



POLITECNICO
MILANO 1863

Software Release Document

An interactive application to support disaster management activities during the prevention and preparedness phase

Junjie Mu, Chaotung Wu, Jiayi Su, Jianwei Deng

| | |
|-----------------------|---|
| Deliverable: | SRD |
| Title: | Software Release Document |
| Authors: | Junjie Mu, Chaotung Wu, Jiayi Su, Jianwei Deng |
| Version: | 1.0 |
| Date: | June 28, 2024 |
| Download page: | https://github.com/Junjie-Mu/SE4GEO |
| Copyright: | Copyright © 2024, M.W.S.D – All rights reserved |

content

| | |
|---|----|
| 1. Introduction | 5 |
| 1.1 Purpose of the Document | 5 |
| 1.2 Overview of the Software | 5 |
| 2. System Requirements | 5 |
| 2.1 Hardware requirements | 5 |
| 2.2 Software dependencies (operating system, libraries, etc.) | 5 |
| 2.3 Network requirements (if applicable) | 8 |
| 3. Installation and Running | 9 |
| 3.1 Step-by-step installation instructions | 9 |
| 3.2 Basic usage instructions | 18 |
| 4. Known Limitations | 25 |
| 4.1 List of known issues or limitations | 25 |
| 4.2 Workarounds | 26 |
| 4.3 Future updates or fixes planned | 27 |

List of Table

| | |
|--|----|
| Figure 1 : webserver.py | 9 |
| Figure 2 : main page | 10 |
| Figure 3 :Set database | 10 |
| Figure 4 :connection is successful | 11 |
| Figure 5 :connection fails | 11 |
| Figure 6 :create the necessary database tables | 12 |
| Figure 7 :Register Now | 13 |

| | |
|--|----|
| Figure 8 :Log in | 13 |
| Figure 9 :wrong credentials | 13 |
| Figure 10 : Begin Now | 14 |
| Figure 11 : Create a new database | 15 |
| Figure 12 : PostGIS extension | 15 |
| Figure 13 : region data | 16 |
| Figure 14 : PIR data | 16 |
| Figure 15 : province data | 16 |
| Figure 16 : Import OSM Map Data | 17 |
| Figure 17 : region boundaries | 17 |
| Figure 18 : province boundaries | 18 |
| Figure 19 : Go to Dashboard | 18 |
| Figure 20 : scale | 19 |
| Figure 21 : disaster | 19 |
| Figure 22 : type | 20 |
| Figure 23 : Click the "search" button | 20 |
| Figure 24 : Visualization | 21 |
| Figure 25 : base map change | 22 |
| Figure 26 : Hierarchical Statistical Map | 23 |
| Figure 27 : Colormap | 23 |
| Figure 28 : scale and type | 24 |
| Figure 29 : Heatmap Data Visualization | 25 |

1. Introduction

1.1 Purpose of the Document

The purpose of this document is to provide a comprehensive overview of the web application designed for visualizing and analyzing environmental data from the Italian Institute for Environmental Protection and Research (ISPRA). This document details the system's architecture, functionality, and user interface, ensuring that stakeholders and developers have a clear understanding of the system's design and capabilities.

1.2 Overview of the Software

The software is an interactive web application that leverages spatial data to present users with dynamic and customizable visualizations. It integrates various tools and libraries to enable users to explore environmental data through maps, charts, and other graphical representations. The application is built on a client-server architecture, utilizing PostgreSQL with PostGIS for data storage, Flask for the backend web server, and Jupyter Notebooks for the frontend dashboard. This architecture ensures efficient data processing and an engaging user experience.

2. System Requirements

2.1 Hardware requirements

Building a Web GIS (Geographic Information System) requires a combination of server hardware, networking components, and client-side hardware to ensure efficient processing, storage, and delivery of geospatial data.

Here are the hardware requirements:

2.1.1 Server Hardware

Processor (CPU):

Multi-core processors (e.g., Intel Xeon or AMD EPYC) to handle concurrent requests and geospatial computations efficiently.

Memory (RAM):

At least 32GB of RAM for small to medium-sized applications.

64GB or more for large datasets and high user concurrency.

Storage:

SSDs (Solid State Drives) for fast data access and retrieval.

High-capacity storage (1TB or more) depending on the size of geospatial data.

Graphics Processing Unit (GPU):

Optional but beneficial for rendering complex maps and performing advanced spatial analysis.

4. Network Interface Card (NIC):

High-speed network interfaces (1 Gbps or higher) to handle data transfer and communication between servers and clients.

2.2 Software dependencies

Overall, a Web GIS can efficiently manage and serve geospatial data, providing robust and interactive mapping capabilities to users on terms of integrating these software dependencies below.

2.2.1 Operating System

Server Operating Systems:

Linux: Preferred for stability, security, and performance.

Windows Server: An option if using Microsoft-centric technologies.

2.2.2 GIS Server Software

the Flask server:

An effective middleware that connects users to the spatial data and services provided by the GIS server, offering Web Framework, front-end interfaces, Scalability, and so on.

2.2.3 Databases

PostgreSQL with PostGIS:

An open-source Database Management System, which uses a client-server model where the server program manages the database files and accepts connections to the database from client applications.

SE4G:

Creating a specific database supports all of the information of the website.

2.2.4 GIS Libraries and APIs

Basemap:

To build an interactive map which enhances features and ongoing support such as Map Plotting, Geospatial Data Visualization, Integration with Matplotlib, and Handling Geographic Features.

Data and color map:

Using color maps to represent different data values visually, enhancing data interpretation.

Heatmap:

Displaying the concentration of data points in a given area, highlighting patterns or hotspots.

Idro_GEO API:

Filtering and customizing queries to extract specific data relevant to user needs.

2.3 Network requirements

A Web GIS system can provide reliable, high-performance access to geospatial data, ensuring a smooth and secure user experience by addressing these network requirements.

2.3.1 Internet Connectivity

High-Speed Internet Connection:

Ensure a reliable high-speed internet connection (minimum 1 Gbps) for both server and client connections to handle data transfers efficiently.

Redundant Internet Connections:

Use multiple internet service providers (ISPs) for redundancy to ensure continuous availability in case one connection fails.

3. Installation and Running

3.1 Step-by-step installation instructions

Find the Folder:

Click the following link to find the software folder:

[download](#)

Open the Folder and run the file:

open the folder and find the file called " webserver.py ", double-click the file to start the server.

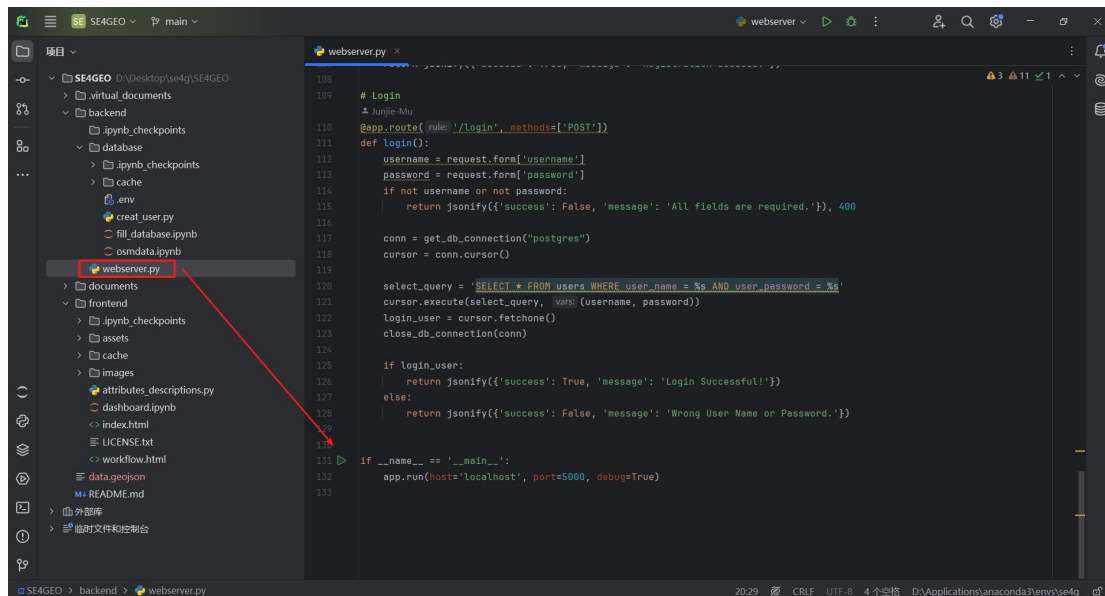


Figure 1: webserver.py

Open the Browser:

In your browser's address bar, type " <http://localhost:5000> " and press Enter to access the main page of the software

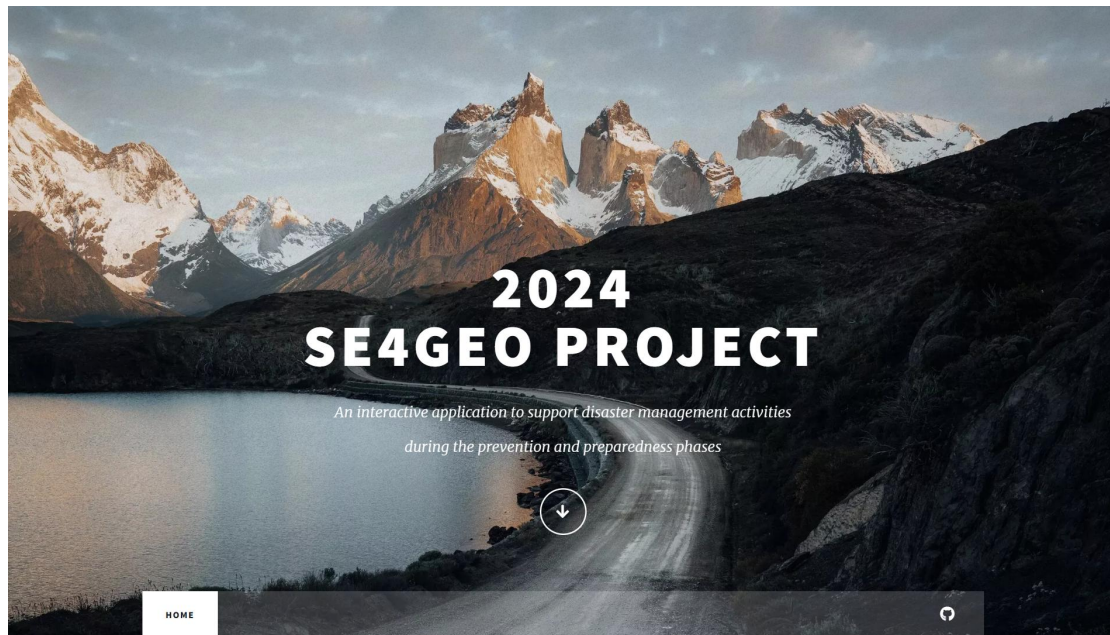


Figure 2: main page

step 1 : Set Database Parameters

On this page, click the "Set " button. Follow the prompts to enter the database information.

Figure 3:Set database

Restart the Flask server. Click the "Test Database Connection" button.

If the connection is successful, proceed to the next step.

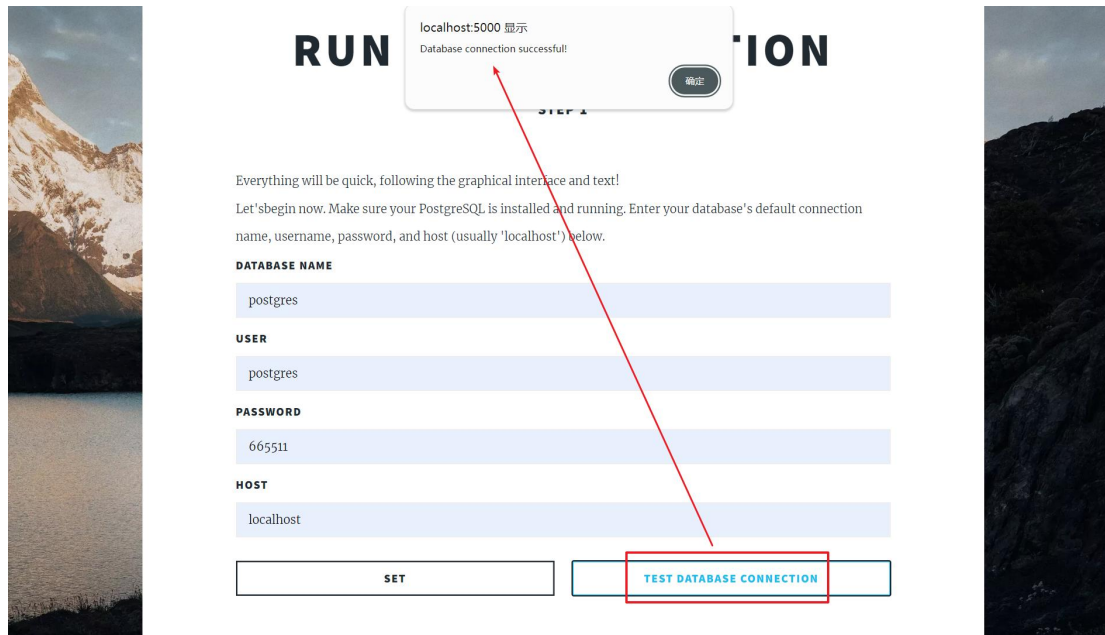


Figure 4:connection is successful

If the connection fails, check the database parameters and try again.

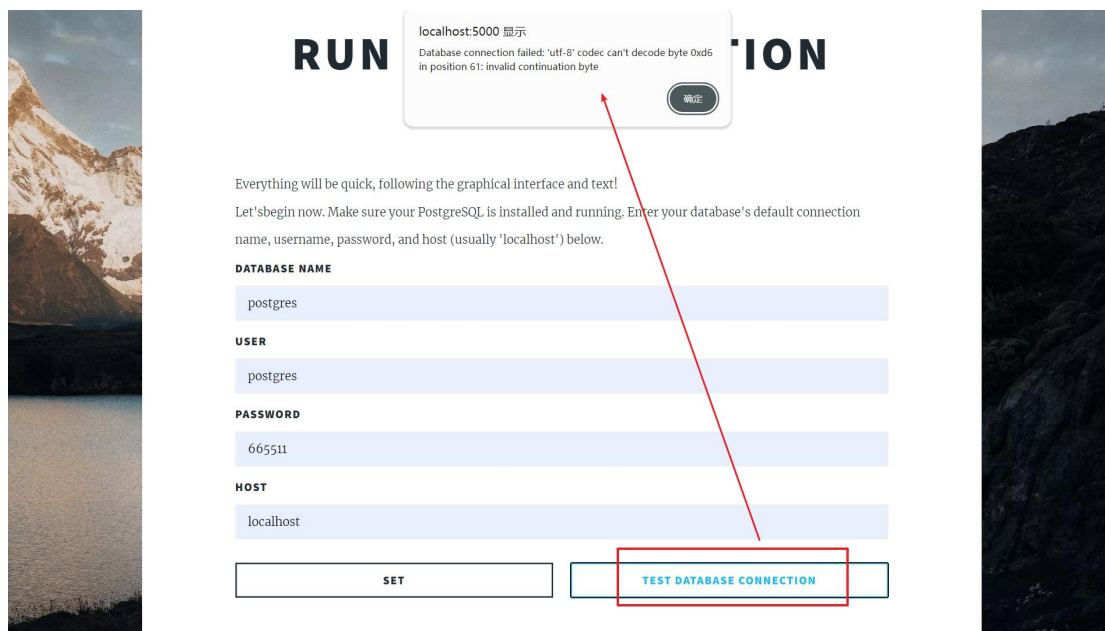


Figure 5:connection fails

step 2. Create Database Tables

Click the "Create" button on the main page. The system will automatically create the necessary database tables.



Figure 6:create the necessary database tables

step 3. User Login

Click the "Login Now" button to open the login interface.

If this is your first time using the software, click the "Register Now" button at the bottom of the login interface.

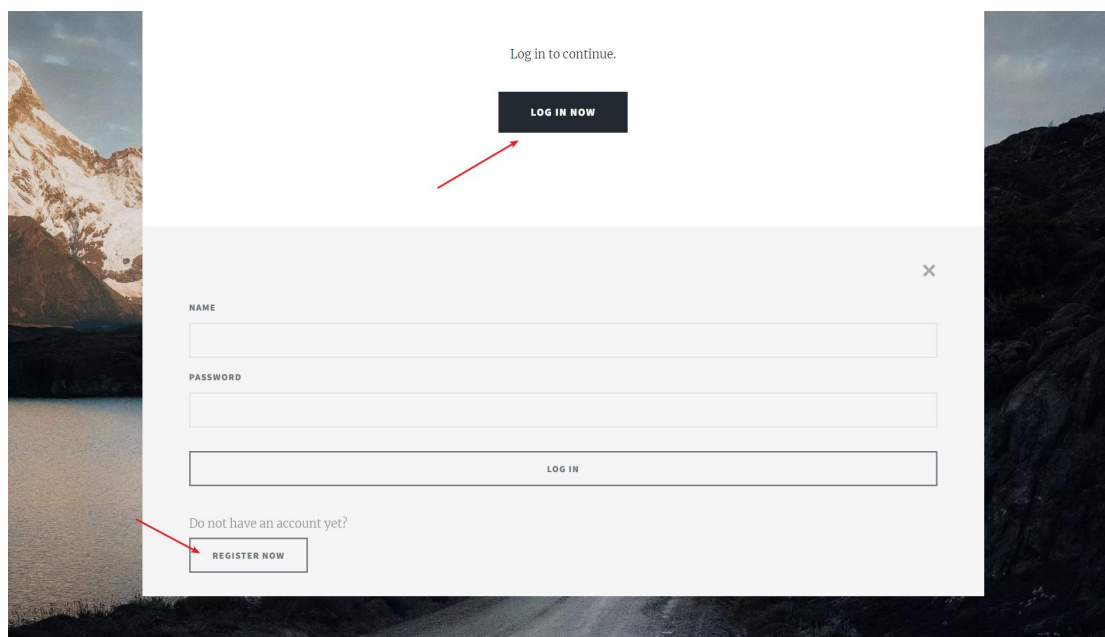


Figure 7:Register Now

Follow the prompts to complete the registration.

Log in using the username and password you just set.

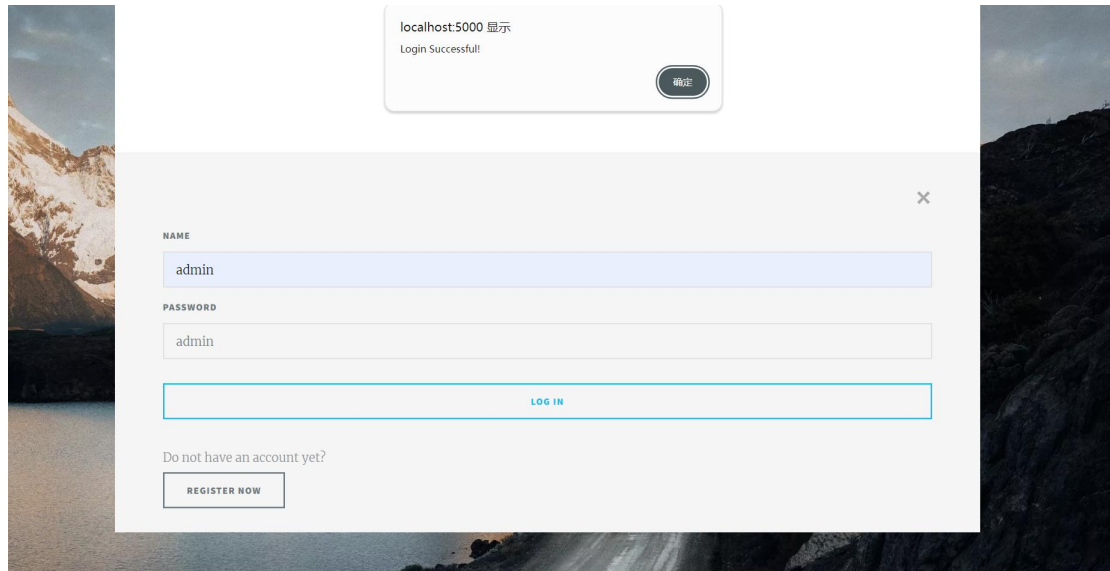


Figure 8:Log in

If you enter the wrong credentials, the system will display an error message.

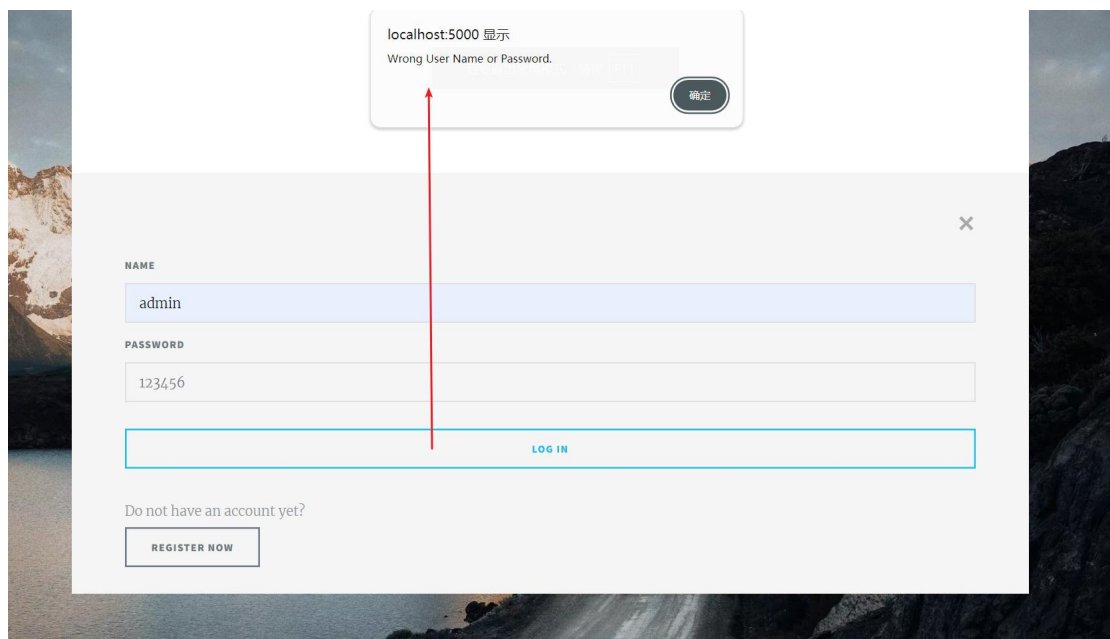


Figure 9:wrong credentials

Upon successful login, the system will automatically redirect you to the workflow page, indicating that you are logged in.

Click the "Logout" button to log out and return to the main page.

step 4. Initialize Database and Configuration

After logging in, on the workflow page, click the "Begin Now" button to open Jupyter Notebook.

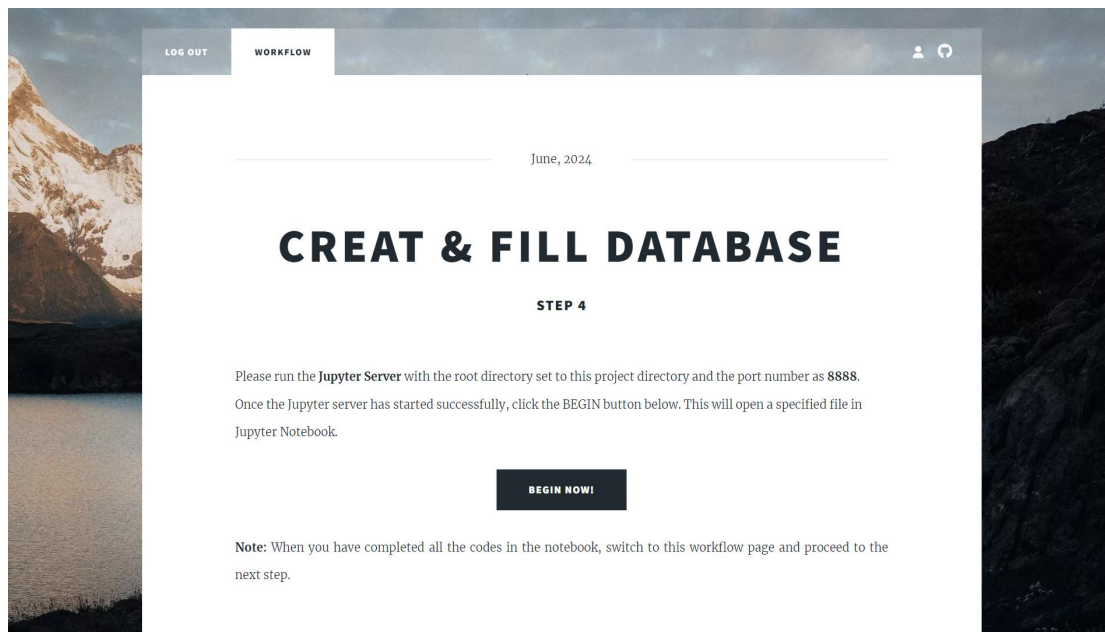
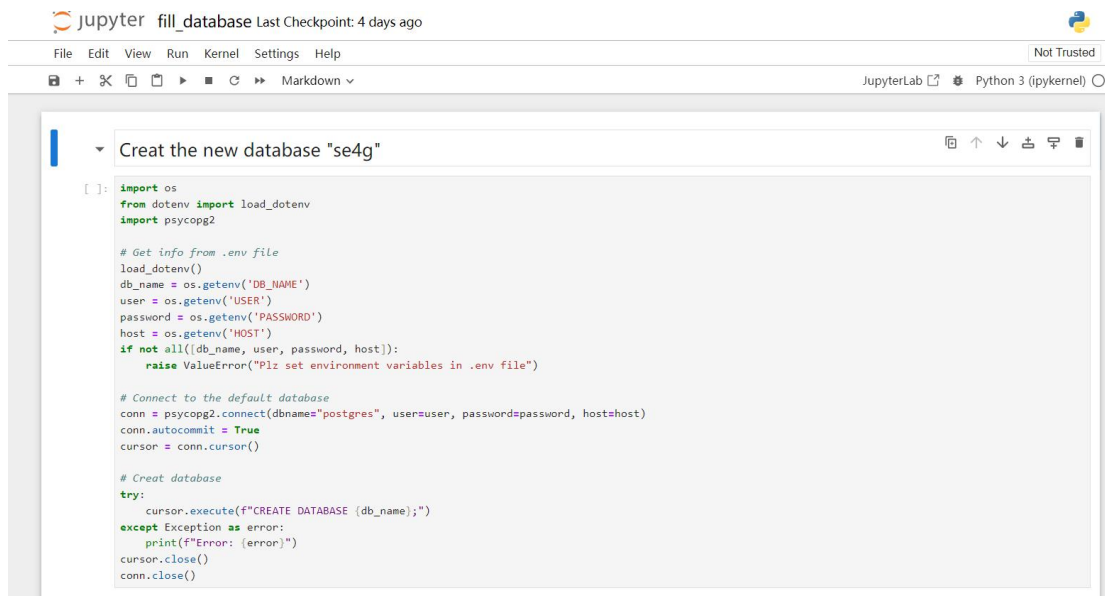


Figure 10: Begin Now

In Jupyter Notebook, execute the following steps:

Create a new database "SE4G" .



The image shows a JupyterLab interface with a code cell titled "Creat the new database 'se4g'". The code is as follows:

```
[ ]: import os
from dotenv import load_dotenv
import psycopg2

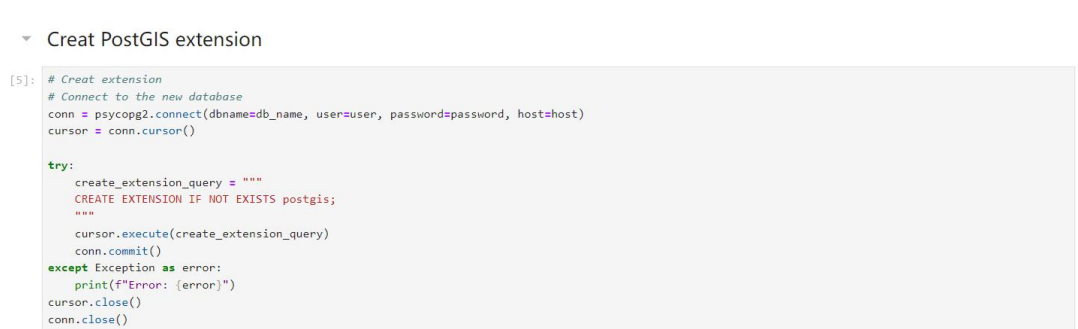
# Get info from .env file
load_dotenv()
db_name = os.getenv('DB_NAME')
user = os.getenv('USER')
password = os.getenv('PASSWORD')
host = os.getenv('HOST')
if not all([db_name, user, password, host]):
    raise ValueError("Plz set environment variables in .env file")

# Connect to the default database
conn = psycopg2.connect(dbname="postgres", user=user, password=password, host=host)
conn.autocommit = True
cursor = conn.cursor()

# Create database
try:
    cursor.execute(f"CREATE DATABASE {db_name};")
except Exception as error:
    print(f"Error: {error}")
cursor.close()
conn.close()
```

Figure 11: Create a new database

Create the PostGIS extension.



The image shows a JupyterLab interface with a code cell titled "Creat PostGIS extension". The code is as follows:

```
[5]: # Creat extension
# Connect to the new database
conn = psycopg2.connect(dbname=db_name, user=user, password=password, host=host)
cursor = conn.cursor()

try:
    create_extension_query = """
    CREATE EXTENSION IF NOT EXISTS postgis;
    """
    cursor.execute(create_extension_query)
    conn.commit()
except Exception as error:
    print(f"Error: {error}")
cursor.close()
conn.close()
```

Figure 12: PostGIS extension

Insert Idro_GEO API region data.

▼ Insert Idro_GEO API region data

```
[4]: import requests

IdroGEOAPI = os.getenv("IDROGEO_API")
regionAPI = IdroGEOAPI + "/regioni"
response = requests.get(regionAPI)
if response.status_code == 200:
    data = response.json()
else:
    print(f"Error: {response.status_code}")

try:
    conn = psycopg2.connect(dbname=db_name, user=user, password=password, host=host)
    cursor = conn.cursor()
    #Creat table
    creat_query = """
    CREATE TABLE IF NOT EXISTS regioni (
        uid INTEGER PRIMARY KEY,
        nome VARCHAR(100),
        osmid INTEGER
    );
    """
    cursor.execute(creat_query)
    conn.commit()
    #Insert data
    insert_query = """
    INSERT INTO regioni (uid, nome, osmid)
    VALUES (%s, %s, %s);
    """
    for item in data:
```

Figure 13: region data

Insert region PIR data(flood&landslide).

▼ Insert region PIR data(flood&landslide)

```
[5]: try:
    conn = psycopg2.connect(dbname=db_name, user=user, password=password, host=host)
    cursor = conn.cursor()
    # Flood table
    creat_query = """
    CREATE TABLE IF NOT EXISTS regioni_flood (
        uid INTEGER PRIMARY KEY,
        osmid INTEGER,
        nome VARCHAR(100),
        ar_kmq NUMERIC,
        ar_id_p3 NUMERIC,
        ar_id_p2 NUMERIC,
    """
```

Figure 14: PIR data

Insert Idro_GEO API province data.

```
Insert Idro_GEO API province data

[6]: provinceAPI = IdroGEOAPI + "/province"
response = requests.get(provinceAPI)
if response.status_code == 200:
    data = response.json()
else:
    print(f"Error: {response.status_code}")

try:
    conn = psycopg2.connect(dbname=db_name, user=user, password=password, host=host)
    cursor = conn.cursor()
    #Creat table
    creat_query = """
    CREATE TABLE IF NOT EXISTS province (
        uid INTEGER PRIMARY KEY,
        nome VARCHAR(100),
        osmid INTEGER,
        cod_reg INTEGER
    );
    """
```

Figure 15: province data

step 5. Import OSM Map Data

Return to the workflow page and click the "Begin Now" button again to execute the following tasks:

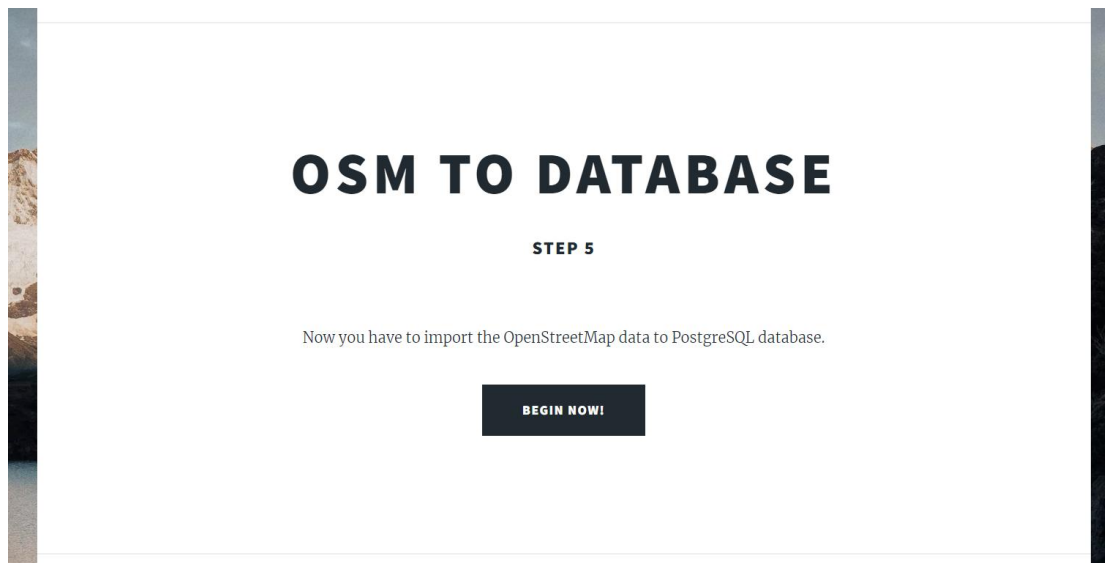


Figure 16: Import OSM Map Data

Import OSM map data into the database.

get the region boundaries.

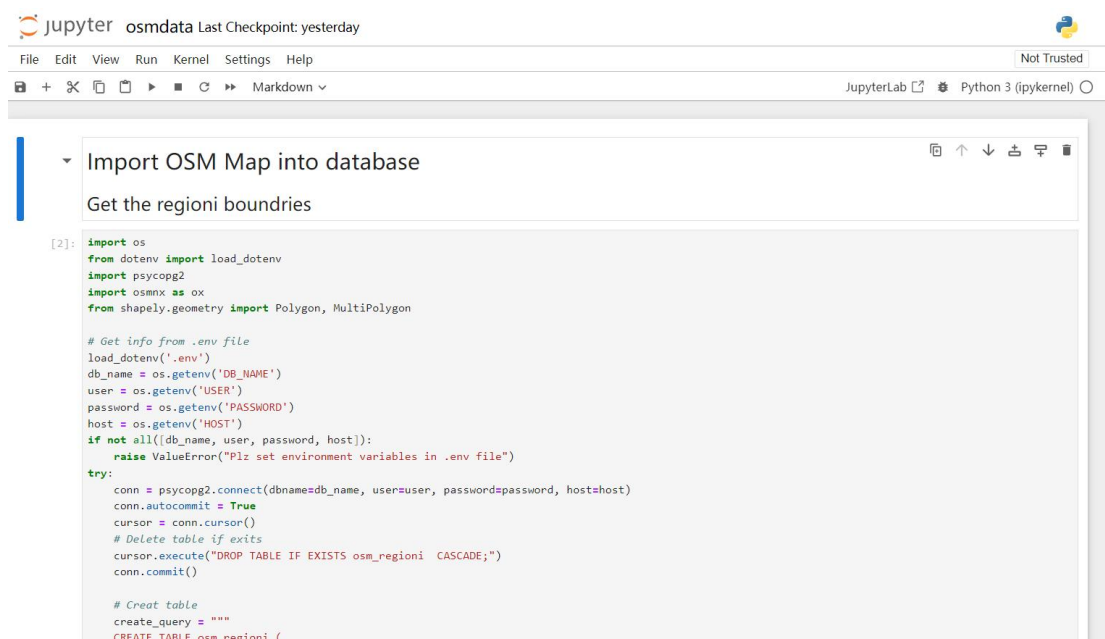


Figure 17: region boundaries

get the province boundaries.

Get the province boundaries

```
[4]: import psycopg2
import osmnx as ox
from shapely import wkb
from shapely.geometry import Polygon, MultiPolygon

try:
    conn = psycopg2.connect(dbname=db_name, user=user, password=password, host=host)
    conn.autocommit = True
    cursor = conn.cursor()
    # Delete table if exists
    cursor.execute("DROP TABLE IF EXISTS osm_province CASCADE;")
    conn.commit()

    # Create table
    create_query = """
    CREATE TABLE osm_province (
        id SERIAL PRIMARY KEY,
        nome VARCHAR(255),
        geom GEOMETRY(MultiPolygon, 4326),
        osmid INTEGER,
        lat NUMERIC,
        lon NUMERIC
    );
    """
    cursor.execute(create_query)
    conn.commit()
```

Figure 18: province boundaries

step 6. After completing these tasks, click the "Go to Dashboard" button to end the configuration process.

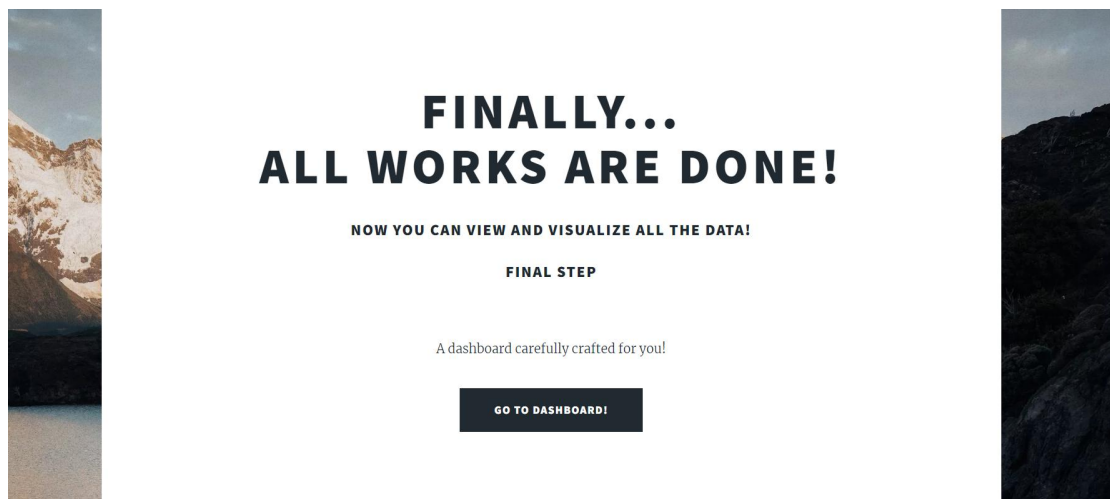


Figure 19: Go to Dashboard

After configuration, you can start using the software's various features for data analysis and management.

3.2 Basic usage instructions

This section provides a detailed guide on how to use the core functionalities of the software.

Functionality 1: Specified Condition Data Query

This functionality allows users to retrieve and visualize data based on specific conditions.

Select Query Scale : In the "scale" dropdown menu, select the desired query scale (e.g. regioni, province).



Figure 20: scale

Select Disaster Type : In the "disaster" dropdown menu, choose the type of disaster (e.g., flood, landslide).

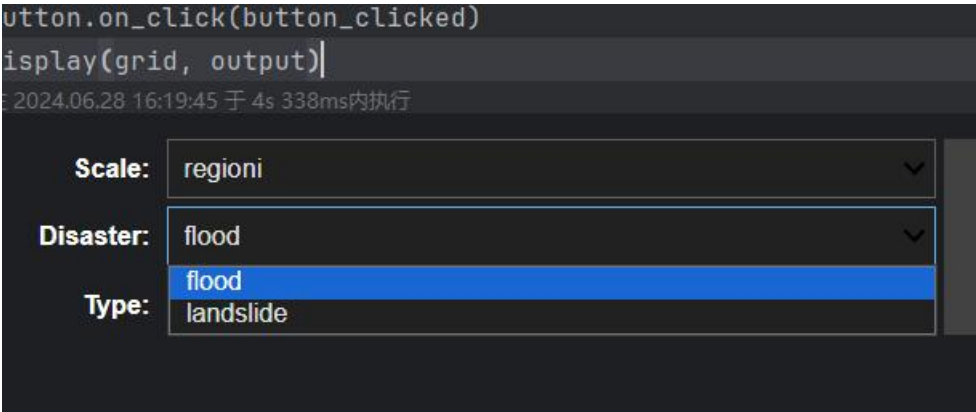


Figure 21: disaster

Select Affected Object Type: In the "type" dropdown menu, select the type of affected objects (e.g., surface area, population, families, buildings,local business, cultural heritage).

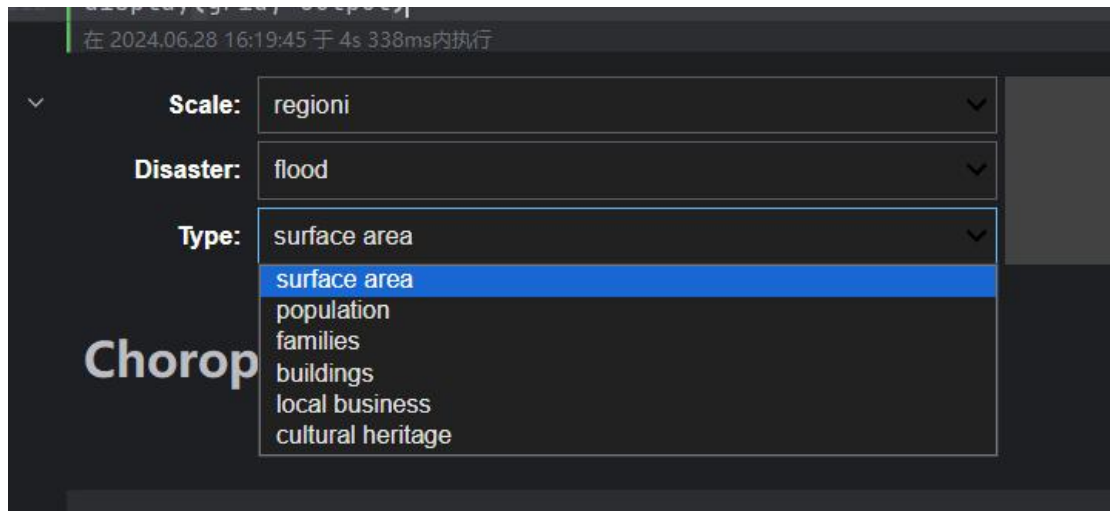


Figure 22: type

Search for Data: Click the "search" button.

The filtered map data will appear below the button, providing a visual representation of the data.



Figure 23: Click the "search" button

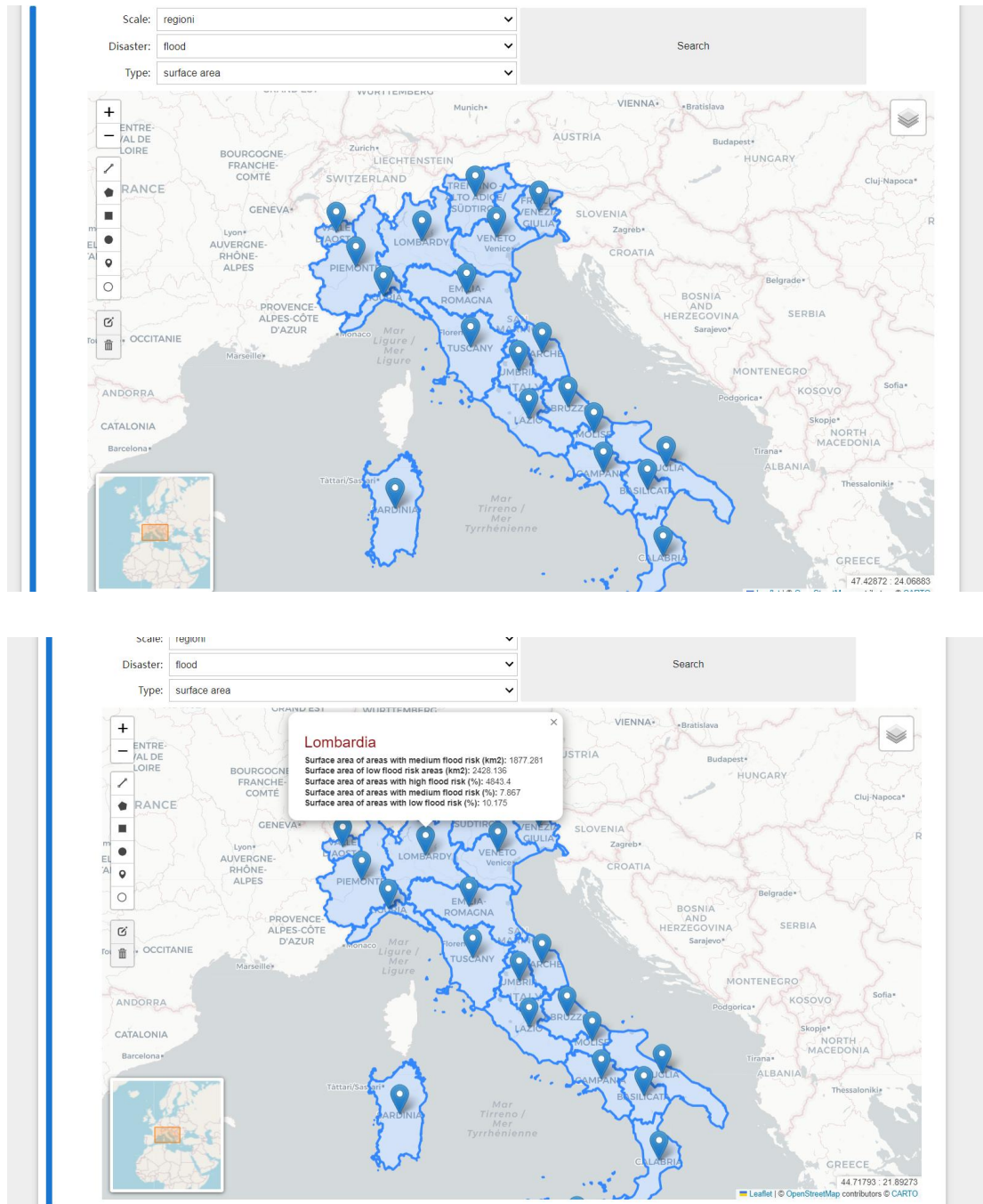


Figure 24: Visualization

Base Map Change: The map layer offers a "base map change" functionality. You can switch between the following map layers:

"cartodbpositron"

"openstreetmap"

"cartodbdark_matter"

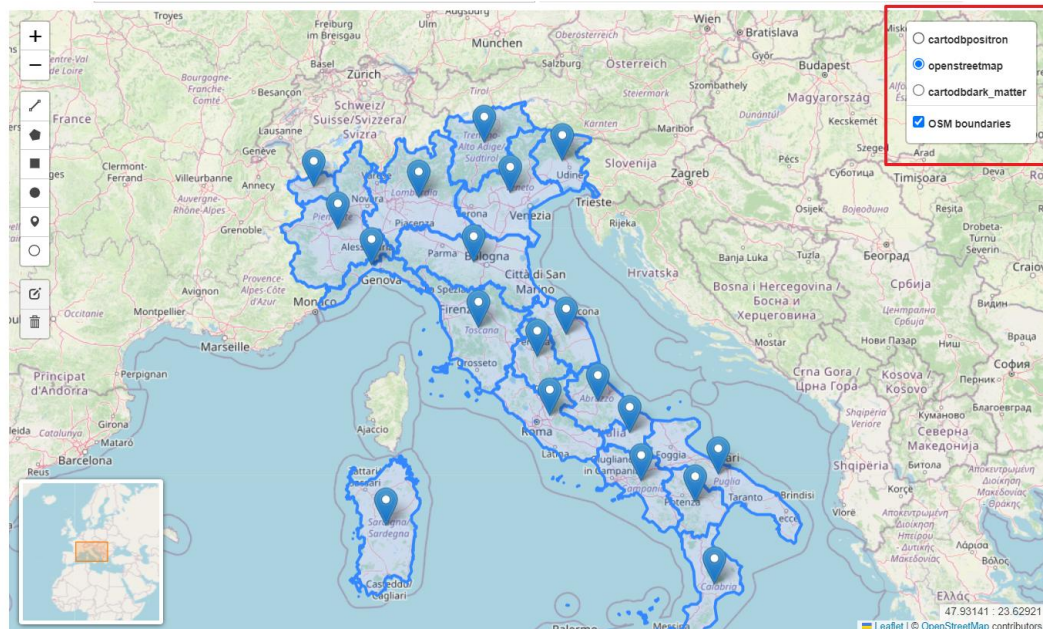


Figure 25: base map change

Functionality 2: Hierarchical Statistical Map

This functionality allows users to create a statistical map with hierarchical data representation.

Select Data Parameters: In the "scale" and "type" dropdown menus, select the desired parameters for your data query.

Search for Data: Click the "search data" button. A new interface will pop up.

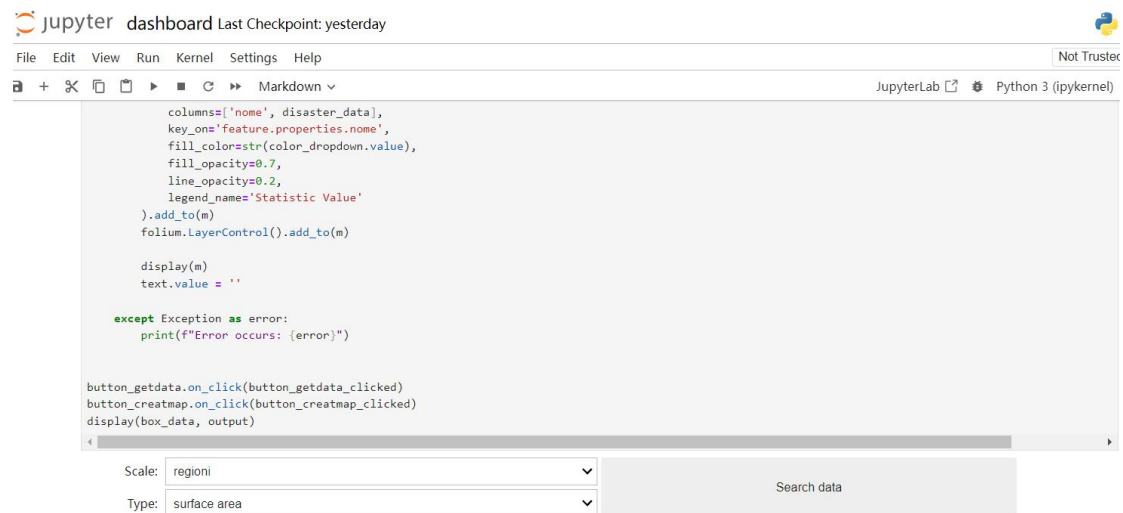


Figure 26: Hierarchical Statistical Map

Select Data and Colormap: In the popup interface, choose the desired "data" and "colormap".

Create Map: Click the "create map" button. The map layer will be displayed using the selected color scheme.

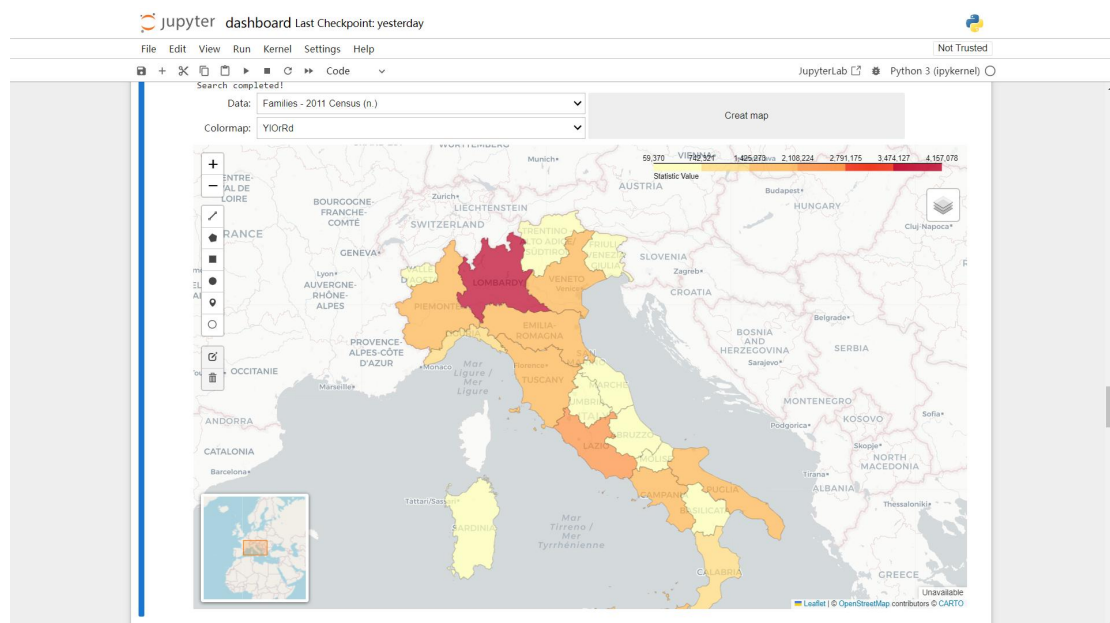


Figure 27: Colormap

Functionality 3: Heatmap Data Visualization

This functionality allows users to create a heatmap based on specified conditions.

Select Query Scale: In the "scale" dropdown menu, select the desired query scale.

Select Disaster Type: In the "type" dropdown menu, choose the disaster type.

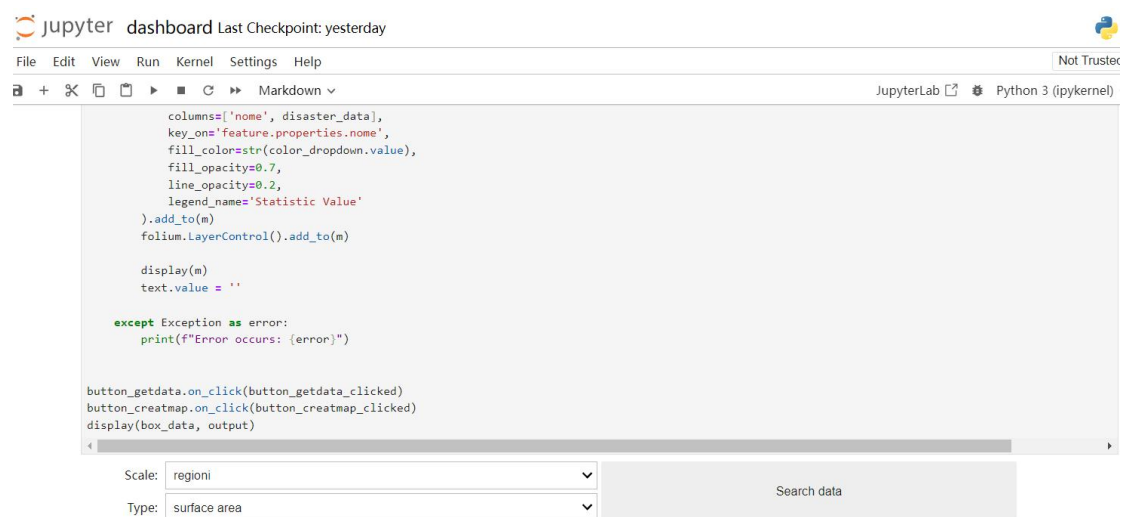


Figure 28: scale and type

Search for Data: After performing the query, a new dropdown menu "data" will appear.

Select Heatmap Data: In the "data" dropdown menu, select the input data for the heatmap.

Visualize Heatmap: The software will generate and display the heatmap based on the selected data.

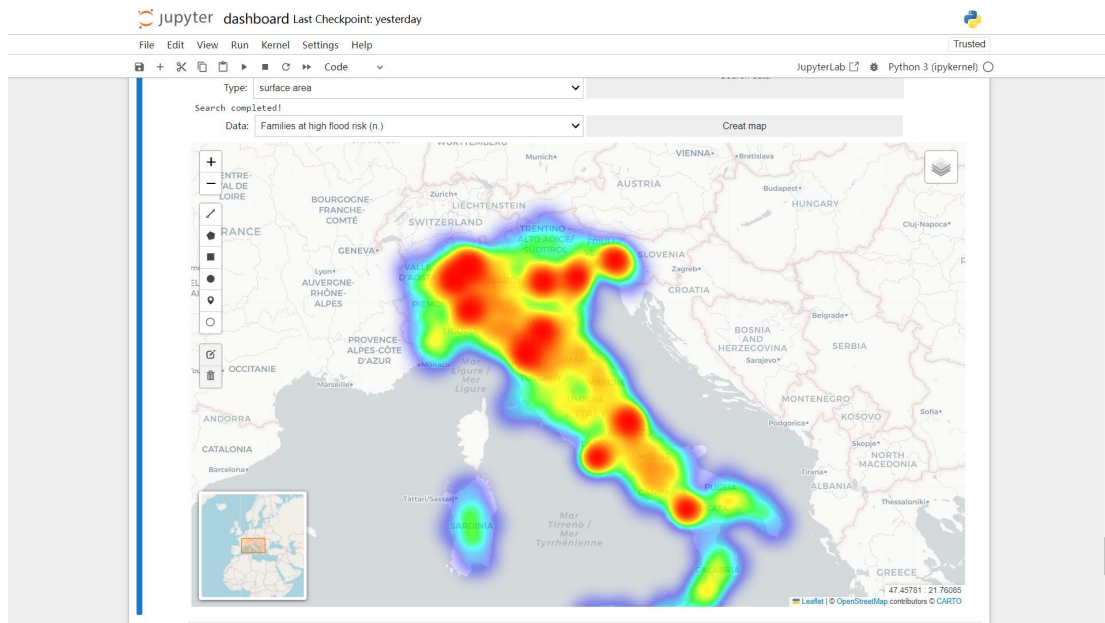


Figure 29: Heatmap Data Visualization

4. Known Limitations

4.1 List of known issues or limitations

4.1.1 Performance Issues with Large Datasets:

The system may experience latency or slow performance when handling very large datasets due to the complexity of spatial queries and data processing.

4.1.2 Browser Compatibility:

Some features may not be fully supported on all web browsers, potentially causing inconsistencies in user experience.

4.1.3 Limited Mobile Support:

The user interface is optimized for desktop use, and certain visualizations and interactions may not function properly on mobile devices.

4.1.4 Real-time Data Processing:

While the system requests and processes data in real time, there may be delays due to server load or network latency, affecting the immediacy of data updates.

4.1.5 User Authentication and Authorization:

The current implementation of user authentication may have vulnerabilities or limitations that could affect the security of user data.

4.1.6 Geospatial Analysis Limitations:

The geospatial analysis capabilities, while robust, may not cover all possible use cases or complex spatial operations required by advanced users.

4.2 Workarounds

4.2.1 Performance Issues with Large Datasets:

- **Data Filtering:** Users can apply data filters to limit the amount of data processed at a time.
- **Incremental Loading:** Implement incremental data loading where only a subset of data is processed initially, with more data loaded as needed.

4.2.2 Browser Compatibility:

- **Recommended Browsers:** Users are advised to use modern browsers such as Chrome, Firefox, or Edge for the best experience.
- **Polyfills and Shims:** Implement polyfills and shims to improve compatibility with older browsers.

4.2.3 Limited Mobile Support:

- **Responsive Design:** Future updates can include enhanced responsive design techniques to improve mobile compatibility.

4.2.4Real-time Data Processing:

- **Caching Mechanisms:** Utilize caching to reduce the need for repeated real-time data requests.
- **Server Optimization:** Optimize server performance to handle data requests more efficiently.

4.2.5User Authentication and Authorization:

- **Regular Security Audits:** Conduct regular security audits and update the authentication mechanism to address vulnerabilities.
- **Two-Factor Authentication:** Implement two-factor authentication to enhance security.

4.2.6Geospatial Analysis Limitations:

- **Custom Scripts:** Advanced users can use custom Python scripts within Jupyter Notebooks for more complex geospatial analyses not supported by the default tools.

4.3 Future updates or fixes planned

4.3.1Enhanced Performance:

Future updates will focus on optimizing data processing algorithms and improving the performance of spatial queries to handle larger datasets more efficiently.

4.3.2Cross-Browser Compatibility:

Efforts will be made to ensure full functionality across all major web browsers, including improvements to the user interface to provide a consistent experience.

4.3.3Mobile Support:

The user interface will be redesigned to ensure better compatibility and usability on mobile devices, including touch-friendly interactions.

4.3.4Real-time Data Improvements:

Enhancements will be made to reduce latency and improve the responsiveness of real-time data updates.

4.3.5 Security Enhancements:

Regular updates will be released to address security vulnerabilities, including the implementation of stronger authentication mechanisms and data protection measures.

4.3.6 Expanded Geospatial Analysis:

Additional geospatial analysis tools and functionalities will be integrated to cater to more advanced user requirements.

4.3.7 User Feedback Integration:

Ongoing updates will incorporate user feedback to address common issues and enhance the overall user experience.