Gis3w di Lorenzetti Walter & C. S.n.c.
Viale Giuseppe Verdi n. 24
51016 Montecatini Terme (PT)
P.IVA 01782000473

Creazione di un server “demo” per la pubblicazione di operazioni geospaziali implementate con PyWPS e loro utilizzo tramite client

Docente: Susanna Grasso (susanna.grasso@gmail.com)

Data: 05/07/2022
Versione documento: 1.0

Sommari

WPS - CONCETTI

OGC Web Processing Service

Il WPS (Web Processing Service) è una delle specifiche tecniche definite dall'OGC per offrire accesso a dati e funzionalità GIS tramite internet secondo uno standard specificato. Altri standard, forse più diffusi e comunemente utilizzati sono per esempio:

- WMS - Web Map Service
  Fornisce/restituisce "mappe" di dati spazialmente riferiti a partire da informazioni geografiche in diversi formati immagine idonei ad essere visualizzate su web browser, incluse in WEBGIS client o programmi GIS Desktop.
- WFS - Web Feature Service
  Permette la richiesta e l'importazione da parte di un client di dati GIS vettoriali attraverso il Web. In questo vengono restituiti i dati in formato geografico (la codifica standard è il GML, basata su XML, ma possono essere usati anche altri formati quali lo Shapefile, JSON, etc) e non come immagini come nel caso del WMS.
- WCS - Web Coverage Service
  Fornisce come immagini o dati raster (formato: geotiff, arcgrid, etc)

Per approfondimenti utile: QGIS Come client di dati/servizi OGC

I servizi **WPS**, a differenza di quelli sopra elencati, non solo "restituiscono" dati geografici archiviati sul server e richiesti dal client tramite web, ma forniscono servizi/processi di elaborazione geospaziale. I risultati/l'output del processo (in funzione ovviamente del processo implementato) possono essere di diverso formato: dati geospaziali, immagini, testo, csv, etc.. I processi e le elaborazioni possono inoltre essere condotte a partire da dati presenti/disponibili sul server o da dati di input forniti attraverso la rete dal client.

Come funziona e come si utilizza un servizio WPS?

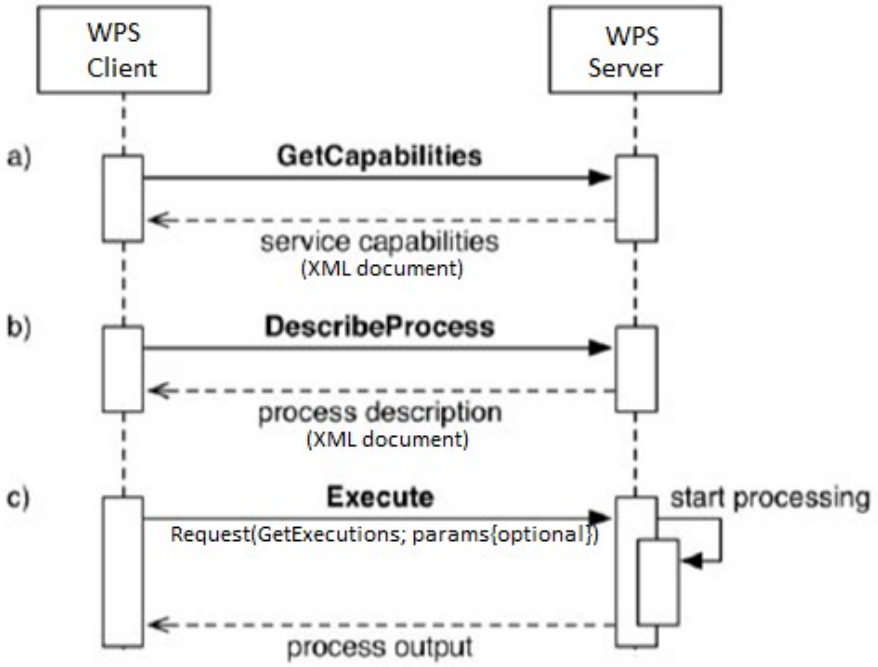
Operazioni e funzionalità di un servizio WPS

Il Web Processing service utilizza tre operazioni:

a) L'interazione iniziale con un WPS utilizza l'operazione ***GetCapabilities*** per restituire l'elenco dei processi offerti da quel servizio.
La risposta è un documento XML chiamato “capabilities document”.

b) La definizione e descrizioni degli input richiesti da un processo e gli output risultanti possono essere descritti tramite l'operazione ***DescribeProcess.*** La risposta è un documento XML.

c) Mentre l'operazione ***Execute*** consente a un client di eseguire un processo “passando” i parametri di input richiesti dal processo.



Metodi per invocare una richiesta

E’ possibile invocare ed effettuare richieste a un server WPS tramite una richiesta HTTP che può essere del tipo:

- **HTTP GET**. Con questo metodo i parametri di input vengono passati in “query string” con il metodo “chiave/valore”.
  Sicuramente è metodo il più semplice e il più immediato. È consigliato soprattutto in quelle richieste in cui è utile salvare nell’URL i parametri richiesti.

- **HTTP POST**
  Il metodo POST si differenza da GET in quanto i parametri della richiesta non vengono passati in query string ma utilizzando il formato XML della richiesta.
  La codifica POST è adatta per richieste “Execute” complesse, comprese quelle che richiedono valori complessi incorporati come inserimento di geometrie in input.

Formato dei dati di input e output

WPS si rivolge a processi che coinvolgono dati geospaziali (vettoriali e / o raster), ma può essere applicato anche a processi non spaziali. I dati richiesti dal WPS possono essere forniti attraverso una rete o disponibili sul server.

WPS definisce tre tipi di dati:

- ***LiteralData*** :singoli valori numerici o stringhe di testo
- ***Complex Data*** includono elementi come immagini, XML, CSV e strutture di dati personalizzate o proprietarie. Questo formato è solitamente principalmente utilizzato per passare dati vettoriali o raster.
- ***Boundingbox Data:*** coordinate geografiche per definire un'area rettangolare.

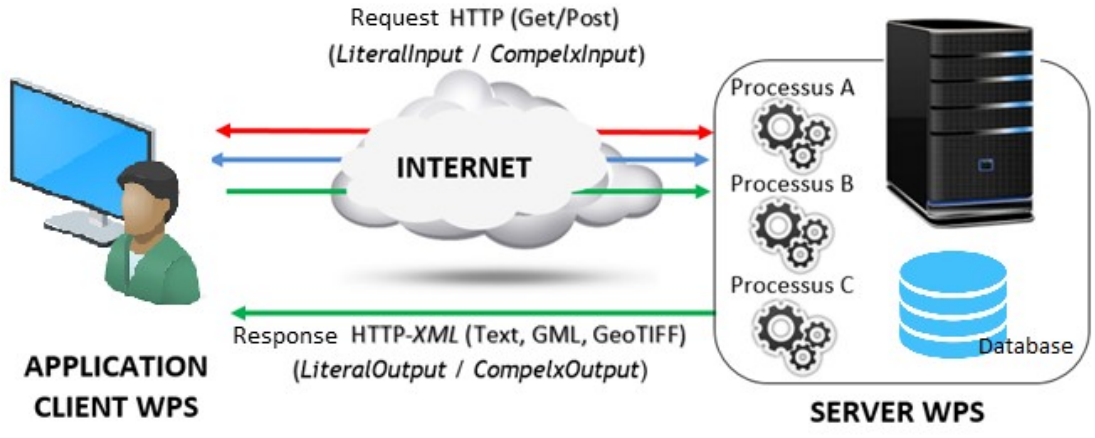
Le opzioni per la risposta in uscita

WPS consente diversi approcci diversi per l'esecuzione di un processo:

Restituzione di “output grezzi”
L'approccio più semplice è applicabile solo quando il WPS produce un solo output. In questo caso, l'output può essere restituito direttamente agli utenti nella sua forma grezza. Ad esempio: potrebbe restituire un'immagine in un file png o un raster o un file di ouput di altro tipo.

Restituzione di output incorporati in XML.
Una risposta a una richiesta *Execute* è un documento XML che include i metadati sulla richiesta, nonché gli output del processo codificati e inseriti nella risposta XML. Questa forma di risposta è consigliata quando la dimensione dell'output è inferiore a pochi megabyte.

Archiviazione degli output sul server
In questo caso il gli output vengono salvati sul server e viene restituito al client un documento XML contenente dei metadati/i riferimenti agli output (invece degli output stessi) corrispondenti alle posizioni accessibili dal Web (URL) da cui è possibile scaricare gli output.



I vantaggi del WPS

- Consente agli utenti di accedere ai calcoli indipendentemente dai software installati (procedura accessibile tramite browser web)
- I big data non devono essere archiviati localmente (lato client), ma vengono mantenuti dall'entità ospitante
- Tempi di elaborazione del server più rapidi rispetto allo scripting lato client
- Processamento lato server
- Distribuzione dello stesso processo all'interno di una agenzia/azienda (no errori dovuti ad installazione, applicazione, riutilizzo)

PYWPS

PyWPS è una implementazione dello standard WPS (Web Processing Service) definito dall' OGC (Open Geospatial Consortium[1]) che permette di accedere tramite web **a operazioni geospaziali personalizzabili** e serviti lato server denominati con *Processi*. In PyWPS i processi sono scritti nel linguaggio di programmazione Python e possono integrare strumenti quali: GRASS GIS, R, GDAL/OGR, proj.4 e altre librerie collegabili con Python.

[1] Open Geospatial Consortium è una organizzazione internazionale, no-profit, che si occupa di definire specifiche tecniche con l'obiettivo di sviluppare ed implementare standard per il contenuto, i servizi e l'interscambio di dati geografici (GIS - Sistema informativo geografico) che siano "aperti ed estensibili". Le specifiche definite da OGC sono pubbliche (PAS) e disponibili gratuitamente.

ESERCITAZIONE

Presentazione del server demo predisposto

La macchina virtuale preconfigurata predispone una “PyWPS-Demo”, ovvero una istanza server di PYWPS (versione 4.0) messa a punto utilizzando FLASK come webframework.

FLASK - Flask è un Python microframework per applicazioni web in Python.
Flask ha molti vantaggi perché permette un modo semplice e agile di sviluppare servizi web. Generalmente le applicazioni Flask sono sviluppate in un *virtualenv* per mantenere le dipendenze per ogni applicazione separata dall'installazione Python a livello di sistema.

> **Flask** is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

La versione di Demo permette di utilizzare un “Flask’s built-in server”.
Come illustrato nel capitolo successivo, digitando
`$ python3 demo.py`
Il server di demo girerà su http://localhost:5000/wps

La macchina virtuale preconfigurata, fornita già con l’installazione degli elementi sopra descritti, permette di creare una linea molto rapida di “sviluppo e messa in produzione” di servizi PYWPS. Potremo infatti utilizzare questa versione di demo per testare nuovi servizi e metterli poi successivamente in produzione tramite Apache. Per utilizzare l’applicazione in produzione sarà necessario far disporre il servizio da parte di un web server vero e proprio (in questo caso Apache HTTP Server). Essendo Apache multi-thread sarà così possibile effettuare più connessioni simultaneamente all'applicazione. Questo viene comunemente reso possibile tramite l’utilizzo di un “wrapper wsgi” per l'applicazione Flask: *mod-wsgi* che attiva l'ambiente virtuale e tutti i suoi moduli e dipendenze installati quando viene eseguito da Apache. [L’IMPLEMENTAZIONE E CONFIGURAZIONE DI SERVER DI PRODUZIONE NON TRATTATO NELLA PRESENTE LEZIONE]

> **Apache** *HTTP Server (usually just called Apache): web server*

> **mod_wsgi** *is an Apache HTTP Server module provides a WSGI (Web Server Gateway Interface) compliant interface for hosting Python based web applications under Apache.*

Prendiamo confidenza con la demo predisposta, l'accesso ai servizi precaricati e le funzionalità di base

Avviare l'istanza demo

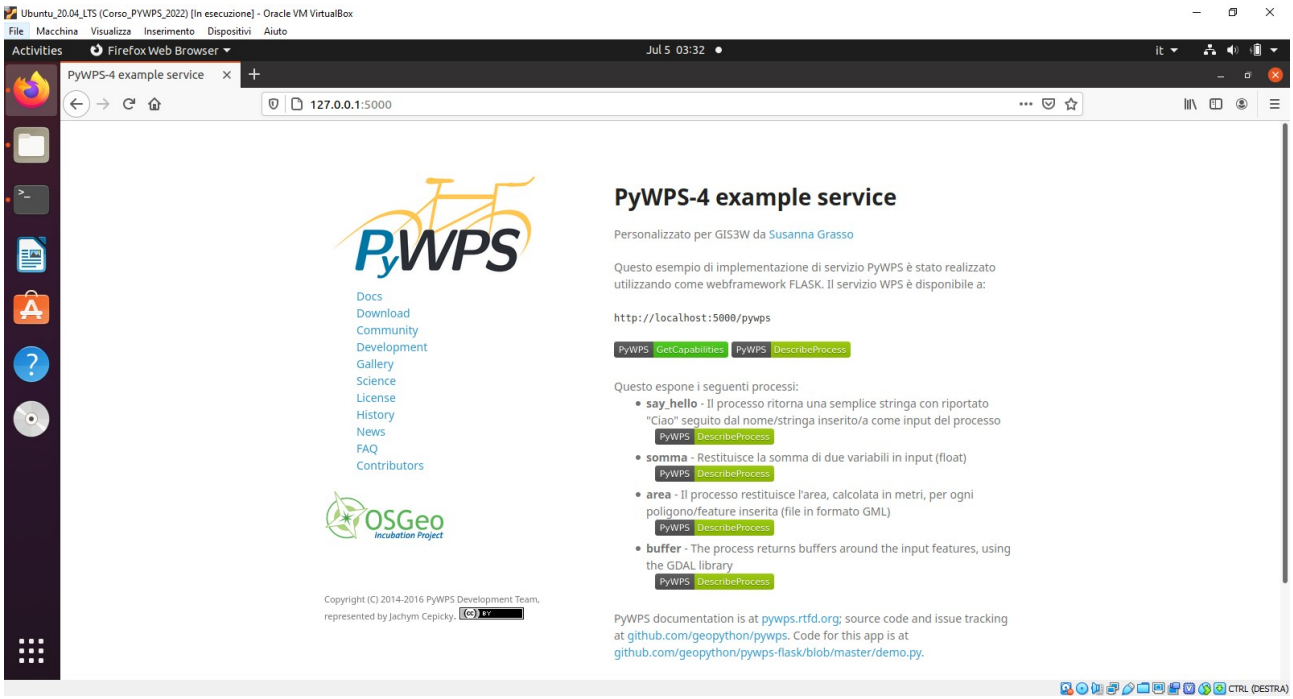
Avviare la macchina virtuale fornita inserendo:
utente: osboxes.org
pass: osboxes.org

Avviare l'istanza demo da prompt dei comandi (cmd):

```
$ cd /home/osboxes/pywps-flask/
$ python3 demo.py
```

L'interfaccia web predisposta con l'app Flask sarà visualizzabile all'indirizzo http://localhost:5000/ o http://127.0.0.1:5000

I servizi WPS saranno invece accessibili all'indirizzo: http://localhost:5000/pywps



Accesso ai servizi - Operazioni e funzionalità di base

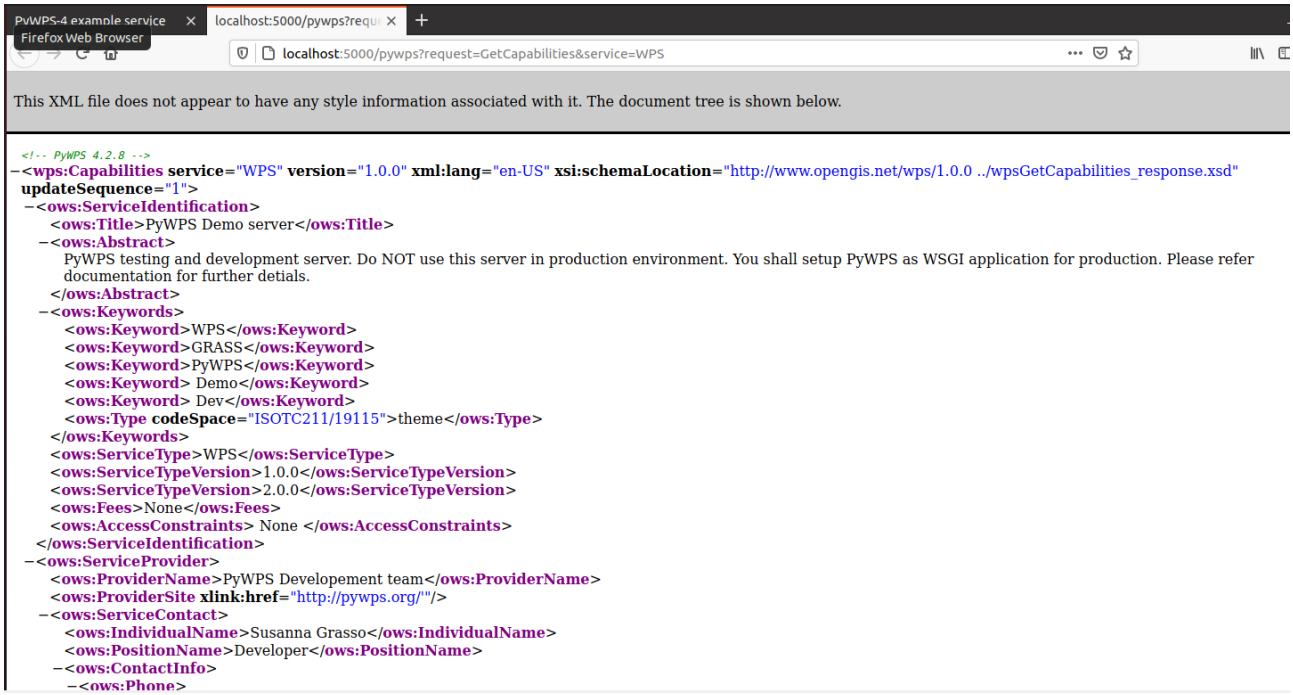
Abbiamo il nostro servizio che risponde alla root principale:

```
http://localhost:5000/pywps
```

Invochiamo le ***GetCapabilities*** per sapere quali sono i processi offerti dal servizio:

```
http://localhost:5000/pywps?service=WPS&request=GetCapabilities
```

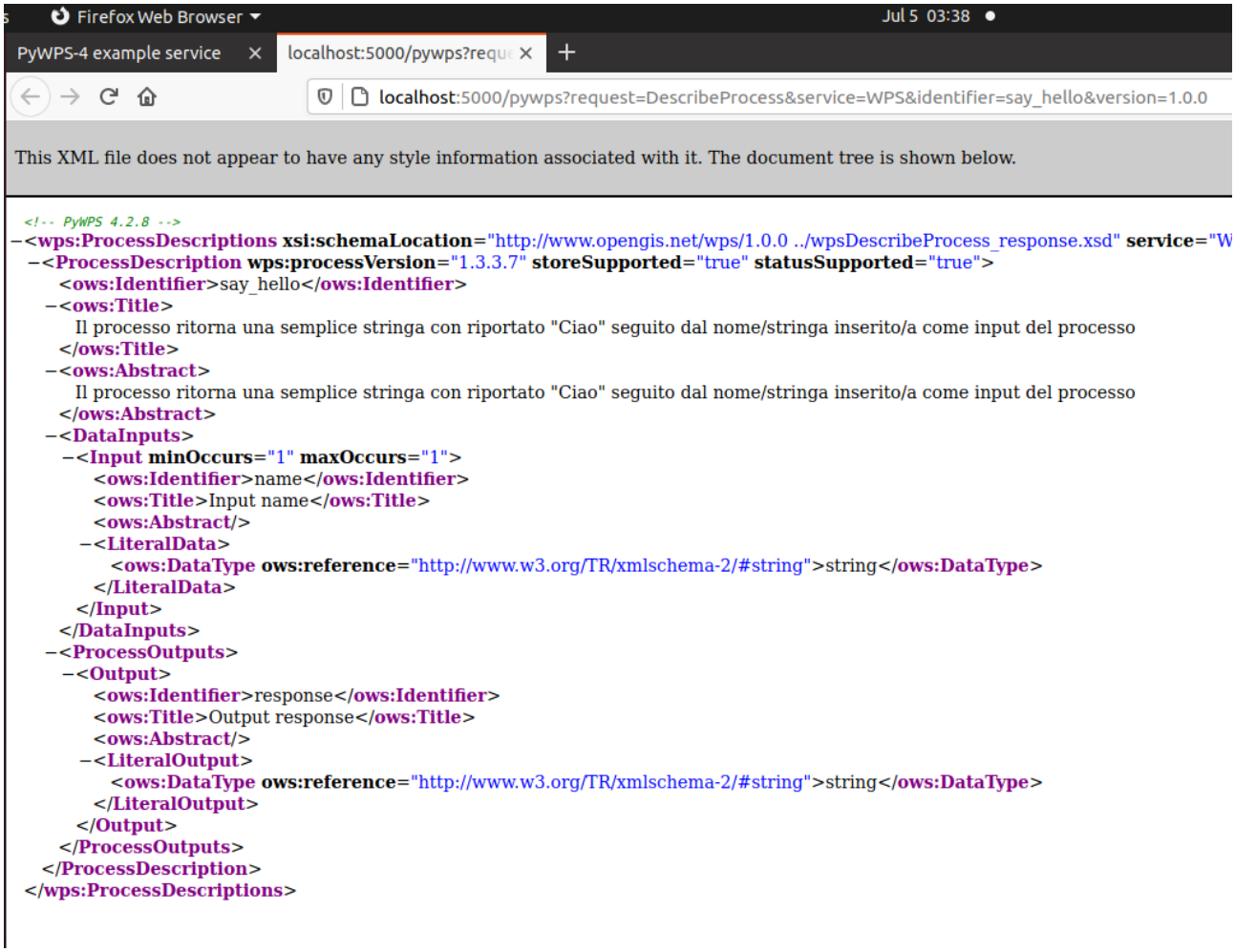
Il servizio ci risponde con un documento XML all'interno del quale possiamo vedere il l'identificativo (identifier); il titolo (Title) e la breve descrizione (Abstract) dei servizi offerti:



Esploriamo il processo (più semplice) *say_hello*. Per sapere quali siano gli input e gli output del servizio richiamiamo le del servizio ***DescribeProcess*** (indicando il sui identifier):

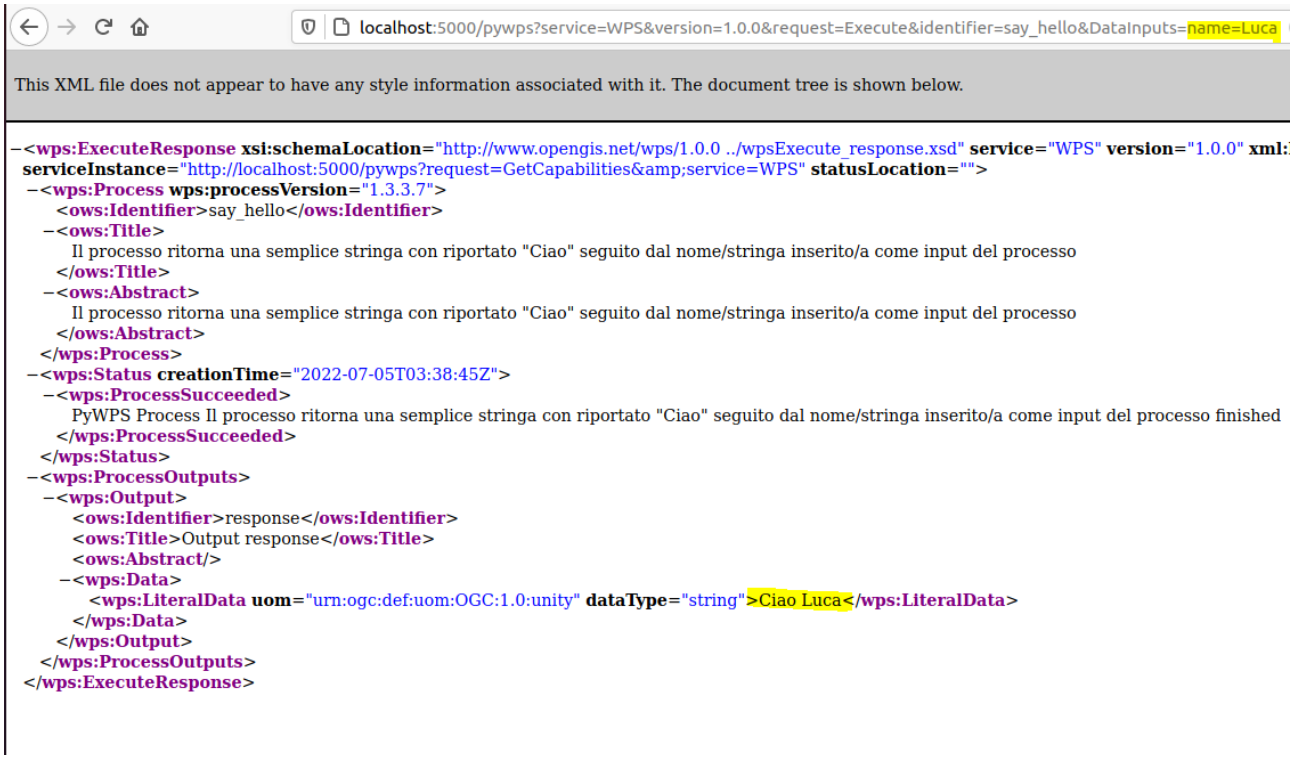
```
http://localhost:5000/pywps?
service=WPS&version=1.0.0&request=DescribeProcess&identifier=say_hello
```

La risposta della richiesta è un documento XML contenente, per il processo *say_hello*:
Title, Identifier, Abstract (optional), DataInputs (optional description), DataOutputs (optional description)



Per eseguire il processo (Execute) passando come input l'argomento: name=Luca

```
http://localhost:5000/pywps?
service=WPS&version=1.0.0&request=Execute&identifier=say_hello&DataInputs=name=Luca
```



Esempio 1.1 – processo “area”

Vediamo la descrizione del processo “area”

```
http://localhost:5000/pywps/?
request=DescribeProcess&service=WPS&identifier=area&version=1.0.0
```

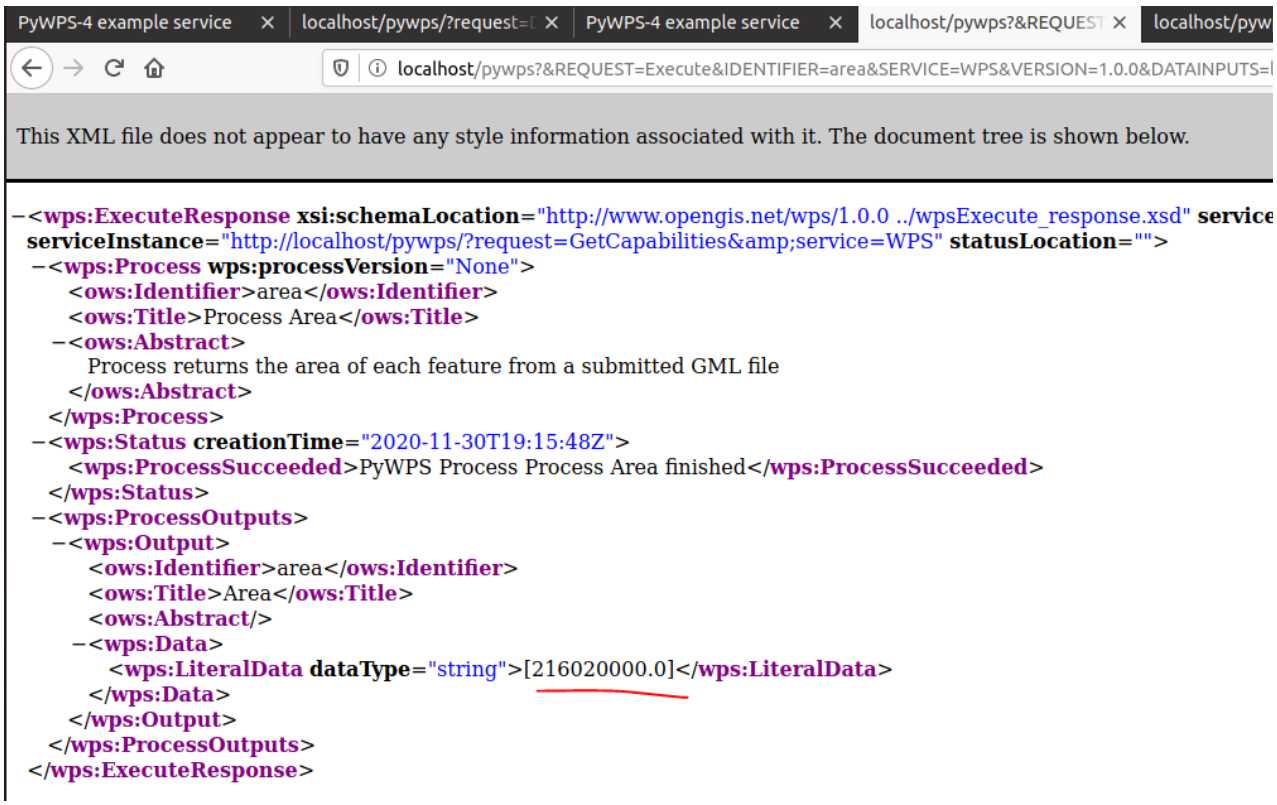
Il processo restituisce l’area di un poligono inviato in formato formato GML.
L’Identifier del dato da inserire in input è “layer”.

Caso 1 – Passo con HTTP GET il percorso di un file che ho caricato sul mio server
Carico nella cartella ..*pywps-flask/static/data* il mio flle *bacino_pellice.gml*
Il file risulta così raggiungibile all’URL
http://localhost:5000/static/data/bacino_pellice.gml

Eseguiamo il processo passandogli quindi la posizione sul server del file:

```
http://localhost:5000/pywps?
&REQUEST=Execute&IDENTIFIER=area&SERVICE=WPS&VERSION=1.0.0&DATAINPUTS=layer=@xli
nk:href=http://localhost:5000/static/data/bacino_pellice.gml
```

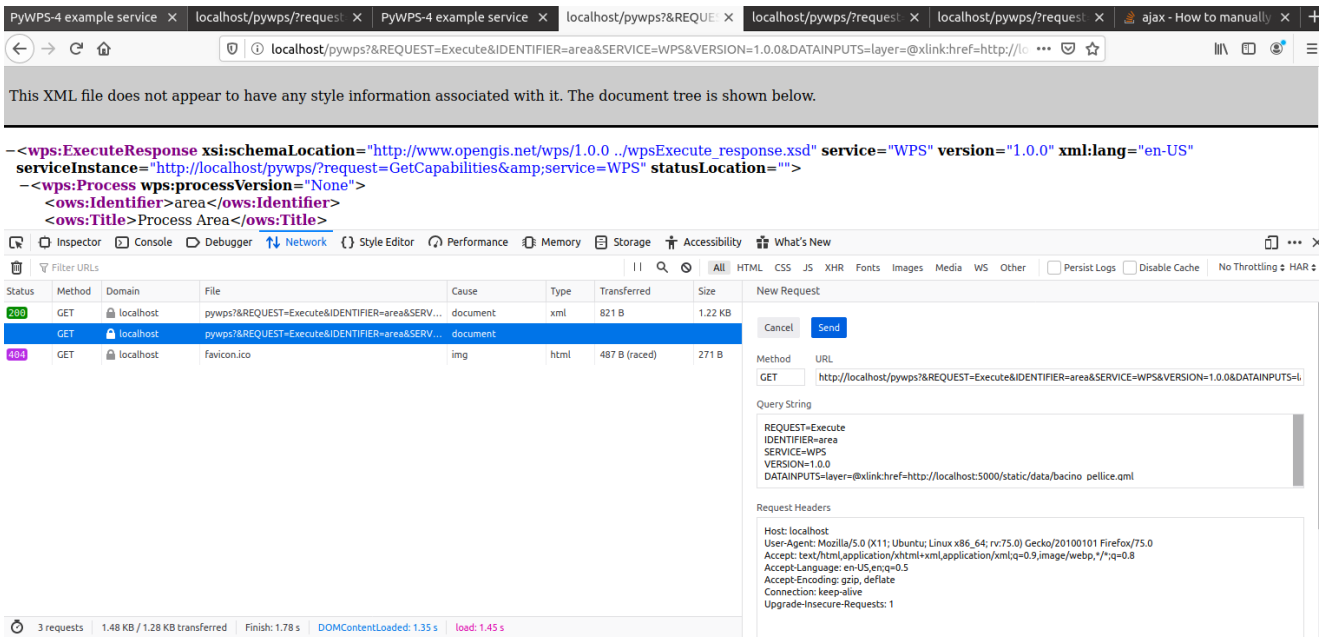
Risultato: 216.02 km^2 circa



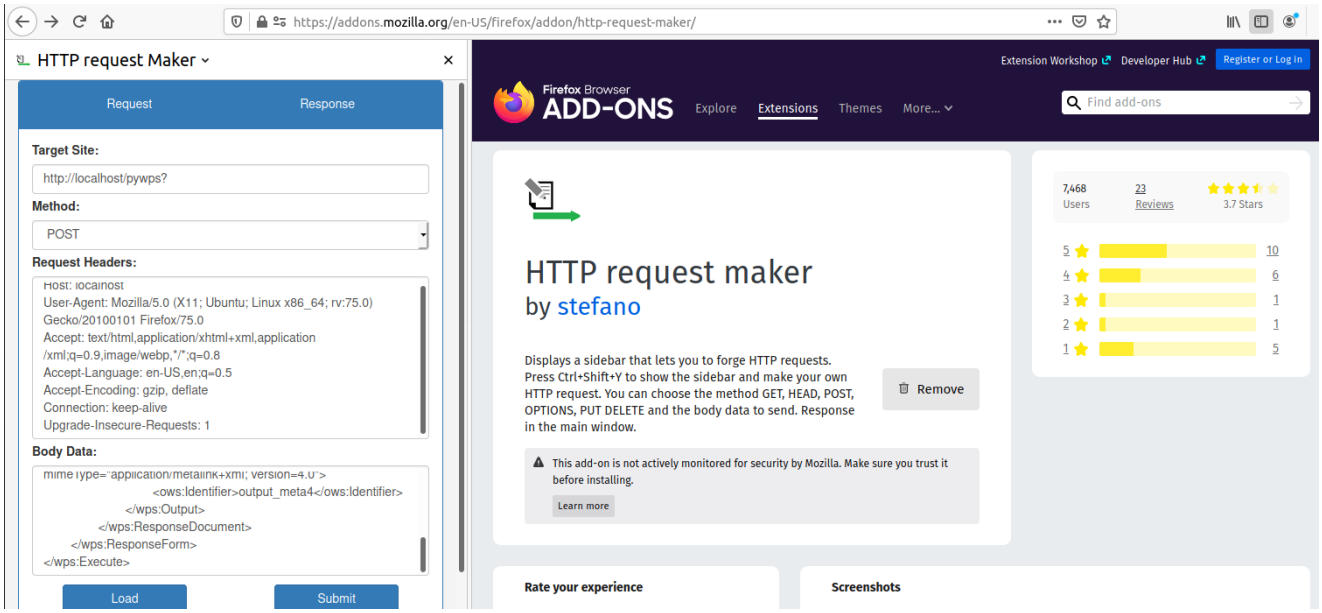
Caso 2 – Passo con HTTP POST la richiesta

Opzione 1 Con Firefox Inspect

Open Network panel in Developer Tools by pressing Ctrl+Shift+E or by going Menubar -> Tools -> Web Developer -> Network. Then Click on small door icon on top-right (in expanded form in the screenshot, you'll find it just left of the highlighted Headers), second row (if you don't see it then reload the page) -> Edit and resend whatever request you want



Opzione 2 con plugin per Firefox: http request marker



Target Site:
http://localhost/pywps?

Method:
POST

Request Header:
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

Upgrade-Insecure-Requests: 1

Body data: POST_pellicegml

Esempio 1.2 – processo “area”

Nell’esempio viene inserito un file vettoriale gml di forma rettangolare di lato 10Km (Zona Bobbio Pellice – Piemonte)

Vettoriale in input: rettangolo.gml_
Risultato: 100’000’000.0 m^2 (100Km^2)

RICHIESTA HTTP GET
http://localhost:5000/pywps?&REQUEST=Execute&IDENTIFIER=area&SERVICE=WPS&VERSION=1.0.0&DATAINPUTS=layer=@xlink:href=http://localhost:5000/static/data/rettangolo.gml

RICHIESTA HTTP POST
Esempio con una geometria rettangolare: POST_rettangologml

Esempio 2 – processo “buffer”

Nell’esempio viene inserito il file “punto.gml” (Zona Bobbio Pellice – Piemonte)

http GET
http://localhost:5000/pywps?&REQUEST=Execute&IDENTIFIER=buffer&SERVICE=WPS&VERSION=1.0.0&DATAINPUTS=poly_in=@xlink:href=http://localhost:5000/static/data/punto.gml;buffer=100

HTTP POST

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1" xmlns:xlink="http://www.w3.org/1999/
xlink" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
  <ows:Identifier>buffer</ows:Identifier>
  <wps:DataInputs>
   <wps:Input>
    <ows:Identifier>poly_in</ows:Identifier>
    <wps:Data>
     <wps:ComplexData mimeType="application/gml+xml">
<ogr:FeatureCollection
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://ogr.maptools.org/ punto.xsd"
     xmlns:ogr="http://ogr.maptools.org/"
     xmlns:gml="http://www.opengis.net/gml">
```

```
  <gml:boundedBy>
    <gml:Box>
      <gml:coord><gml:X>792477.8699874171</gml:X><gml:Y>5591104.959135194</
gml:Y></gml:coord>
      <gml:coord><gml:X>792477.8699874171</gml:X><gml:Y>5591104.959135194</
gml:Y></gml:coord>
    </gml:Box>
  </gml:boundedBy>

  <gml:featureMember>
    <ogr:punto fid="punto.0">
      <ogr:geometryProperty><gml:Point
srsName="EPSG:3857"><gml:coordinates>792477.869987417,5591104.95913519</
gml:coordinates></gml:Point></ogr:geometryProperty>
      <ogr:id>1</ogr:id>
    </ogr:punto>
  </gml:featureMember>
</ogr:FeatureCollection>
        </wps:ComplexData>
    </wps:Data>
   </wps:Input>
   <wps:Input>
    <ows:Identifier>buffer</ows:Identifier>
    <wps:Data>
     <wps:LiteralData>1000</wps:LiteralData>
    </wps:Data>
   </wps:Input>
  </wps:DataInputs>
</wps:Execute>
```

Per farsi restituire il link alla risorsa anziché il file xml
DOC: *https://pywps.readthedocs.io/en/latest/process.html#returning-large-data*

```
...ResponseDocument=<outputidentifier>=@asReference=true...
```

Or a POST request:

```
...
<wps:ResponseForm>
    <wps:ResponseDocument>
        <wps:Output asReference="true">
            <ows:Identifier>output</ows:Identifier>
            <ows:Title>Some Output</ows:Title>
        </wps:Output>
    </wps:ResponseDocument>
</wps:ResponseForm>
...
```

http://localhost:5000/pywps?
&REQUEST=Execute&IDENTIFIER=buffer&SERVICE=WPS&VERSION=1.0.0&DATAINPUTS=poly_in=@xlink:href=http://localhost:5000/static/data/
punto.gml;buffer=10&ResponseDocument=buff_out=@asReference=true

Es. risposta: http://localhost:5000/outputs/f64ce9a6-3375-11eb-ab8e-138577e053d2/
punto_buffer.gml

Per far scaricare diversi multiple files
https://pywps.readthedocs.io/en/latest/process.html#returning-multiple-files

Creazione di un nuovo servizio PYWPS

Servizio base: “somma”

Creazione del servizio somma.py

Servizi PYWPS con operazioni GIS utilizzando le librerie GDAL

Creazione dei servizi
- buffer_json.py
- rasterstats.py
- plot_timeseries.py

buffer_json.py

Crea un buffer intorno a una geometria vettoriale

rasterstats.py

Inserimento di uno shapefile in formato GML e ritorno delle statistiche (min, max e media) dei valori del DEM (Oregon)

ywps-flask/static/data/

NB. Prima di scrivere passare a scrivere e testare lo script installare la libreria rasterstats

```
sudo apt install python3-pip)
pip3 install rasterstats
```

http://localhost:5000/pywps?&REQUEST=Execute&IDENTIFIER=rasterstats&SERVICE=WPS&VERSION=1.0.0&DATAINPUTS=poly_in=@xlink:href=http://localhost:5000/static/data/vettoriale_per_rasterstatistics.gml

plot_timeseries.py

Esempio di un script che plotta, per un dataset in formato NETCF, i valori per un punto nel tempo (timeseries). Dataset: “ tg_0.25deg_day_2020_01_grid_ensmean.nc “
Come file si può sempre usare: “punto.gml”

Nell’esempio proposto viene utilizzato il dataset dell’ EOBS della temperatura minima per il mese di Luglio 2020.

E-OBS comes as an ensemble dataset and is available on a 0.1 and 0.25 degree regular grid for the elements daily mean temperature TG, daily minimum temperature
Risorsa online: https://surfobs.climate.copernicus.eu/dataaccess/access_eobs_months.php

Scaricamento del dato:

https://knmi-ecad-assets-prd.s3.amazonaws.com/ensembles/data/months/ens/tg_0.25deg_day_2020_07_grid_ensmean.nc

NB Testare librerie

```
sudo apt install python3-pip
pip3 install netCDF4
pip3 install pandas
```

risk-analysis.py

Esempio di un script che a partire dall'inserimento della matrice di pericolo e di vulnerabilità e di un punto per cui ci interessa conoscere il valore della matrice di rischio calcolata come R = P x V. (con i raster riportati nell'esempio di questo caso il valore di V viene dimezzato ovvero vale da 0.5 a 2.5. P assume valori da 1 a 3)

Tratto spunto da e parte dei dati da: https://www.itc.nl/ilwis/applications-guide/application-1/

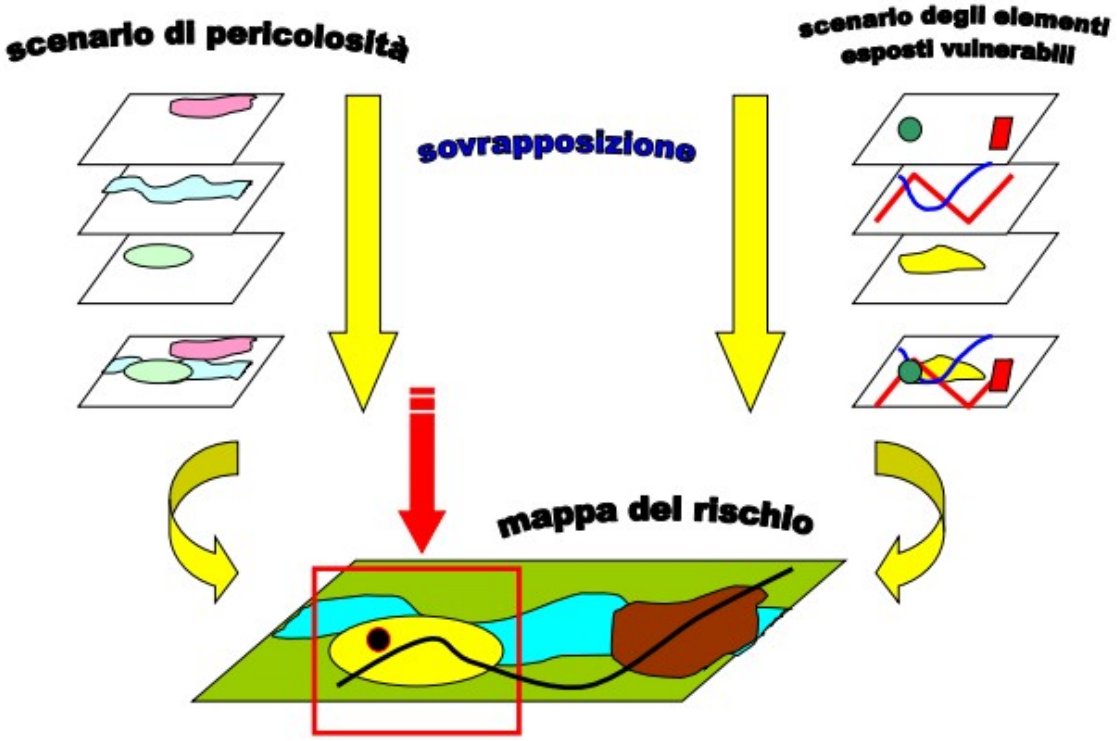
Il rischio quindi è traducibile nella formula: **R = P x V x E**

P = PERICOLOSITÀ: la probabilità che un fenomeno di una determinata intensità si verifichi in un certo periodo di tempo, in una data area.

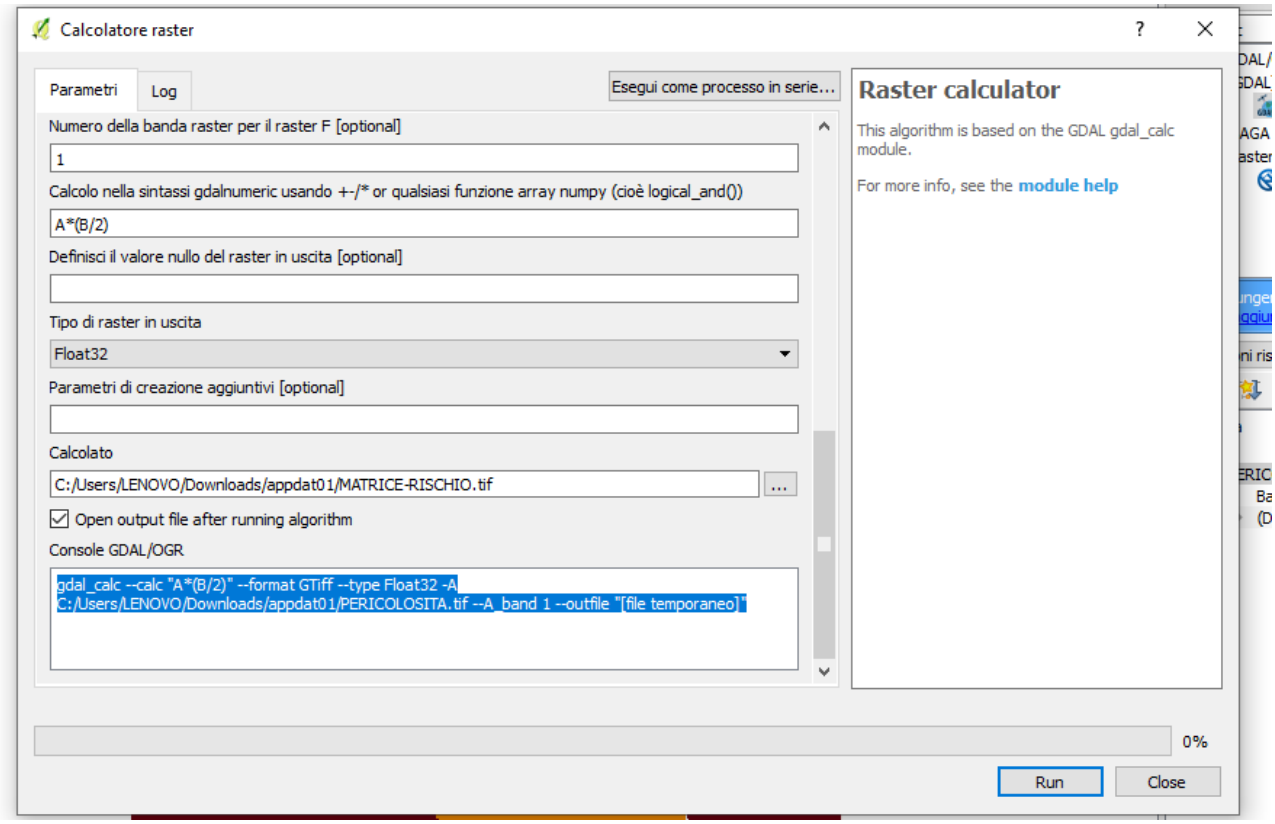
V = VULNERABILITÀ: la vulnerabilità di un elemento (persone, edifici, infrastrutture, attività economiche) è la propensione a subire danneggiamenti in conseguenza delle sollecitazioni indotte da un evento di una certa intensità.

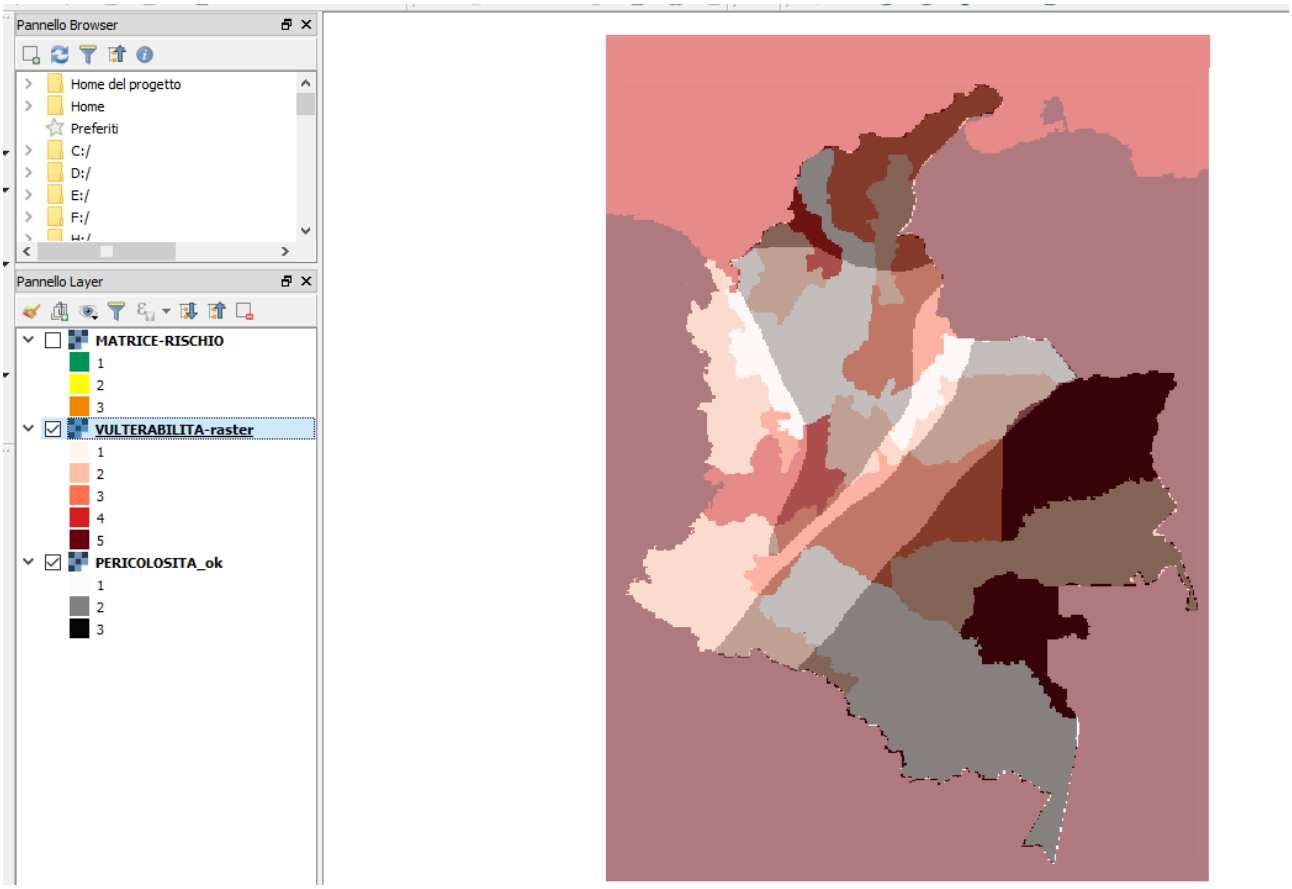
E = ESPOSIZIONE o Valore esposto: è il numero di unità (o "valore") di ognuno degli elementi a rischio presenti in una data area, come le vite umane o gli insediamenti.

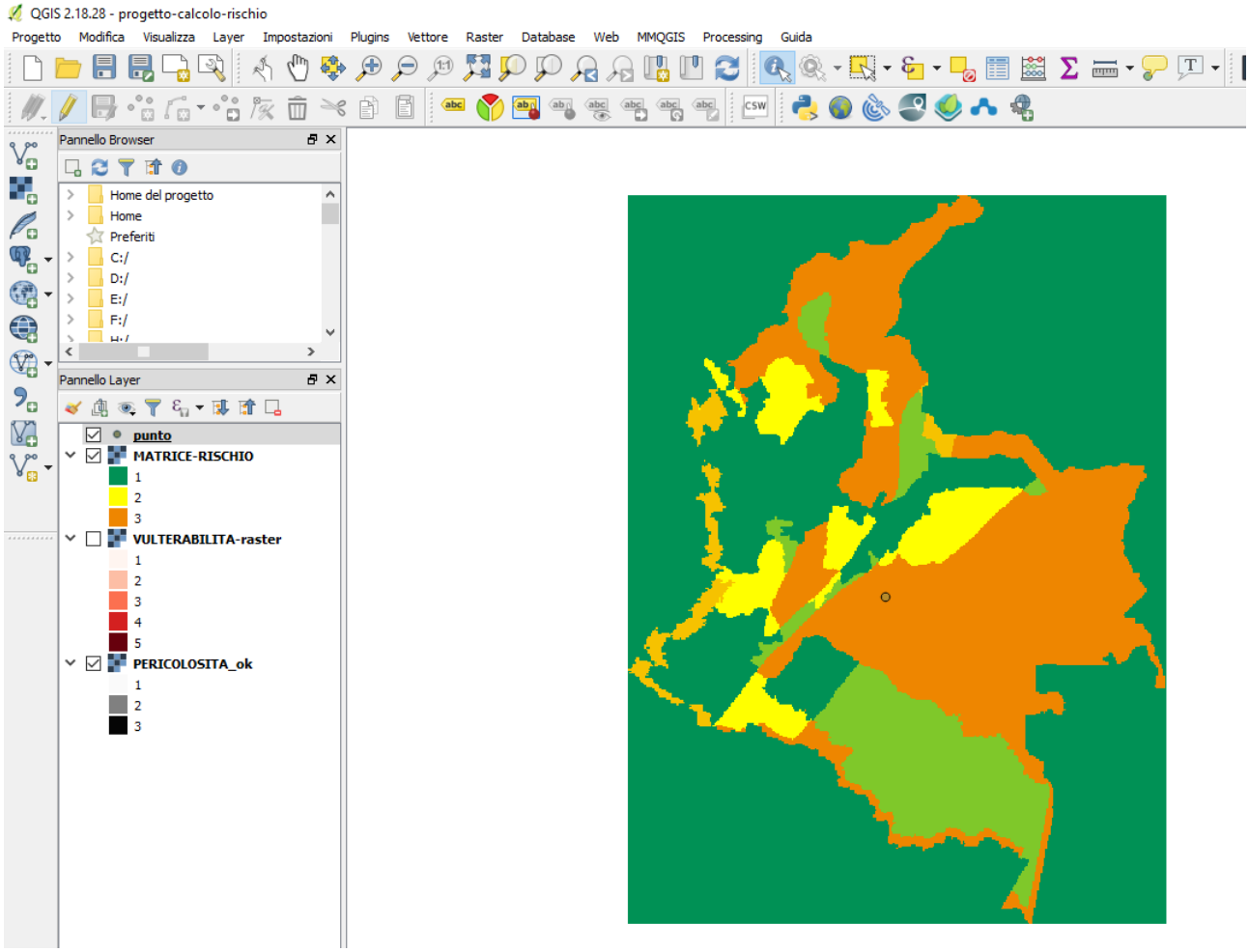
		Intensità		
		bassa	media	elevata
Probabilità	bassa	P1	P2	P3
	media	P1-P2	P2	P3
	elevata	P2	P2-P3	P3



Classe	Rischio
R1	**Rischio moderato:** per il quale i danni sociali, economici e al patrimonio ambientale sono marginali;
R2	**Rischio medio:** per il quale sono possibili danni minori agli edifici, alle infrastrutture e al patrimonio ambientale che non pregiudicano l'incolumità del personale, l'agibilità degli edifici e la funzionalità delle attività economiche;
R3	**Rischio elevato:** per il quale sono possibili problemi per l'incolumità delle persone, danni funzionali agli edifici e alle infrastrutture con conseguente inagibilità degli stessi, la interruzione di funzionalità delle attività socioeconomiche e danni rilevanti al patrimonio ambientale;
R4	**Rischio molto elevato:** per il quale sono possibili la perdita di vite umane e lesioni gravi alle persone, danni gravi agli edifici, alle infrastrutture e al patrimonio ambientale, la distruzione di attività socioeconomiche.


Calcolatore raster
Parametri
Log
Esegui come processo in serie...
Numero della banda raster per il raster F [optional]
1
Calcolo nella sintassi gdalnumeric usando +-/* or qualsiasi funzione array numpy (cioè logical_and())
A*(B/2)
Definisci il valore nullo del raster in uscita [optional]
Tipo di raster in uscita
Float32
Parametri di creazione aggiuntivi [optional]
Calcolato
C:/Users/LENOVO/Downloads/appdat01/MATRICE-RISCHIO.tif
Open output file after running algorithm
Console GDAL/OGR
gdal_calc --calc "A*(B/2)" --format GTiff --type Float32 -A C:/Users/LENOVO/Downloads/appdat01/PERICOLOSITA.tif --A_band 1 --outfile "[file temporaneo]"
Raster calculator
This algorithm is based on the GDAL gdal_calc module.
For more info, see the module help
0%
Run
Close


Pannello Browser
Home del progetto
Home
Preferiti
C:/
D:/
E:/
F:/
Pannello Layer
MATRICE-RISCHIO
1
2
3
VULTERABILITA-raster
1
2
3
4
5
PERICOLOSITA_ok
1
2
3



Esempio valore per il punto di coordinate (EPSG 4326): 793152, 971969
valore matrice rischio R = P x (V/2) → 3

Esempio di un servizio PYWPS utilizzando funzioni di GRASS

Delimitazione bacini dighe (vedere renerfor_delimitazione.py)

Obiettivo:

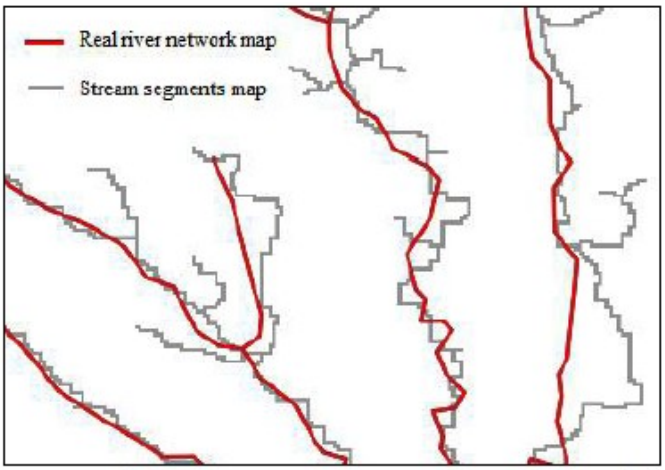
Procedura per la delimitazione di bacini idrografici a partire dalla coordinata della sezione di chiusura (il processo volendo si potrebbe implementare in serie ovvero per più punti!! – Vedere script definito per delimitazioni bacini a monte delle dighe catalogate dal "Progetto Dighe" per la regione Sardegna).

Input:

- *Raster della drainage direction (direzione di flusso)* creato a partire dal DEM utilizzando il comando r.watershed
- *Coordinata della diga/sezione di chiusura* "riposizionate sul raster *stream_segments"*

Output:

Bacini delimitati utilizzando l'algoritmo che richiama il comando di GRASS: r.water.outlet



Comparazione tra il raster stream segments e il reticolo idrografico ISPRA

Posizionamento delle coordinate delle dighe/sezioni di chiusura sullo stream_segments

LINK UTILI

https://pywps.readthedocs.io/en/latest/wps.html

https://pywps.readthedocs.io/projects/PyWPS-Demo/en/latest/

PyWPS 4.x - stable

https://www.ogc.org/standards/wps

MATERIALE PER IL CORSO
https://drive.google.com/drive/folders/1mlzovHIctsI1l25Nvc3d6ai8-mQO1z8t?usp=sharing

ALTRO
Esempi di servizi WPS disponibili online
POLITECNICO DI TORINO – PROGETTO RENERFOR
WebGIS sviluppato con G3Wsuite con applicazione PyWPS
http://idrologia.org:8090/

WMO WOUDC (World Ozone & Ultraviolet Radiation Data Centre)
World Meteorological Organization (WMO) data centre supporting the Global Atmosphere Watch (GAW) program operated by Environment and Climate Change Canada.
https://woudc.org/about/data-access.php#ogc-wps

Esempi di operazioni geospaziali più complesse che è possibile implementare magari come servizio WPS

- https://unidata.github.io/python-gallery/examples/500hPa_Absolute_Vorticity_winds.html

  https://towardsdatascience.com/object-based-land-cover-classification-with-python-cbe54e9c9e24

  https://www.earthdatascience.org/courses/use-data-open-source-python/spatial-data-applications/lidar-remote-sensing-uncertainty/extract-data-from-raster/