POLITECNICO
MILANO 1863

# Design Document

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

| | |
|---|---|
| **Deliverable:** | DD |
| **Title:** | Design Document |
| **Authors:** | Junjie Mu, Chaotung Wu, Jiayi Su, Jianwei Deng |
| **Version:** | 2.0 |
| **Date:** | May 28, 2024 |
| **Download page:** | https://github.com/Junjie-Mu/SE4GEO |
| **Copyright:** | Copyright © 2024, M.W.S.D – All rights reserved |

## Revision history

| **Version** | **Date** | **Change** |
|---|---|---|
| 1.0 | May 30, 2024 | First submitted version |
| 2.0 | June 28, 2024 | Revised version |

# Table of Contents

## List of Tables

## List of figures

# 1. Introduction

## 1.1 Design Document

Before starting to code and implement the software, an important step is to have a clear understanding of the design goals that our Web application will achieve. All of these goals should be established in this particular design document.

The key reason for writing this document is to define the structure and organization that will continue throughout the creation period until the software project is finalized. This is an important guide document for the engineering team, which they can rely on for any subsequent steps.

A software design document is a technical description that serves both the developer and the client. The developer's goal is to meet the customer's goals, but at the same time help themselves provide an efficient workflow.

It is important to emphasize the fact that the Design Document is built upon the (RASD) Requirements Analysis Specification Document, written by Junjie Mu, Chaotung Wu, Jiayi Su, Jianwei Deng. You can find this document in the following GitHub link.

The connection between the above-mentioned documents helps us to be careful with not avoiding any important functionality.

The characteristics specified would not be changed or described in another way but will remain an implicit knowledge for the development team.

More specifically it will include the following characteristics:

**Project Database:**

This is the most important part of the project we are developing. Database design can be defined as a collection of tasks or processes that enhance the designing, development, and maintenance of data management system. With other words, it is the most complex part where the designer determines what data must be stored and how the data elements are connected and interacting with each other.

**System Overview:**

This section will provide a general description of the structure and functionality of the software system.

**Software Structure:**

Our software, as a traditional client-server application, will be established as a three-tier architecture. This structure will make possible the interaction between the client and the server, considering both static and dynamic units. According to this architecture the application is organized into three logical and physical layers or tiers, which are:

• Presentation Server

• Application Server

• Database Server

Further details will be explained in the corresponding section in the body of the document.

**User Case Application:**

This section will include the use cases and requirements map for each component of the software. Further details are prementioned in the RASD Document.

## 1.2 Product Description

The purpose of developing this product is to inform about visualize and elaborate the exposure at a specific Italian administrative level by means of an Open-Source well application. This web-application will allow users to query, visualize, analyze data, and save their analysis of the chosen data. On the website the user will be able to find general information regarding this environmental issue.

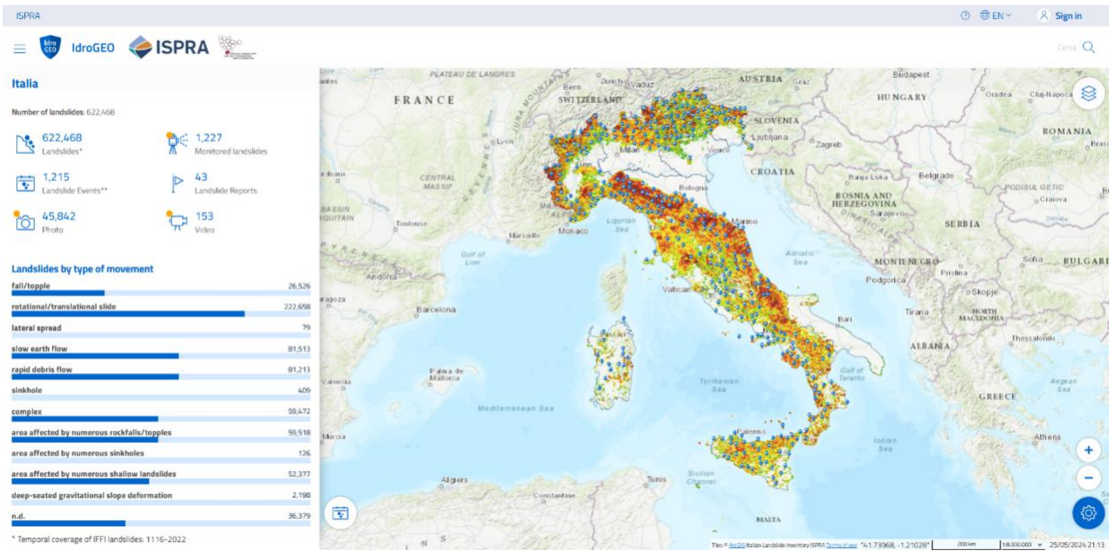
## 2. Project Database

### 2.1 ISPRA

The project database for mapping flood and landslide hazards in Italy leverages the IdroGEO API and ISPRA's open data to provide an interactive and comprehensive tool for visualizing geological risks. The IdroGEO API and ISPRA's open data initiatives represent significant advancements in the accessibility and usability of environmental data. By providing detailed and up-to-date information on geological hazards and other environmental parameters, these tools enable better decision-making and enhance our understanding of environmental dynamics.

Information on ISPRA datasets

Background Geodata



Italia

Number of landslides: 622,468

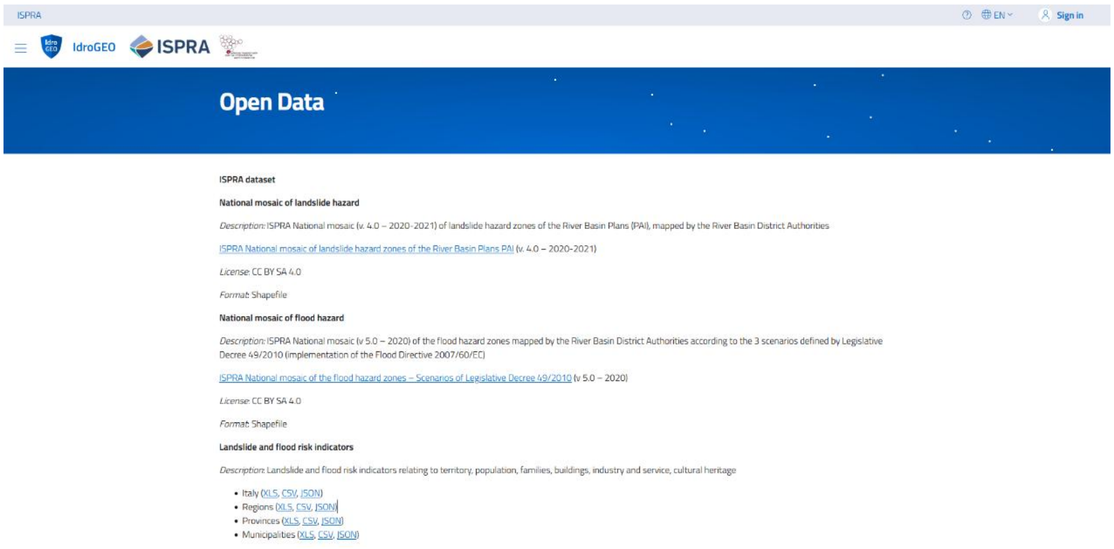
| | |
|---|---|
| 622,468 Landslides* | 1,227 Monitored landslides |
| 1,215 Landslide Events** | 43 Landslide Reports |
| 45,842 Photo | 153 Video |

Landslides by type of movement

| | |
|---|---|
| fall/topple | 26,526 |
| rotational/translational slide | 222,658 |
| lateral spread | 79 |
| slow earth flow | 81,513 |
| rapid debris flow | 81,213 |
| sinkhole | 409 |
| complex | 59,472 |
| area affected by numerous rockfalls/topples | 59,518 |
| area affected by numerous sinkholes | 126 |
| area affected by numerous shallow landslides | 52,377 |
| deep-seated gravitational slope deformation | 2,198 |
| n.d. | 36,379 |

* Temporal coverage of IFFI landslides: 1116-2022

Figure 1 The interface of Italian Landslide Inventory IFFI



Open Data

**ISPRA dataset**

**National mosaic of landslide hazard**

*Description:* ISPRA National mosaic (v. 4.0 – 2020-2021) of landslide hazard zones of the River Basin Plans (PAI), mapped by the River Basin District Authorities

ISPRA National mosaic of landslide hazard zones of the River Basin Plans PAI (v. 4.0 – 2020-2021)

*License:* CC BY SA 4.0

*Format:* Shapefile

**National mosaic of flood hazard**

*Description:* ISPRA National mosaic (v 5.0 – 2020) of the flood hazard zones mapped by the River Basin District Authorities according to the 3 scenarios defined by Legislative Decree 49/2010 (implementation of the Flood Directive 2007/60/EC)

ISPRA National mosaic of the flood hazard zones – Scenarios of Legislative Decree 49/2010 (v 5.0 – 2020)

*License:* CC BY SA 4.0

*Format:* Shapefile

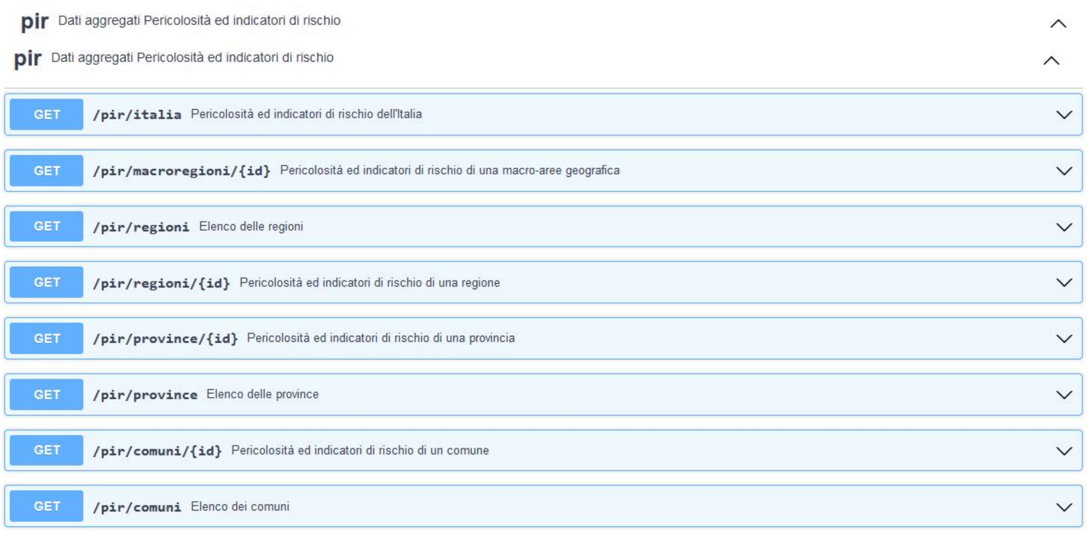
**Landslide and flood risk indicators**

*Description:* Landslide and flood risk indicators relating to territory, population, families, buildings, industry and service, cultural heritage

- Italy (XLS, CSV, JSON)
- Regions (XLS, CSV, JSON)
- Provinces (XLS, CSV, JSON)
- Municipalities (XLS, CSV, JSON)

Figure 2 ISPRA's open data

By leveraging PostgreSQL, the project database ensures reliable, efficient, and secure management of all data related to flood and landslide hazards in Italy. The app relies on PostgreSQL for robust data storage and management, leading to its reliability, scalability, and powerful feature set. These are essential for handling the dynamic and extensive datasets required by the platform.



pir Dati aggregati Pericolosità ed indicatori di rischio

pir Dati aggregati Pericolosità ed indicatori di rischio

```
GET /pir/italia Pericolosità ed indicatori di rischio dell'Italia
GET /pir/macroregioni/{id} Pericolosità ed indicatori di rischio di una macro-aree geografica
GET /pir/regioni Elenco delle regioni
GET /pir/regioni/{id} Pericolosità ed indicatori di rischio di una regione
GET /pir/province/{id} Pericolosità ed indicatori di rischio di una provincia
GET /pir/province Elenco delle province
GET /pir/comuni/{id} Pericolosità ed indicatori di rischio di un comune
GET /pir/comuni Elenco dei comuni
```

Figure 3 Structure of the API response

.

## 2.2 PostgreSQL Database

The data of the web app needs structured data management, which depends on PostgreSQL. It can efficiently store structured data from the IdroGEO API and ISPRA's open data, ensuring that all information on floods and landslides is organized and easily accessible. With its PostGIS extension, PostgreSQL can store and manage geospatial data, which is crucial for mapping and analyzing the geographic distribution of hazards according to the necessity of spatial data support:

| |
|---|
| <<Feature Type>><br><br>Province_flood/ls |
| uid: INTEGER<br><br>Osmid:INTEGER<br><br>nome: VARCHAR(100)<br><br>ar_kmq NUMERIC,<br><br>ar_id_p3 NUMERIC,<br><br>…<br><br>pop_adu_p NUMERIC,<br><br>pop_anz INTEGER,<br><br>im_idr_p3 INTEGER,<br><br>…<br><br>bbccidp3_p NUMERIC,<br><br>bbccidp2_p NUMERIC,<br><br>bbccidp1_p NUMERIC |

Table 1 Data Structure

| User |
|---|
| user_id: INTEGER<br>user_name: VARCHAR(255)<br>user_password:VARCHAR(255) |

Table 2 User table attributes definition

## 3. System Overview

The system which is based on disaster management is designed to support the prevention and preparedness phases of disaster management activities. It leverages geospatial data and advanced visualization tools to provide decision-makers, researchers, and the public with critical insights into natural hazards such as landslides and floods. By splitting into multiple highly interactive pages based on a client-server architecture.

The system is composed of three main components: a database, a web server, and an interactive dashboard.

### 3.1 System Architecture

The system is designed with a robust architecture to efficiently manage, process, and visualize geospatial data. Below are detailed descriptions of each component within the system:

#### 3.1.1 Database

**PostgreSQL with PostGIS Extension**

• Spatial Data Storage: The system uses PostgreSQL with the PostGIS extension to store and manage spatial data. This extension allows PostgreSQL to handle geographic objects and perform spatial queries efficiently.

• Data Source: The database holds information retrieved from the Italian Institute for Environmental Protection and Research (ISPRA) datasets, which include detailed hazards and risk indicators.

• Data Management: The database schema is designed to handle complex spatial data, with tables for different types of hazards, risk indicators, geographical boundaries, and related metadata.

### 3.1.2 Web Server

#### Flask Framework

- Backend Development: The backend of the application is built using Flask, a lightweight Python web framework known for its simplicity and flexibility.

- REST API: The web server exposes a REST API that allows the frontend to query the database and retrieve data. This API includes endpoints for various data operations, such as fetching hazard data, querying risk indicators.

### 3.1.3 Dashboard

#### Jupyter Notebooks

- Interactive Interface: The frontend is implemented using Jupyter Notebooks, providing an interactive interface for users. This setup allows for dynamic interaction with the data and seamless integration of code, text, and visualizations.

- Widgets and Visualization Tools: The dashboard includes various widgets and visualization tools to enable users to interact with the data through maps, charts, and other visual elements. Key libraries used include:

Folium: For visualizing geospatial data with Leaflet maps, making it easy to create interactive maps.

#### Dashboard Functionalities

- Data Visualization: Users can visualize data on maps, charts, and graphs, allowing for comprehensive analysis and understanding of the geospatial data.

- Interactive Maps: The dashboard supports interactive maps where users can pan, zoom, and switch between different map layers.

### 3.1.4 Additional Components

User Authentication: The system includes functionalities for user registration, login, and logout. Users must authenticate themselves to access certain features, such as the "workflow" page.

### 3.1.5 High level Architechture

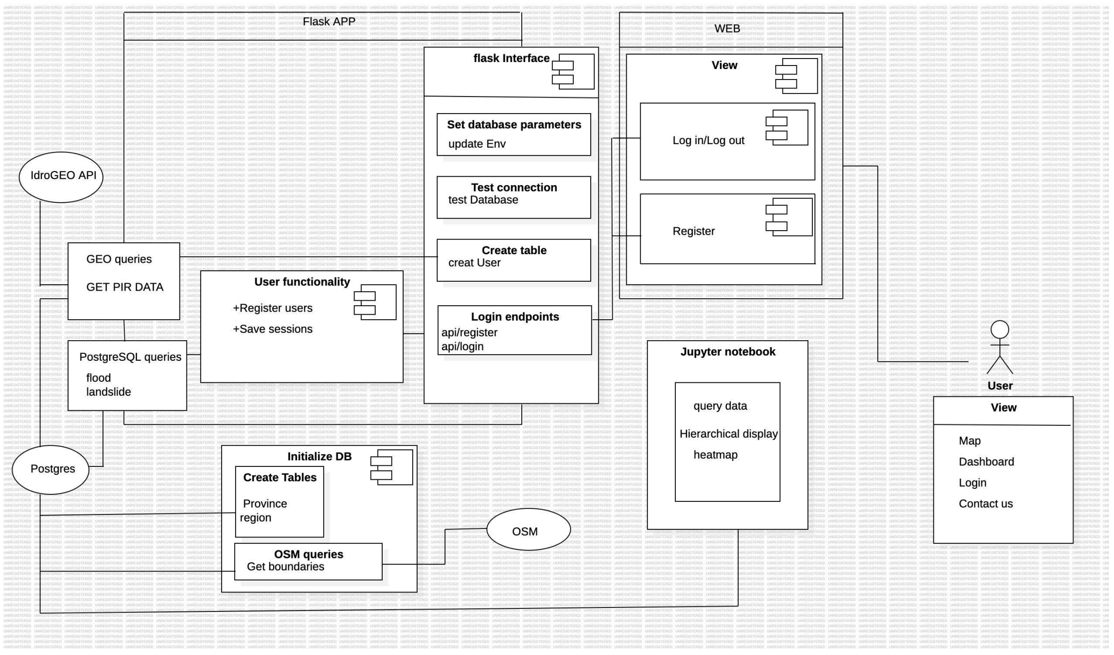


Figure 4：High level architecture of the web app

The diagram shows how these components are interconnected. The Flask APP interacts with the database (Postgres) and APIs (IdroGEO and OSM). The web interface allows users to register, log in, and view data, while the Jupyter Notebook is used for data analysis and visualization. The user can access various views such as maps and dashboards through the web interface.

## 3.2 Key Features

**Data Retrieval and Visualization:** Users can search for specific datasets using robust filters and search functionalities. The system provides interactive maps, charts, and graphs to present the data in an easily understandable manner.

**User Management:** The system includes functionalities for user registration and login.

### 3.3 Technologies Used

**Backend:** Flask (Python), PostgreSQL, PostGIS

**Frontend:** Jupyter Notebooks, ipywidgets, Folium

**Version Control:** Git, GitHub

**Development Methodology:** Agile (Scrum), Jira for project management

## 4. Software structure

The software's structure will be organized into three logical and physical computing tiers or system layers, which are:

- Presentation Server(User's PCs)
- Application Server
- Database Server

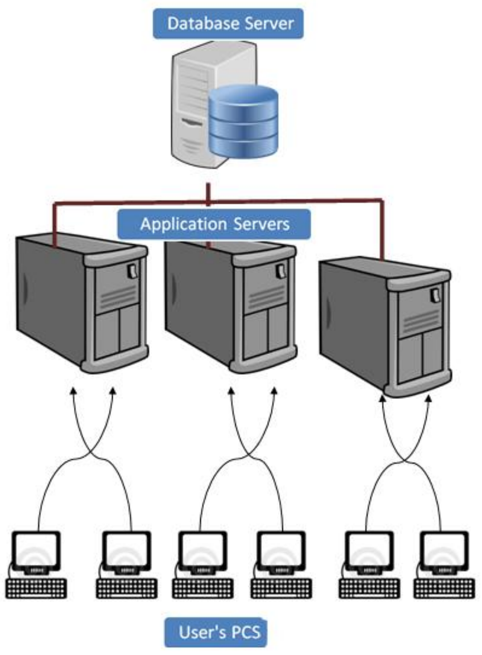


Figure 5: Software structure

### 4.1 Presentation Server

This top-level layer is the application's user interface, where users interact with the system. Its role is to present information to the users and simultaneously gather input from them.

The presentation layer operates within a web browser. Users can communicate with the HTTP server through actions like entering URLs, clicking, and filling out forms. The primary functions of the HTTP server are to store, process, and deliver web pages to the clients.

In response to user requests, the HTTP server will send a reply to the user's browser. This response will be crafted using HTML, CSS, JavaScript, and Python.

HTML: provides the structure of web pages

CSS: styles and layouts them

Javascript: interactivity and dynamic content.

## 4.2 Application Server

The application server, also known as the logic tier, handles all the necessary logical operations to meet the software requirements. It processes information gathered from the presentation tier to generate the required web pages. Additionally, the application tier can add, delete, or modify data in the data tier.

Python, along with SQL, will be used to develop the application server. Python will handle all the software's functions, interacting with the DBMS, managing operations between web pages, and controlling the interactive maps that are central to the application's display and analysis.

The essential libraries, categorized by function, are:

- **DBMS:** PostgreSQL
- **Web Development:** Flask
- **Data Analysis:** Pandas, GeoPandas (for data and geodata manipulation)

### 4.2.1 software function

**Connect to database:** All the credentials are contained in this file. Here the user can find the Database name, the password and the user name.

**Query data from database:** This function is to send query to get from database.

**Json data to Dataframe:** This function is used to create a Dataframe, starting from the json file.

### 4.3 Database Server

The data tier, also known as the database server or back-end, is where the information processed by the application is stored and managed. The web application will use PostgreSQL, a free and open-source database management system, to perform queries and select the data to be displayed.PostgreSQL will be integrated with Python, utilizing a database adapter to facilitate communication between the database and the Python programming language.

## 5. User interface

In this block we will analyze the design of the user interface, our aim is to create an intuitive and user-friendly interface. A well-organized web structure ensures easy navigation and access to key functions.

### 5.1 Homepage

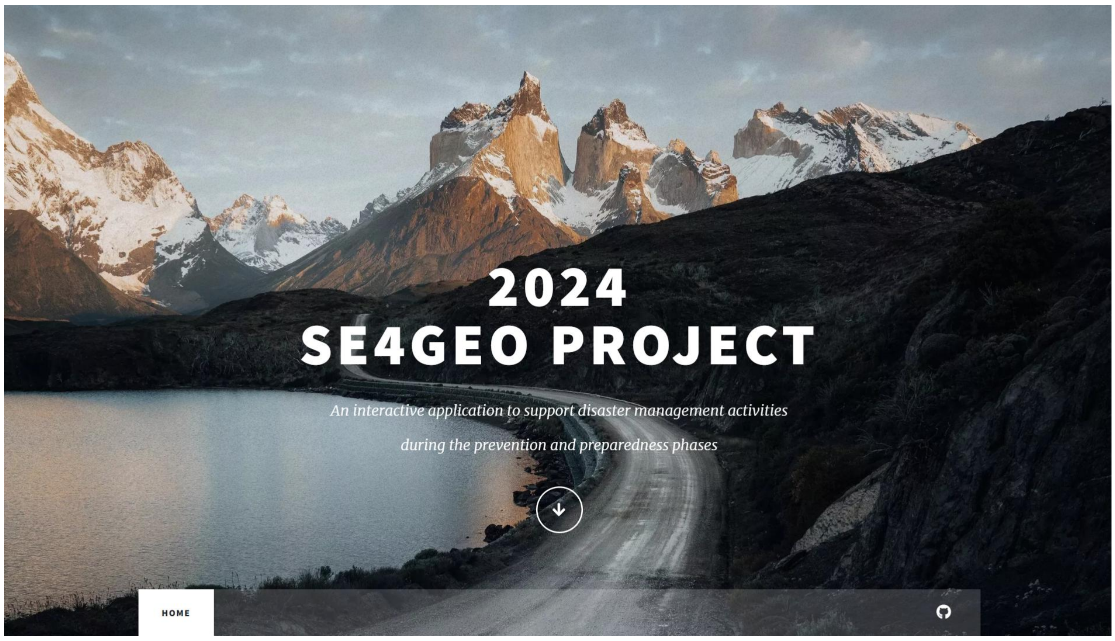
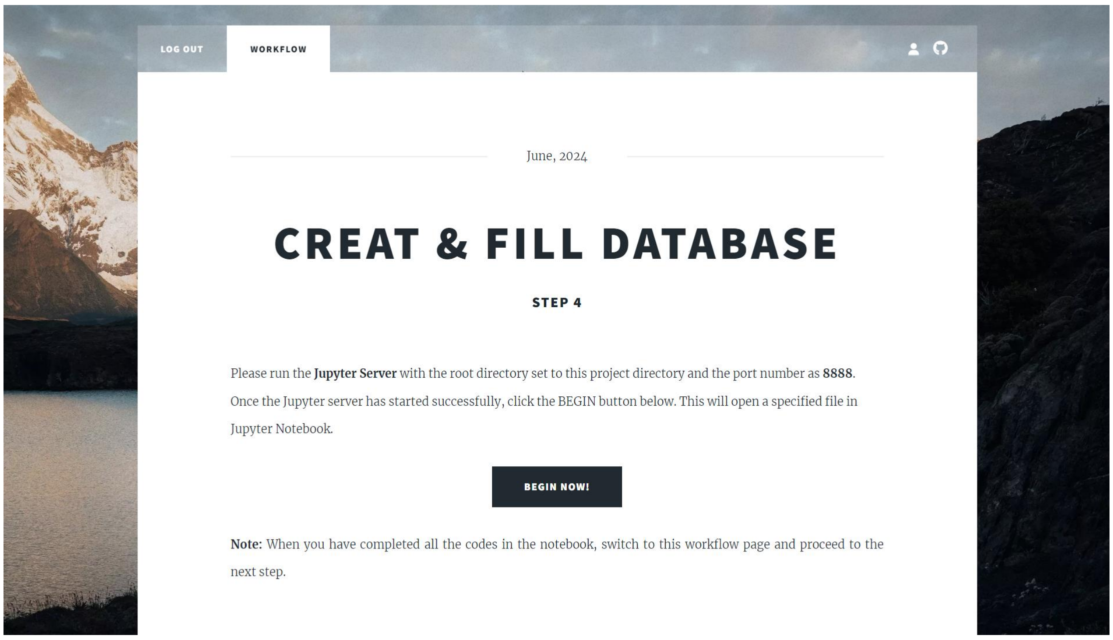

2024
SE4GEO PROJECT
An interactive application to support disaster management activities
during the prevention and preparedness phases
HOME

Figure 6: Home page

## 5.2 Workflow page



June, 2024

CREAT & FILL DATABASE

STEP 4

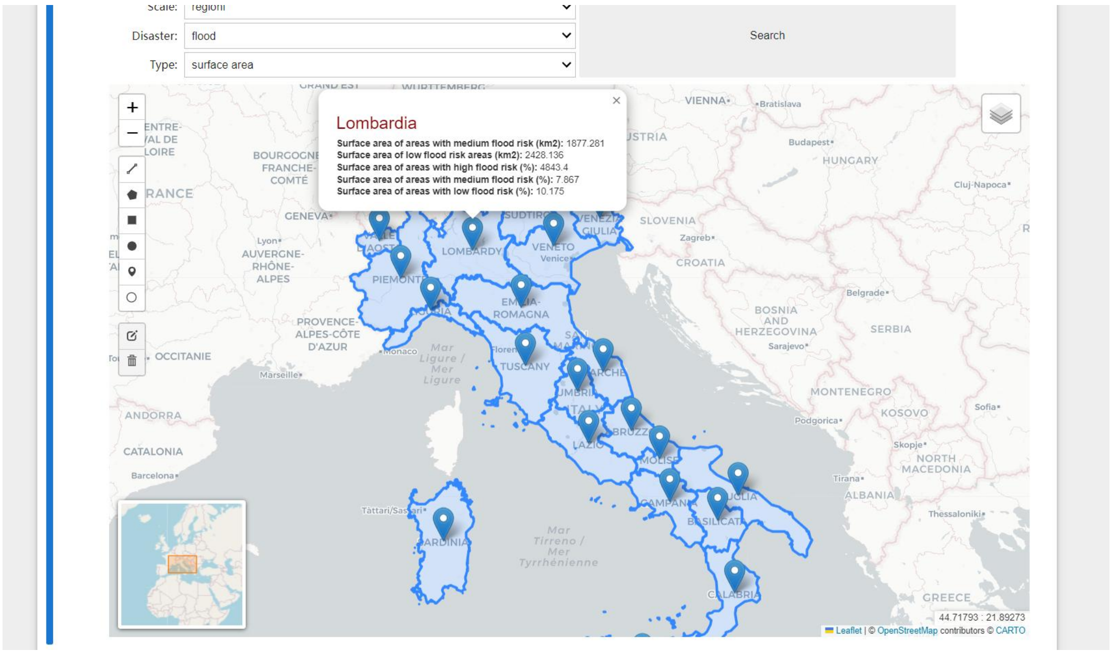
Please run the **Jupyter Server** with the root directory set to this project directory and the port number as **8888**. Once the Jupyter server has started successfully, click the BEGIN button below. This will open a specified file in Jupyter Notebook.

BEGIN NOW!

**Note:** When you have completed all the codes in the notebook, switch to this workflow page and proceed to the next step.

Figure 7: Workflow page

## 5.3 Dashboard



Scale: regioni

Disaster: flood

Type: surface area

Lombardia

**Surface area of areas with medium flood risk (km2):** 1877.281
**Surface area of low flood risk areas (km2):** 2428.136
**Surface area of areas with high flood risk (%):** 4843.4
**Surface area of areas with medium flood risk (%):** 7.867
**Surface area of areas with low flood risk (%):** 10.175

Figure 8: Dashboard

## 6. User Case Application

In this section, we will show the possible functions ans use cases of the website, and consider the exceptions that may occur under different condition.

| Use case 1:Specified Condition Data Query | |
|---|---|
| Name | Specified Condition Data Query |
| User | Register User |
| Condition | The users who have successfully logged in and connected to the Internet |
| Flow of Events | The events follow this order:<br>User inputs or selects query conditions on the interface;<br>After confirming the query conditions, the user clicks the "Query" button;<br>Display eligible data;<br>The user reviews the query results and may perform further filtering or export data operations. |
| Exit condition | Users can exit the query function at any time by closing the web page or navigating to other pages |
| Exceptions | The user enters query conditions in an incorrect format<br>There are system errors or data source connection issues<br>The query results are empty |

Table 3: Specified Condition Data Query

| Use case 2: Data Geographic Visualization | |
|---|---|
| Name | Data Geographic Visualization |
| User | Register User |
| Condition | The users who have successfully logged in and connected to the Internet;<br>The users have completed the criteria query |
| Flow of Events | The events follow this order:<br>Users trigger the visualization process by clicking the "Visualize" button;<br>The system generates a geographic visualization |
| Exit condition | They can exit by clicking the "Exit Visualization" button;<br>Users can exit the function at any time by closing the web page or navigating to other pages |
| Exceptions | No data matches the selected criteria;<br>In case of data retrieval errors or map rendering issues |

Table 4: Data Geographic Visualization

| Use case 3: Base Map Switch | |
|---|---|
| Name | Base Map Switch |
| User | Register User |
| Condition | The users who have successfully logged in and connected to the Internet;<br>The users have completed the criteria query or data visualization |
| Flow of Events | The events follow this order:<br>Users locate the base map switch functionality on the map interface;<br>Users click the “base map switch” button, triggering the action to switch the base map;<br>The system loads the appropriate map data based on the user's selected base map type, replacing the current map display |
| Exit condition | Users can exit the function at any time by closing the web page or navigating to other pages<br>They can exit by clicking the "Exit " button |
| Exceptions | Users select an invalid base map type;<br>Users encounter issues while switching base maps |

Table 5: Base Map Switch

| Use case 4: Register | |
|---|---|
| Name | Register |
| User | All no-registered user |
| Condition | The user can register access the website by clicking the “ Register” button. |
| Flow of Events | The events follow this order:<br>A user open the website;<br>Clicks on the “ Register ” button;<br>User puts the “ username ” and “ password”;<br>After completing the above steps, users successfully register as a new user of the website and relocates to the login screen. |
| Exit condition | The username is already in use |
| Exceptions | The user already exists in the database |

Table 6: Register

| Use case 5: Log In | |
|---|---|
| Name | Log In |
| User | Register User |
| Condition | The users have registered an account. |
| Flow of Events | The events follow this order:<br>User accesses the web page and navigates to the Log In section;<br>User enters their username and password into the designated fields;<br>The user triggers the log in process by clicking the "Log In" button;<br>Upon successful log in, the user is redirected to a dashboard, account page. |
| Exit condition | The users can log out by clicking the "Log Out" button;<br>Users can exit the Log In feature by navigating away from the logged-in area of the website |
| Exceptions | The user enters incorrect username/ password<br>The user forgets their password |

Table 7: Log In