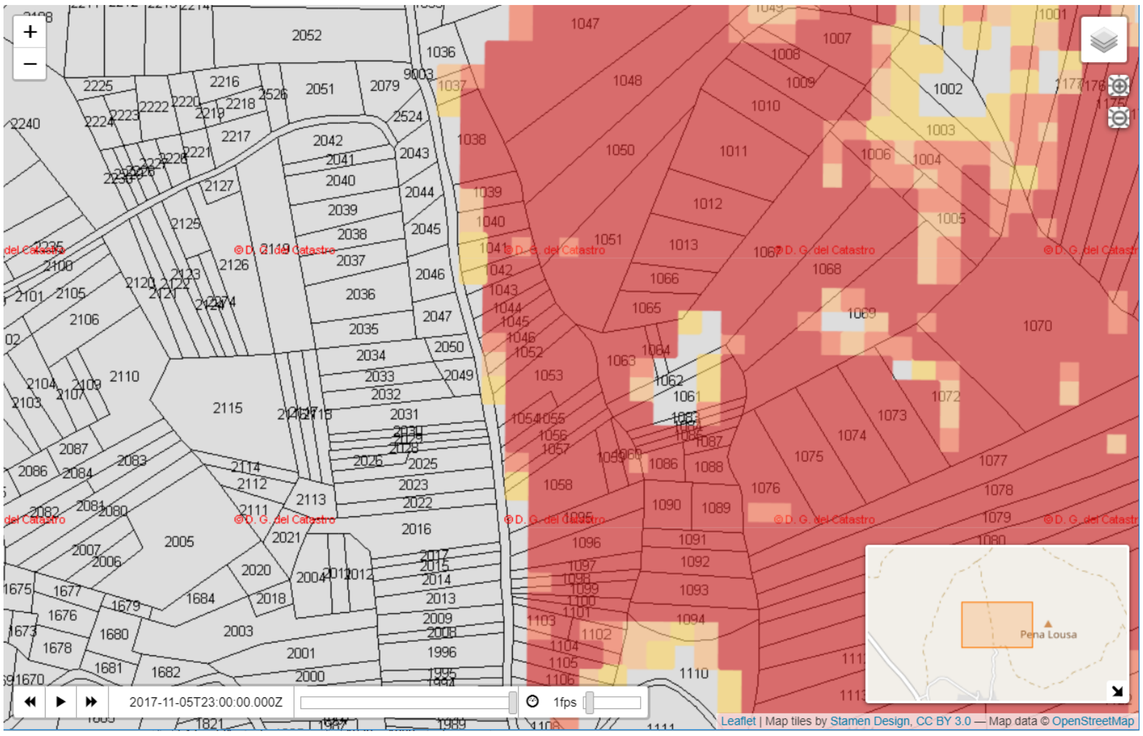
33 - Web Map Base Layer Options

Image 35 shows the NBR difference layer superimposed to the satellite base layer. Grey areas are clouds, so those parts were not processed.



34 - NBR difference Layer

The following image shows a map base layer, with NBR index and Cadastral parcels superimposed.


2017-11-05T23:00:00.000Z
1fps
Leaflet | Map tiles by Stamen Design, CC BY 3.0 — Map data © OpenStreetMap
© D. G. del Catastro
Pena Lousa

36 – Cadastral Parcels and NBR Index

A web map demo with a set of data covering one month time for the south region of Galicia is available in the following link:

http://178.79.139.171/TFG

10. Conclusions

To finish this Project, in this section some conclusions will be presented, as well as an analysis of learned lessons, problems found and future work lines.

Results

For the results of this project, checking the final status of the goals proposed, the following has been achieved:

- Automatable system for downloading and processing satellite images and cartography burned areas.
- Interactive Web Map that shows the information gathered.

With those two functionalities, the main goals of the project were achieved.
Also, some useful resources were created (e.g. Shapefiles of Galicia land mass); and with minimal modifications the project can suit for different areas or purposes.

Learned Lessons

Geographical Information Systems are a powerful tool to analyse, understand and improve our world, and they are still at the beginning of their potential.

By combining the data gathered by different sources as sensors, satellites, etc. and without need for expensive equipment, amazing achievements can be done.

As student, it was really interesting to research about image processing and geographical oriented programming.

I would like to highlight the easiness and good behaviour of python as programming language, although sometimes it was challenging to carry on.

Problems Found

Most of the risks predicted happened, especially the ones related to lack of time or lack of experience.

Another problem was cloud detection, which I didn’t have the time and resources to solve, and I think that is a key aspect or fully unsupervised burnt areas detection.

Furthermore, the volume of data in this project is enormous, reason why in some occasions testing features was really slow and unefficient, both processing and downloading or uploading.

Future work lines

As future work lines, I think there are three important aspects:

- Improve cloud detection using an fmask algorithm or similar.
- Improve web map user experience, by allowing data manipulation and calendar date selection, etc. Design aspects are important too.
- GEOBIA land analysis and classification would provide better results, as described on chapter 5.

11. Annexes

SLD style for NBR difference layer

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor                     xmlns="http://www.opengis.net/sld"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ogc="http://www.opengis.net/ogc"   xmlns:se="http://www.opengis.net/se"
xsi:schemaLocation="http://www.opengis.net/sld
http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd"
version="1.1.0">
  <NamedLayer>
    <se:Name>nbr</se:Name>
    <UserStyle>
      <se:Name>nbr</se:Name>
      <se:FeatureTypeStyle>
        <se:Rule>
          <se:Name>1</se:Name>
          <se:Description>
            <se:Title>1</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>dn</ogc:PropertyName>
              <ogc:Literal>1</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#ffd560</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#ffd560</se:SvgParameter>
              <se:SvgParameter name="stroke-width">4</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name>2</se:Name>
          <se:Description>
            <se:Title>2</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>dn</ogc:PropertyName>
              <ogc:Literal>2</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#ffbc7d</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#ffbc7d</se:SvgParameter>
              <se:SvgParameter name="stroke-width">4</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name>Clouds</se:Name>
          <se:Description>
            <se:Title>Clouds</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>dn</ogc:PropertyName>
              <ogc:Literal>200</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#c3c3c3</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#000000</se:SvgParameter>
              <se:SvgParameter name="stroke-width">4</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name>3</se:Name>
          <se:Description>
            <se:Title>3</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>dn</ogc:PropertyName>
              <ogc:Literal>3</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#fc7050</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#fc7050</se:SvgParameter>
              <se:SvgParameter name="stroke-width">4</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name>4</se:Name>
          <se:Description>
            <se:Title>4</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>dn</ogc:PropertyName>
              <ogc:Literal>4</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#d42020</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#d42020</se:SvgParameter>
              <se:SvgParameter name="stroke-width">4</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

SLD style for NBR difference layer

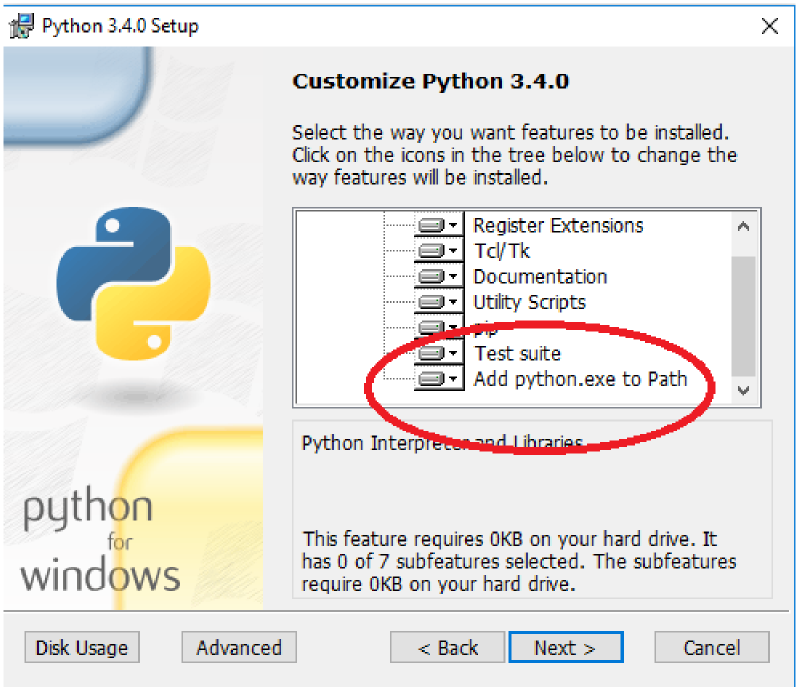
```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor                      xmlns="http://www.opengis.net/sld"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/sld
http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:se="http://www.opengis.net/se"   xmlns:ogc="http://www.opengis.net/ogc"
version="1.1.0">
  <NamedLayer>
    <se:Name>nbr</se:Name>
    <UserStyle>
      <se:Name>nbr</se:Name>
      <se:FeatureTypeStyle>
        <se:Rule>
          <se:Name>1</se:Name>
          <se:Description>
            <se:Title>1</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>DN</ogc:PropertyName>
              <ogc:Literal>1</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#ffd560</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#ffd560</se:SvgParameter>
              <se:SvgParameter name="stroke-width">1</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name>2</se:Name>
          <se:Description>
            <se:Title>2</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>DN</ogc:PropertyName>
              <ogc:Literal>2</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#ffbc7d</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#ffbc7d</se:SvgParameter>
              <se:SvgParameter name="stroke-width">1</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name>3</se:Name>
          <se:Description>
            <se:Title>3</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>DN</ogc:PropertyName>
              <ogc:Literal>3</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#fc7050</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#fc7050</se:SvgParameter>
              <se:SvgParameter name="stroke-width">1</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name>4</se:Name>
          <se:Description>
            <se:Title>4</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>DN</ogc:PropertyName>
              <ogc:Literal>4</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#d42020</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#d42020</se:SvgParameter>
              <se:SvgParameter name="stroke-width">1</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
        <se:Rule>
          <se:Name></se:Name>
          <se:Description>
            <se:Title>DN is ''</se:Title>
          </se:Description>
          <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>DN</ogc:PropertyName>
              <ogc:Literal></ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#67000d</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#232323</se:SvgParameter>
              <se:SvgParameter name="stroke-width">1</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

Windows 10 Installation

This annex will cover how to install the backend system on a Windows 10 machine.

1 Python

- Download Python 3.4[86]
- Install Python following the instructions.Make sure the “Add Python 3.4 to PATH” is checked, as seen in the image below.



Python 3.4.0 Setup

Customize Python 3.4.0

Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.

Register Extensions
Tcl/Tk
Documentation
Utility Scripts
pip
Test suite
Add python.exe to Path

Python Interpreter and Libraries

This feature requires 0KB on your hard drive. It has 0 of 7 subfeatures selected. The subfeatures require 0KB on your hard drive.

- Finally, to check the installation, open the Command Prompt and type

```
py --version
```

If the installation was successful, information about python version should appear.

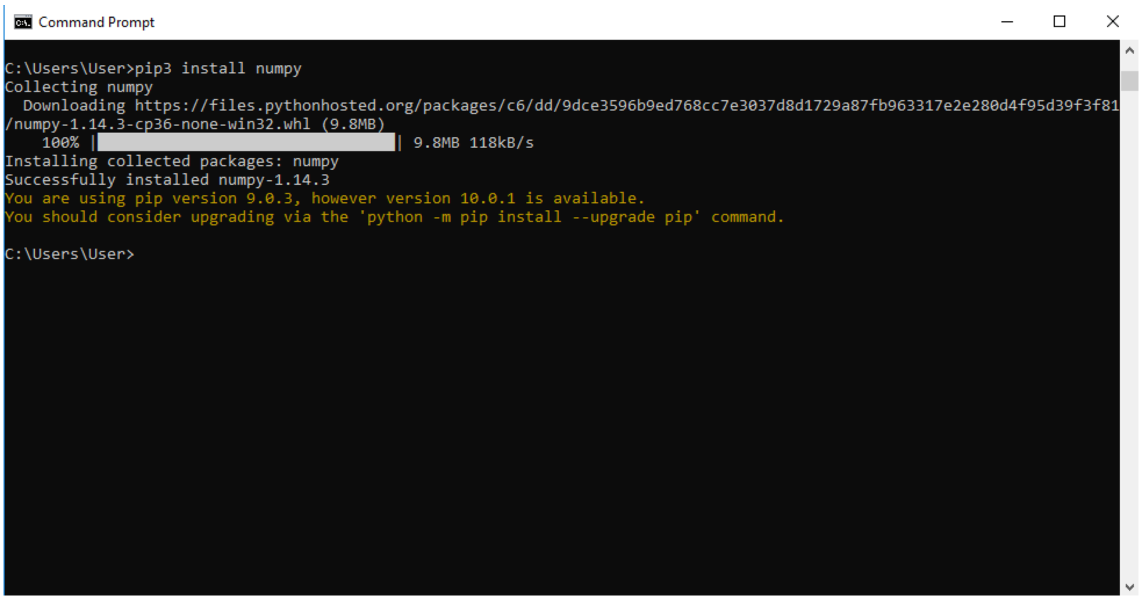
Numpy

- Open the Command Prompt and type

```
pip3 install numpy
```

86 https://www.python.org/downloads/release/python-340/

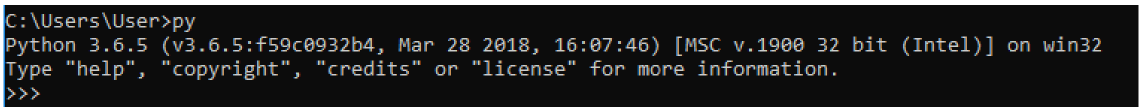
If everything goes right, something like the image below should appear.



```
C:\Users\User>pip3 install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/c6/dd/9dce3596b9ed768cc7e3037d8d1729a87fb963317e2e280d4f95d39f3f81
/numpy-1.14.3-cp36-none-win32.whl (9.8MB)
    100% |████████████████████████████████| 9.8MB 118kB/s
Installing collected packages: numpy
Successfully installed numpy-1.14.3
You are using pip version 9.0.3, however version 10.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\User>
```

GDAL

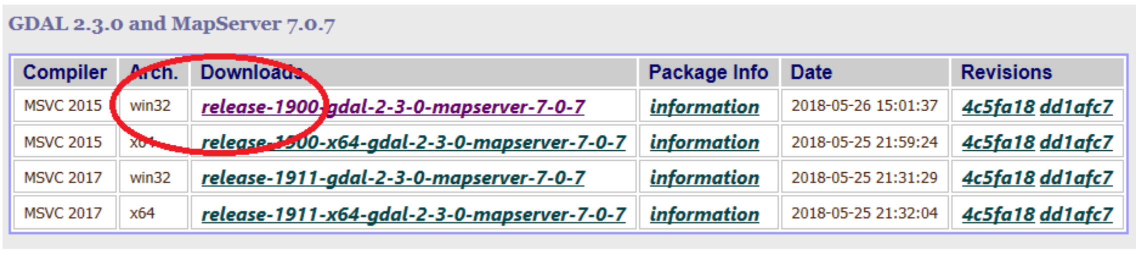
Installing GDAL on windows systems can be a bit tricky. First of all, by typing "py" in the Command Prompt, some information needs to be gathered from python.



```
C:\Users\User>py
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

As can be seen, there is a bracket with the letters "MSC v.1900". The number is needed to download the right GDAL compilation. Type "exit()" to exit python.

Once in the download page[87] select the adequate release for your system architecture and for your python version (the number inside brackets).

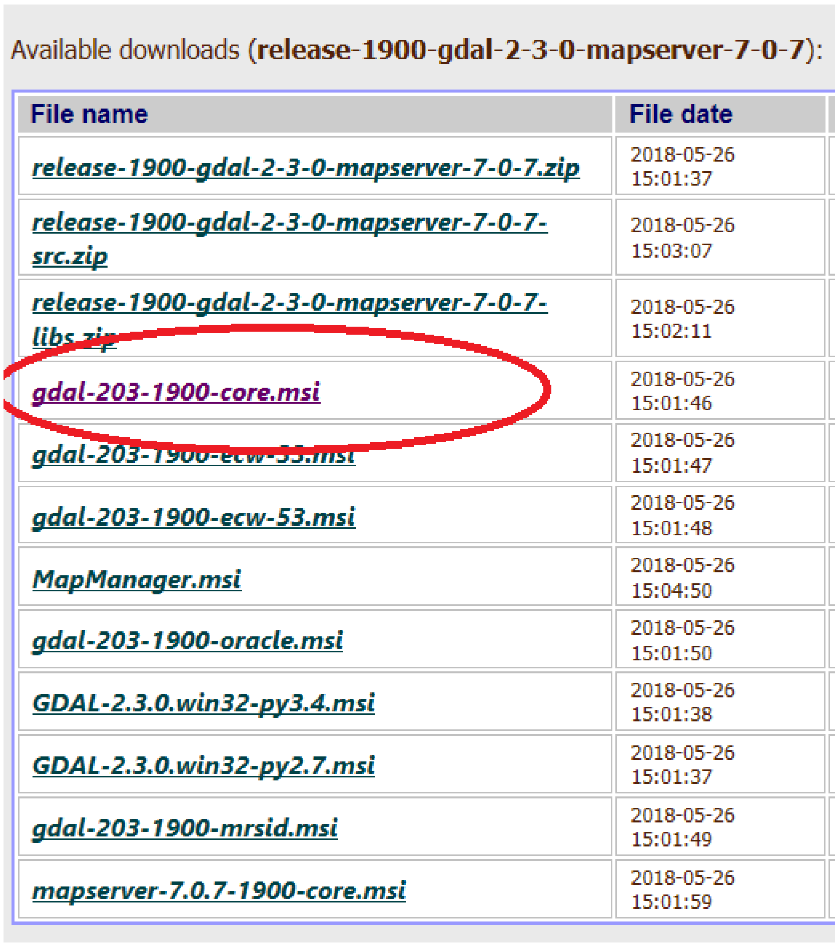


GDAL 2.3.0 and MapServer 7.0.7

Compiler	Arch.	Downloads	Package Info	Date	Revisions
MSVC 2015	win32	release-1900-gdal-2-3-0-mapserver-7-0-7	information	2018-05-26 15:01:37	4c5fa18 dd1afc7
MSVC 2015	x64	release-1900-x64-gdal-2-3-0-mapserver-7-0-7	information	2018-05-25 21:59:24	4c5fa18 dd1afc7
MSVC 2017	win32	release-1911-gdal-2-3-0-mapserver-7-0-7	information	2018-05-25 21:31:29	4c5fa18 dd1afc7
MSVC 2017	x64	release-1911-x64-gdal-2-3-0-mapserver-7-0-7	information	2018-05-25 21:32:04	4c5fa18 dd1afc7

Then select the package with GDAL core only, download it and install it selecting "Typical Installation".

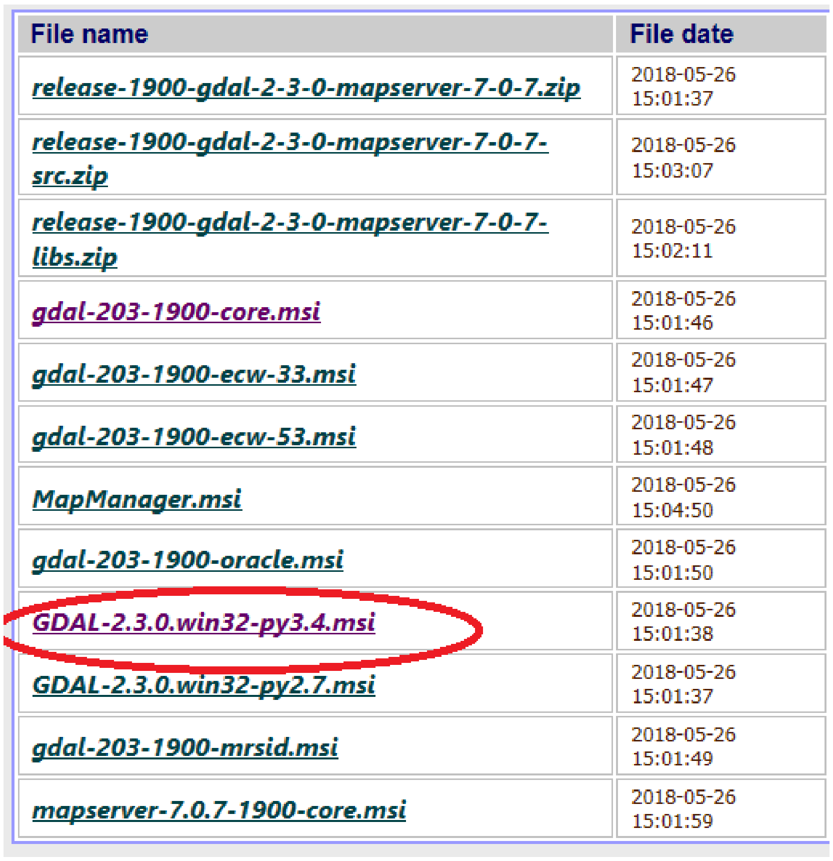
87 http://www.gisinternals.com/release.php



Available downloads (**release-1900-gdal-2-3-0-mapserver-7-0-7**):

File name	File date
release-1900-gdal-2-3-0-mapserver-7-0-7.zip	2018-05-26 15:01:37
release-1900-gdal-2-3-0-mapserver-7-0-7-src.zip	2018-05-26 15:03:07
release-1900-gdal-2-3-0-mapserver-7-0-7-libs.zip	2018-05-26 15:02:11
gdal-203-1900-core.msi	2018-05-26 15:01:46
gdal-203-1900-ecw-33.msi	2018-05-26 15:01:47
gdal-203-1900-ecw-53.msi	2018-05-26 15:01:48
MapManager.msi	2018-05-26 15:04:50
gdal-203-1900-oracle.msi	2018-05-26 15:01:50
GDAL-2.3.0.win32-py3.4.msi	2018-05-26 15:01:38
GDAL-2.3.0.win32-py2.7.msi	2018-05-26 15:01:37
gdal-203-1900-mrsid.msi	2018-05-26 15:01:49
mapserver-7.0.7-1900-core.msi	2018-05-26 15:01:59

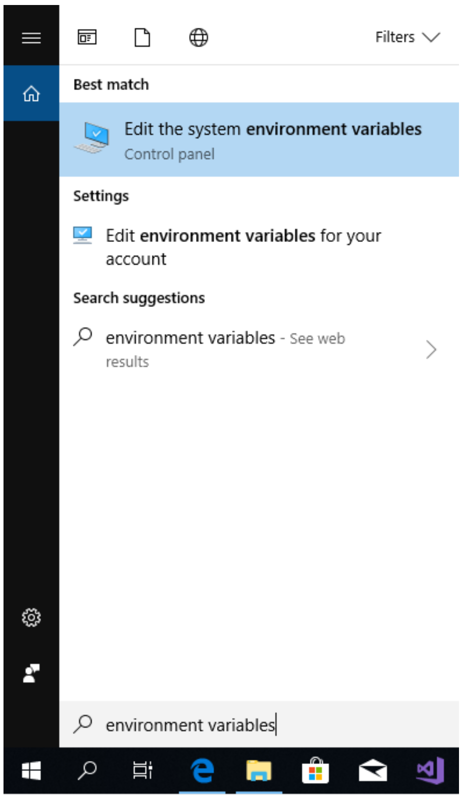
Once installed, download GDAL bindings for python:



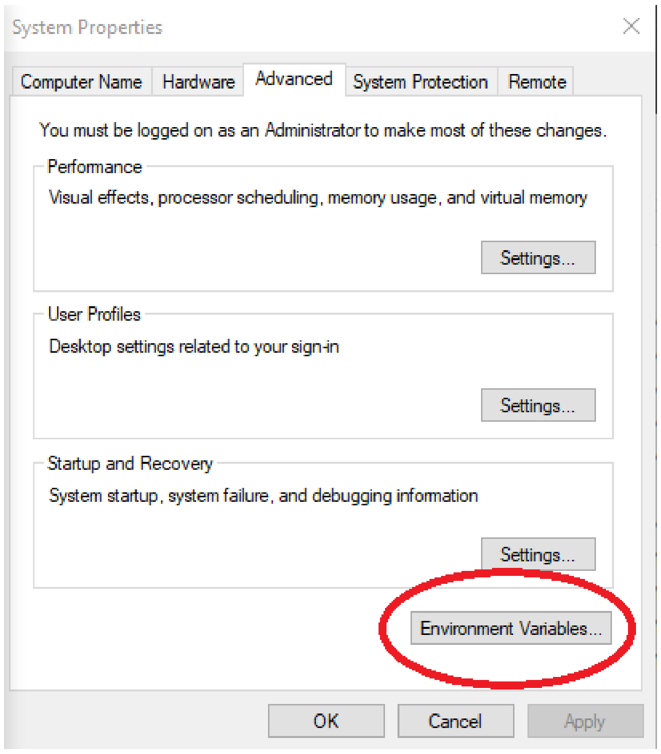
File name	File date
release-1900-gdal-2-3-0-mapserver-7-0-7.zip	2018-05-26 15:01:37
release-1900-gdal-2-3-0-mapserver-7-0-7-src.zip	2018-05-26 15:03:07
release-1900-gdal-2-3-0-mapserver-7-0-7-libs.zip	2018-05-26 15:02:11
gdal-203-1900-core.msi	2018-05-26 15:01:46
gdal-203-1900-ecw-33.msi	2018-05-26 15:01:47
gdal-203-1900-ecw-53.msi	2018-05-26 15:01:48
MapManager.msi	2018-05-26 15:04:50
gdal-203-1900-oracle.msi	2018-05-26 15:01:50
GDAL-2.3.0.win32-py3.4.msi	2018-05-26 15:01:38
GDAL-2.3.0.win32-py2.7.msi	2018-05-26 15:01:37
gdal-203-1900-mrsid.msi	2018-05-26 15:01:49
mapserver-7.0.7-1900-core.msi	2018-05-26 15:01:59

And install them.

Last step is to configure system Path. In order to do that, type environment variables on the windows search bar, as in the image below


Filters
Best match
Edit the system environment variables
Control panel
Settings
Edit environment variables for your account
Search suggestions
environment variables - See web results
environment variables

And click on Edit the system environment variables. A system properties windows will open. Click on Environment Variables as shown in the image below:


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

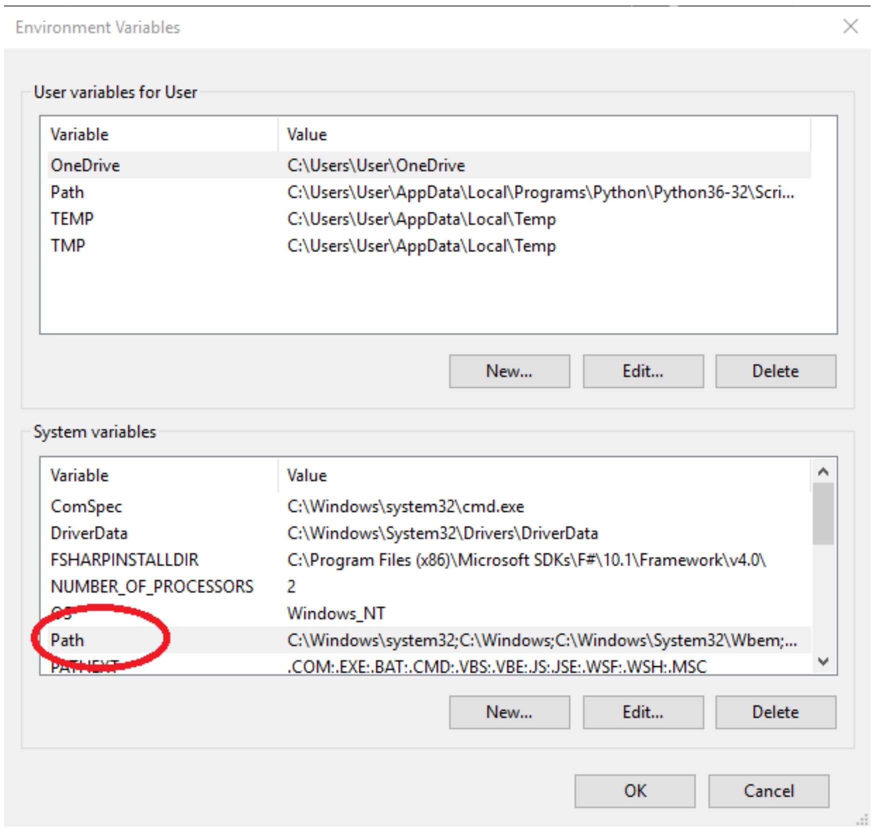
User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK Cancel Apply

The following window will open:


Environment Variables

User variables for User

Variable	Value
OneDrive	C:\Users\User\OneDrive
Path	C:\Users\User\AppData\Local\Programs\Python\Python36-32\Scri...
TEMP	C:\Users\User\AppData\Local\Temp
TMP	C:\Users\User\AppData\Local\Temp

New... Edit... Delete

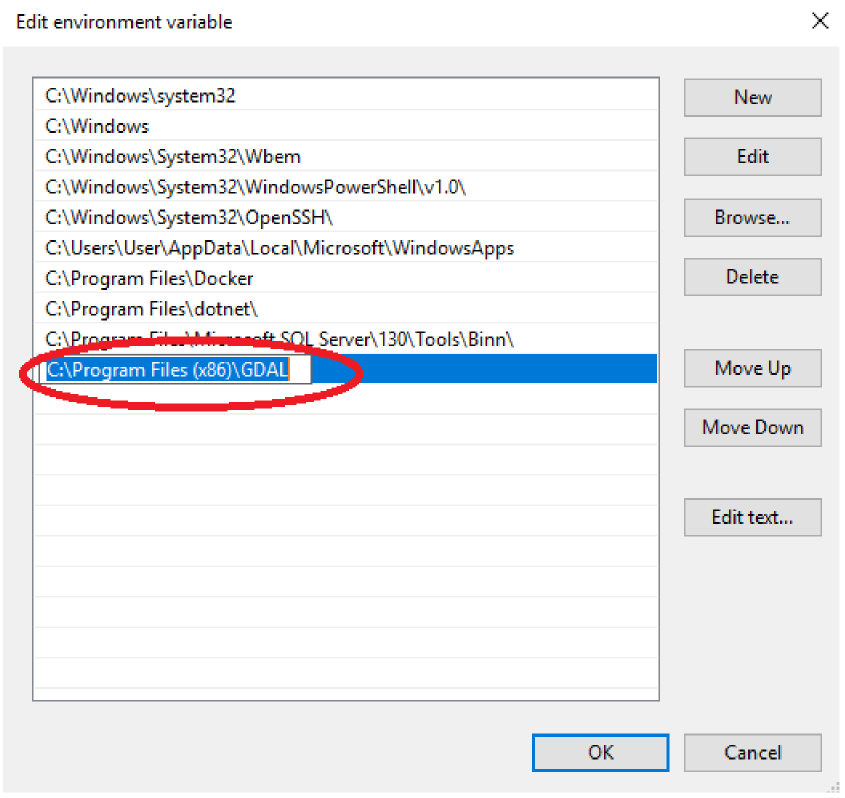
System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
FSHARPINSTALLDIR	C:\Program Files (x86)\Microsoft SDKs\F#\10.1\Framework\v4.0\
NUMBER_OF_PROCESSORS	2
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

New... Edit... Delete

OK Cancel

Double click on Path on the section System Variables, and a new window will open:

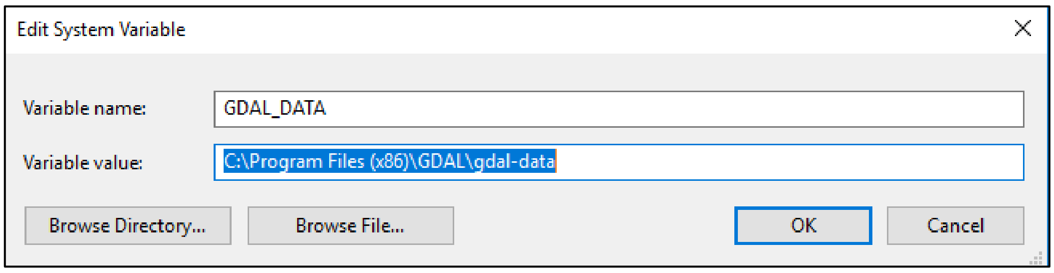

Edit environment variable

C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Users\User\AppData\Local\Microsoft\WindowsApps
C:\Program Files\Docker
C:\Program Files\dotnet\
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
C:\Program Files (x86)\GDAL

On this window, double click on an empty line, and add the path to GDAL, which should be either “C:\Program Files\GDAL\” or “C:\Program Files (x86)\GDAL”. Press Ok to save the configuration.

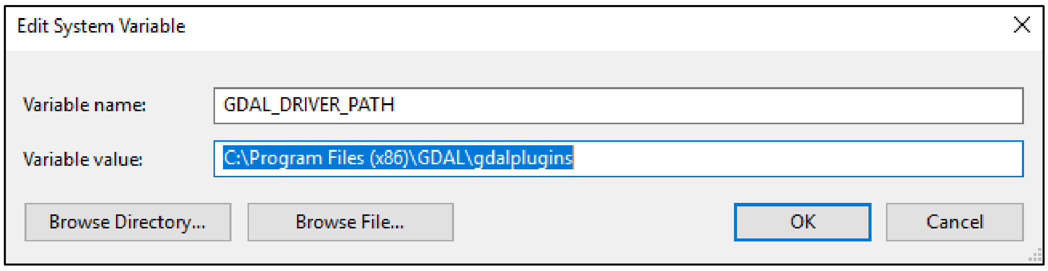
In the same System variables pane, click on “New” and then add the following in the dialogue box:

Variable name: GDAL_DATA
Variable value: C:\Program Files (x86)\GDAL\gdal-data


Edit System Variable

Variable name: GDAL_DATA
Variable value: C:\Program Files (x86)\GDAL\gdal-data

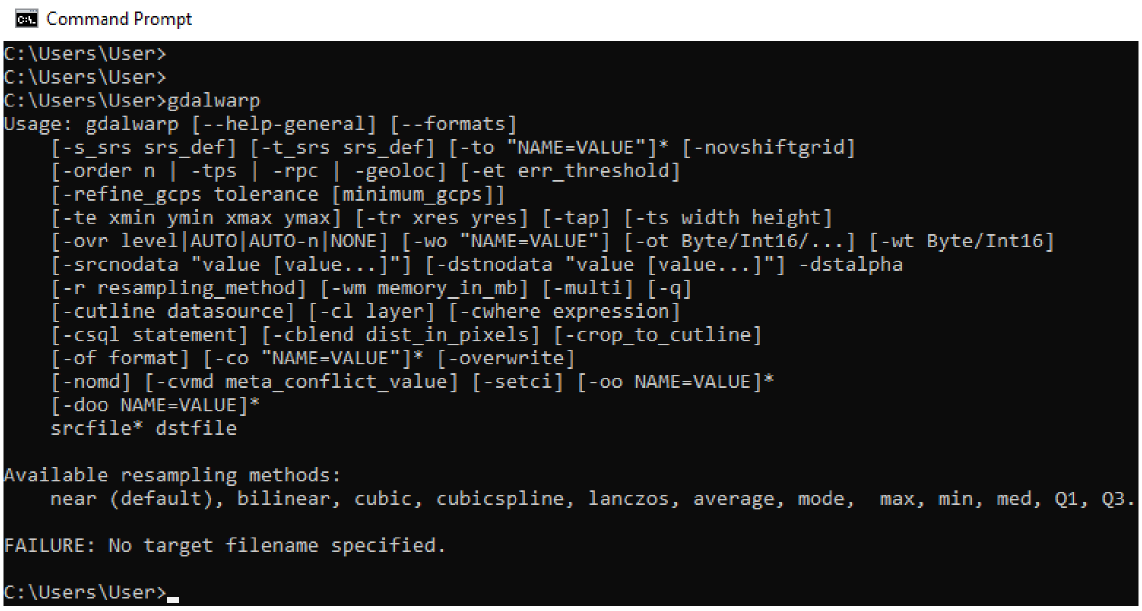
And once more:



```
Variable name:   GDAL_DRIVER_PATH
Variable value:  C:\Program Files (x86)\GDAL\gdalplugins
```

Variable name: GDAL_DRIVER_PATH
Variable value: C:\Program Files (x86)\GDAL\gdalplugins

To check if the installation was successful, open a new command prompt and type "gdalwarp". If some information about how to use gdalwarp appears installation was ok, as in the image below.

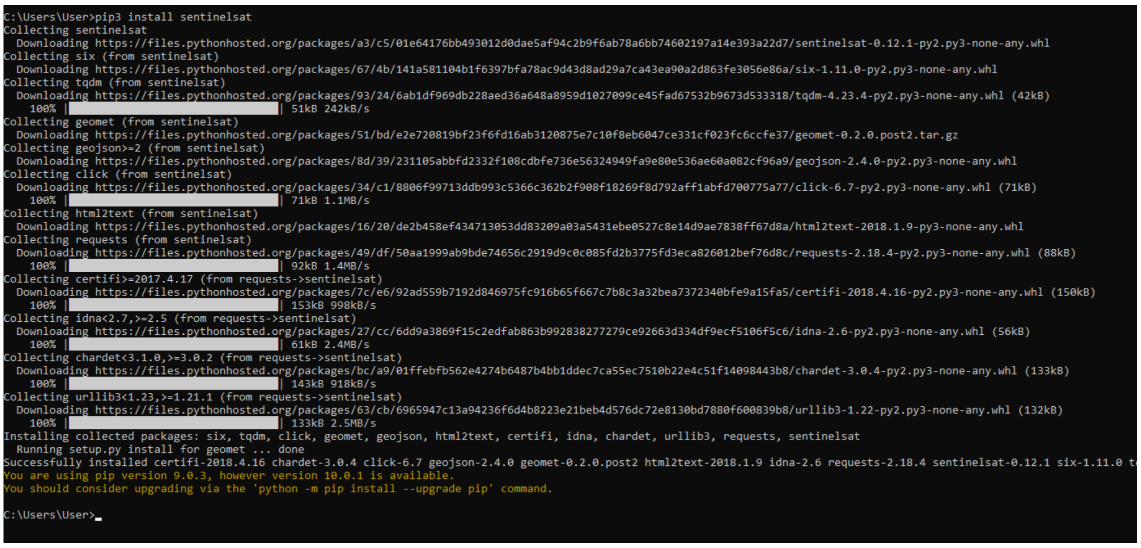


```
C:\Users\User>
C:\Users\User>
C:\Users\User>gdalwarp
Usage: gdalwarp [--help-general] [--formats]
    [-s_srs srs_def] [-t_srs srs_def] [-to "NAME=VALUE"]* [-novshiftgrid]
    [-order n | -tps | -rpc | -geoloc] [-et err_threshold]
    [-refine_gcps tolerance [minimum_gcps]]
    [-te xmin ymin xmax ymax] [-tr xres yres] [-tap] [-ts width height]
    [-ovr level|AUTO|AUTO-n|NONE] [-wo "NAME=VALUE"] [-ot Byte/Int16/...] [-wt Byte/Int16]
    [-srcnodata "value [value...]"] [-dstnodata "value [value...]"] -dstalpha
    [-r resampling_method] [-wm memory_in_mb] [-multi] [-q]
    [-cutline datasource] [-cl layer] [-cwhere expression]
    [-csql statement] [-cblend dist_in_pixels] [-crop_to_cutline]
    [-of format] [-co "NAME=VALUE"]* [-overwrite]
    [-nomd] [-cvmd meta_conflict_value] [-setci] [-oo NAME=VALUE]*
    [-doo NAME=VALUE]*
    srcfile* dstfile

Available resampling methods:
    near (default), bilinear, cubic, cubicspline, lanczos, average, mode,  max, min, med, Q1, Q3.

FAILURE: No target filename specified.

C:\Users\User>_
```

SentinelSat
To install SentinelStat, type on the command prompt:

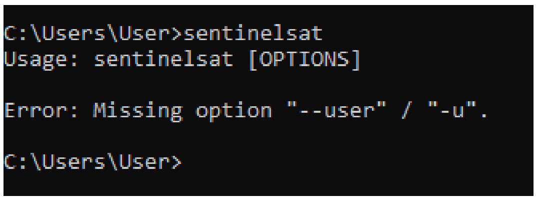
```
pip3 install sentinelsat
```

If everything was fine, the following screen will appear:



```
C:\Users\User>pip3 install sentinelsat
Collecting sentinelsat
  Downloading https://files.pythonhosted.org/packages/a3/c5/01e64176bb493012d0dae5af94c2b9f6ab78a6bb74602197a14e393a22d7/sentinelsat-0.12.1-py2.py3-none-any.whl
Collecting six (from sentinelsat)
  Downloading https://files.pythonhosted.org/packages/67/4b/141a581104b1f6397bfa78ac9d43d8ad29a7ca43ea90a2d863fe3056e86a/six-1.11.0-py2.py3-none-any.whl
Collecting tqdm (from sentinelsat)
  Downloading https://files.pythonhosted.org/packages/93/24/6ab1df969db228aed36a648a8959d1027099ce45fad67532b9673d533318/tqdm-4.23.4-py2.py3-none-any.whl (42kB)
    100% |████████████████████████████████| 51kB 242kB/s
Collecting geomet (from sentinelsat)
  Downloading https://files.pythonhosted.org/packages/51/bd/e2e720819bf23f6fd16ab3120875e7c10f8eb6047ce331cf023fc6ccfe37/geomet-0.2.0.post2.tar.gz
Collecting geojson>=2 (from sentinelsat)
  Downloading https://files.pythonhosted.org/packages/8d/39/231105abbfd2332f108cdbfe736e56324949fa9e80e536ae60a082cf96a9/geojson-2.4.0-py2.py3-none-any.whl
Collecting click (from sentinelsat)
  Downloading https://files.pythonhosted.org/packages/34/c1/8806f99713ddb993c5366c362b2f908f18269f8d792aff1abfd700775a77/click-6.7-py2.py3-none-any.whl (71kB)
    100% |████████████████████████████████| 71kB 1.1MB/s
Collecting html2text (from sentinelsat)
  Downloading https://files.pythonhosted.org/packages/16/20/de2b458ef434713053dd83209a03a5431ebe0527c8e14d9ae7838ff67d8a/html2text-2018.1.9-py3-none-any.whl
Collecting requests (from sentinelsat)
  Downloading https://files.pythonhosted.org/packages/49/df/50aa1999ab9bde74656c2919d9c0c085fd2b3775fd3eca826012bef76d8c/requests-2.18.4-py2.py3-none-any.whl (88kB)
    100% |████████████████████████████████| 92kB 1.4MB/s
Collecting certifi>=2017.4.17 (from requests->sentinelsat)
  Downloading https://files.pythonhosted.org/packages/7c/e6/92ad559b7192d846975fc916b65f667c7b8c3a32bea7372340bfe9a15fa5/certifi-2018.4.16-py2.py3-none-any.whl (150kB)
    100% |████████████████████████████████| 153kB 998kB/s
Collecting idna<2.7,>=2.5 (from requests->sentinelsat)
  Downloading https://files.pythonhosted.org/packages/27/cc/6dd9a3869f15c2edfab863b992838277279ce92663d334df9ecf5106f5c6/idna-2.6-py2.py3-none-any.whl (56kB)
    100% |████████████████████████████████| 61kB 2.4MB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests->sentinelsat)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |████████████████████████████████| 143kB 918kB/s
Collecting urllib3<1.23,>=1.21.1 (from requests->sentinelsat)
  Downloading https://files.pythonhosted.org/packages/63/cb/6965947c13a94236f6d4b8223e21beb4d576dc72e8130bd7880f600839b8/urllib3-1.22-py2.py3-none-any.whl (132kB)
    100% |████████████████████████████████| 133kB 2.5MB/s
Installing collected packages: six, tqdm, click, geomet, geojson, html2text, certifi, idna, chardet, urllib3, requests, sentinelsat
  Running setup.py install for geomet ... done
Successfully installed certifi-2018.4.16 chardet-3.0.4 click-6.7 geojson-2.4.0 geomet-0.2.0.post2 html2text-2018.1.9 idna-2.6 requests-2.18.4 sentinelsat-0.12.1 six-1.11.0 t
You are using pip version 9.0.3, however version 10.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\User>_
```

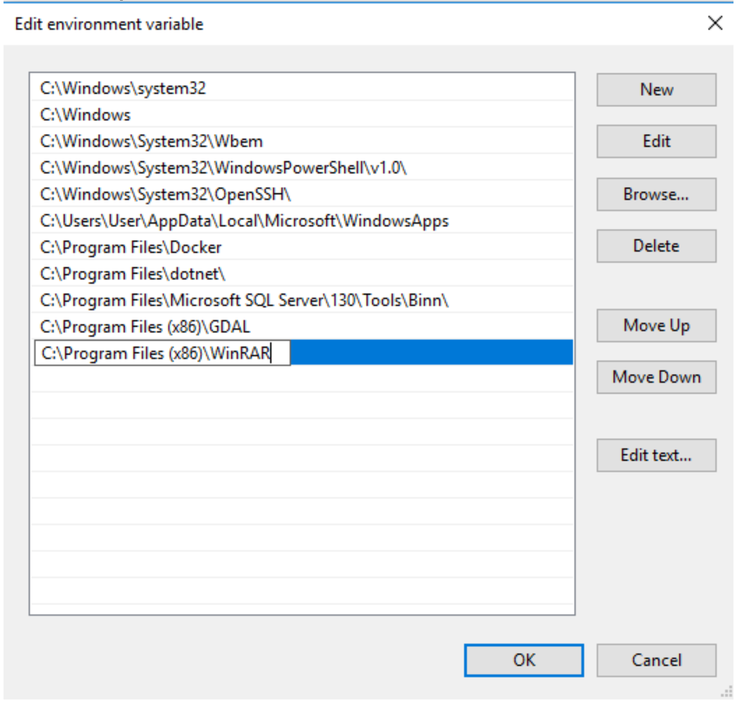
To check sentinelSat installation, type “sentinelsat” on the command prompt. A how to use message should appear as in the image below.



```
C:\Users\User>sentinelsat
Usage: sentinelsat [OPTIONS]

Error: Missing option "--user" / "-u".

C:\Users\User>
```

Winrar

Download winrar from the webpage[88] and install it normally. Finally, and following the same steps as in GDAL, add the winrar folder to the system path:



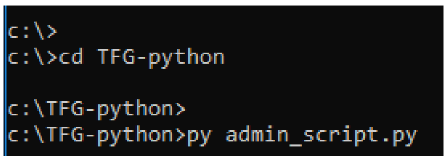
Edit environment variable

```
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Users\User\AppData\Local\Microsoft\WindowsApps
C:\Program Files\Docker
C:\Program Files\dotnet\
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
C:\Program Files (x86)\GDAL
C:\Program Files (x86)\WinRAR
```

88 https://www.win-rar.com/

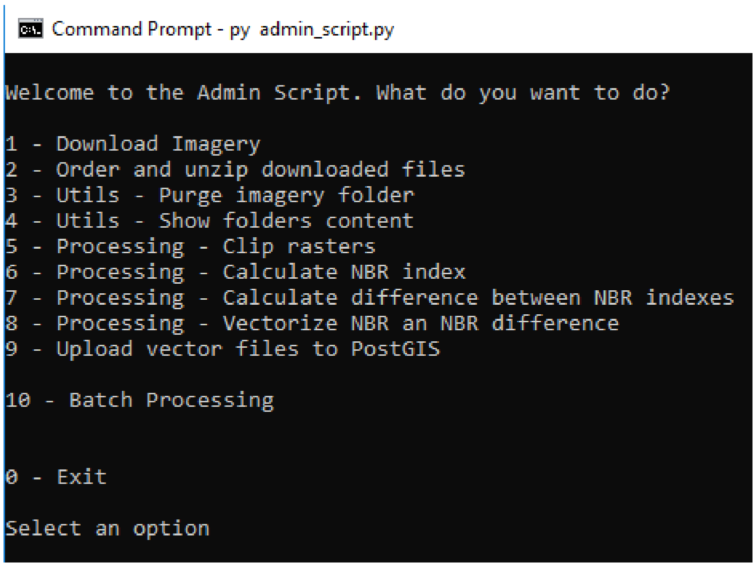
Run the backend!

In the command prompt, navigate to the folder containing the python scripts:



```
c:\>
c:\>cd TFG-python

c:\TFG-python>
c:\TFG-python>py admin_script.py
```

And press enter.



```
Command Prompt - py  admin_script.py

Welcome to the Admin Script. What do you want to do?

1 - Download Imagery
2 - Order and unzip downloaded files
3 - Utils - Purge imagery folder
4 - Utils - Show folders content
5 - Processing - Clip rasters
6 - Processing - Calculate NBR index
7 - Processing - Calculate difference between NBR indexes
8 - Processing - Vectorize NBR an NBR difference
9 - Upload vector files to PostGIS

10 - Batch Processing


0 - Exit

Select an option
```

12. Glossary

CNIG Centro Nacional de Información Geográfica.

GDAL Geospatial Data Abstraction Library. Library for Geographical imagery processing.

GeoJson Open standard format for graphical information, based on Json, JavaScript Object Notation.

GIS Geographical Information Systems.

Java Object Oriented and interpreted programing language.

JavaScript Interpreted programming language used for web pages.

OGR Library for vector file processing.

Raster Image format based on a matrix of points containing information about colour or/and brightness values.

RGB Red Green Blue. Colour system used in dark background devices.

Shapefile Vector file format, developed by ESRI, which is a de facto standard.

Vector Graphics Images that are defined using lines, curves and points in a 2D space instead of as a matrix of pixels.

13. Bibliography

1. **Consellería de Medio Rural.** mediorural.xunta.gal. [Online] 2006. [Cited: 30 December 2017.] http://mediorural.xunta.gal/fileadmin/arquivos/estatisticas/2010/11401/11401_lumes_2006.pdf.

2. **La Opinión A Coruña.** www.laopinioncoruna.es. [Online] 29 11 2017. [Cited: 30 12 2017.] http://www.laopinioncoruna.es/galicia/2017/11/07/galicia-acapara-tercio-superficie-quemada/1233262.html.

3. *LA MAGNITUD DE LA CRISIS INCENDIARIA DE 2006 EN GALICIA.* **BALSA BARREIRO, JOSÉ.** [ed.] Revista Montes. 109, A Coruña : s.n., 2012, pp. 39-44.

4. **Consellería de Medio Rural.** PLADIGA. *mediorural.xunta.gal.* [Online] 2017. [Cited: 30 12 2017.] http://mediorural.xunta.gal/fileadmin/arquivos/forestal/pladiga/2017/2_MEMORIA.pdf.

5. **European Commission.** Copernicus, Europe's eyes on Earth. [Online] 2015. [Cited: 3 January 2018.] http://copernicus.eu/sites/default/files/documents/Brochure/Copernicus_brochure_EN_web_Oct2017.pdf.

6. —. Copernicus in Brief. [Online] [Cited: 3 January 2018.] http://www.copernicus.eu/main/copernicus-brief.

7. **European Space Agency.** MultiSpectral Instrument (MSI) Overview. [Online] [Cited: 3 January 2018.] https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-2-msi/msi-instrument.

8. —. Sentinel-2 Overview. [Online] [Cited: 3 January 2018.] https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-2/overview.

9. **Planet Labs.** High Resolution Monitoring. [Online] [Cited: 4 January 2018.] https://www.planet.com/products/hi-res-monitoring/.

10. *OBJECT-BASED CLASSIFICATION VS. PIXEL-BASED CLASSIFICATION: COMPARITIVE IMPORTANCE OF MULTI-RESOLUTION IMAGERY.* **Robert C. Weih, Jr. and Norman D. Riggan, Jr.** Arkansas : The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vols. Vol. XXXVIII-4/C7 .

11. **GISGeography.** OBIA – Object-Based Image Analysis (GEOBIA) – Think Objects, Not Pixels. [Online] [Cited: 10 January 2018.] http://gisgeography.com/obia-object-based-image-analysis-geobia/.

12. **Open Source Geospatial Foundation.** MapServer. [Online] [Cited: 5 January 2018.] http://www.mapserver.org/.

13. *"Improvement and expansion of the Fmask algorithm: Cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images.* **Zhu, Zhe, Shixiong Wang, and Curtis E. Woodcock.** Remote Sensing of Environment 159, 2015, pp. 269-277.