Università degli studi di Messina

DIPARTIMENTO DI SCIENZE MATEMATICHE E INFORMATICHE,
SCIENZE FISICHE E SCIENZE DELLA TERRA

# Impact Matrices And Risk Measures

**Supervisor:**
**Prof. Salvatore Distefano**

**Candidate:**
**Gunaratna Yoshan**

# Contents

# List of Figures

# Chapter 1

# Introduction

The PIGRECO project have biggest focuse on multi risk assessment through data management and dynamic map based interfaces. Over the first three months i were mainlythe focus has been on building a robust database for the data handaling and creating interactive map interfaces that allow stakeholders to visualize and analyze complex risk data.

During the data management and map interface development, the work was on designing and implementing a powerful spatial database, utilizing PostgreSQL and PostGIS, which store and manage geographic and thematic data. This database serves as the foundation for the project's enabling stroing th complex spatial data and run the queries and ensuring data integrity.

At the same time, I worked on the development of interactive map interfaces using GeoServer and Leaflet. These tools make it possible to turn raw data into web-based, interactive maps. Through these maps, users can explore different hazard layers, view real-time spatial data, and better understand risks and potential mitigation strategies.

# Chapter 2

# Database Design and Implementation

At the beginning of the project, the first step was to study and choose the right tools for managing spatial and secure data. After evaluating different options, we selected PostgreSQL, which provides the PostGIS extension, which allows the database to store and work with geographic data such as shapes, coordinates, and areas.

Once the tools were selected, I carefully studied the matrices and attributes provided by the stakeholders and described in Deliverable 7.2.2. These matrices explained what types of indicators were needed, how they relate to different hazards, and what kind of data should be collected to measure exposure, vulnerability, and risk. After this I started with the designing and implementing of the database.

## 2.1 Main Components and How They Work

- **mappa:** Stores geographic map layers, defining the reference background for all spatial data.
- **item** represents elements exposed to risk, for example, a hospital, school, or road

segment.

- **spot-proxy:** Links locations to proxy indicators, like population density or elderly percentage.
- **sorgenti, hazard, vulnerabilita, esposizione, proxy:** The three key pillars of risk modelling are what hazards exist, what elements are exposed, and how vulnerable they are.
- **modello, rischio, alternativa, effetti, indicatore:** These tables are used to build risk models, calculate potential impacts, and compare different mitigation strategies.
- **attributo, attributo-solo, attributo-composto, subattributo:** Manage the indicators used in the matrices. For example, the number of people affected in an area is a simple attribute, while a vulnerability index might be a composite one made up of many sub-attributes.

All relationships between these tables are defined using foreign keys, which ensure that the data is linked correctly. This structure also allows the database to support matrix-based evaluations across dimensions like the built environment, public services, financial system, and more.

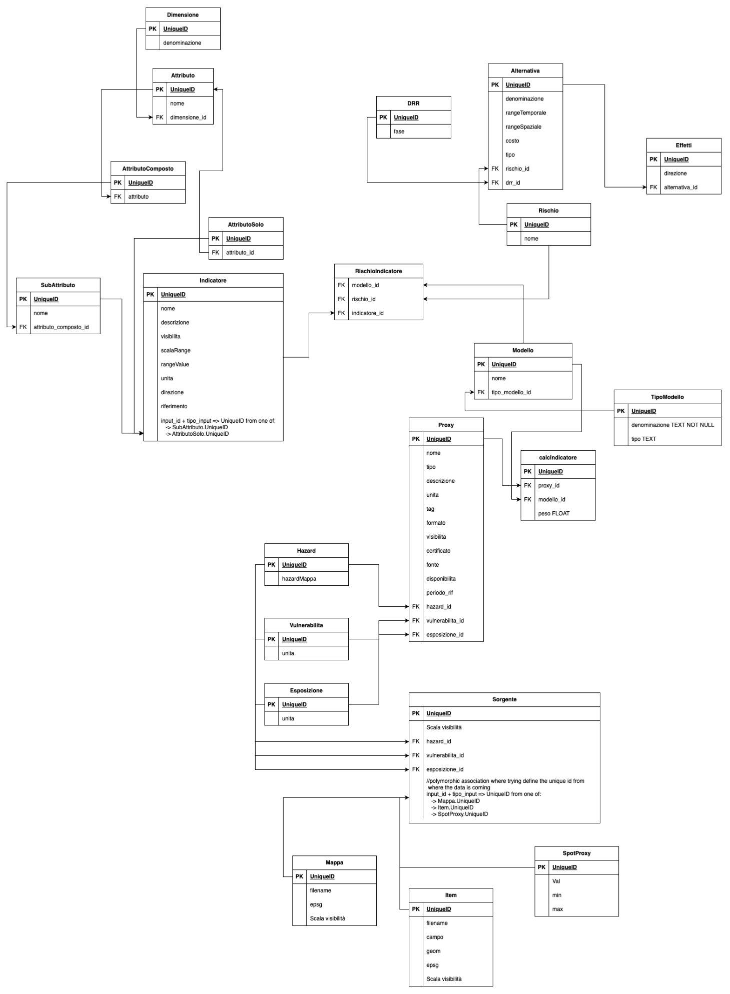


Figure 2.1: The architecture of the DB

## 2.2 Alignment with Deliverable 7.2.2

The database design is fully aligned with the structure and goals outlined in Deliverable 7.2.2 in the following ways

### 2.2.1 Support for Multi-Risk Assessment Across Sectors

The deliverable tells the need to assess risk across several social dimensions like: Built Environment, Public Services, Individual Well-being, Communities, Environmental Systems, Financial System, and Business Activities. And the tables like Dimensione, Attributo, Indicatore, and SubAttributo reflect these sectors and support linking each indicator to a specific matrix dimension.

### 2.2.2 Indicator to evaluation

Deliverable 7.2.2 mentions the need for the indicators to evaluate impacts and measure the effectiveness of mitigation, which mean that the use of Indicatore, AttributoSolo, and AttributoComposto, allow reach our requirement.

### 2.2.3 Integration of Hazard, Exposure, and Vulnerability

A key component of the deliverable is the integration of hazard types, what is exposed, and how vulnerable that exposure is. Using the Hazard, Esposizione, and Vulnerabilita tables, which are then combined through Sorgente and Proxy to connect to the impact analysis.

### 2.2.4 Measure the effectiveness and Modelling Scenarios

As well as the deliverables, evaluating the effectiveness of alternatives and risk reduction measures.The table like Alternativa, Effetti, Modello, and Rischio support building different scenarios and comparing outcomes, which fits the deliverable's logic for assessing strategies.

### 2.2.5 Scalability and Spatial Reference

The deliverable requires spatially resolved, scalable analysis. With input data like mapa, item and PostGIS spatial capabilities to ensure that each data point is linked to a geographic area, supporting analysis at multiple spatial levels (e.g., municipality, province, national).

# Chapter 3

# System Architecture and Tools Used

The next step was to connect the tools that allow us to view and explore the data visually and in real time. This was done using a combination of three tools: Docker,PgAdmin, GeoServer, and Leaflet.

## 3.1 Docker and Microservices Architecture

### 3.1.1 What is Docker?

Docker is a platform that allows to package up applications and all their dependencies into a single unit called a container. This makes it possible to run the application in a reliable and repeatable way, regardless of the machine or operating system. It solves the common problem and it works on computers by ensuring that everything, like code, runtime, and libraries, aligns together.

### 3.1.2 Container and an Image

A Docker image is like a blueprint or recipe, it contains everything needed to create a container, including the operating system, installed programs, configurations, and scripts. A Docker container is a running instance of an image. And the containers are isolated from each other, lightweight, and very fast to start or stop.

### 3.1.3 How Microservices Work in This Context

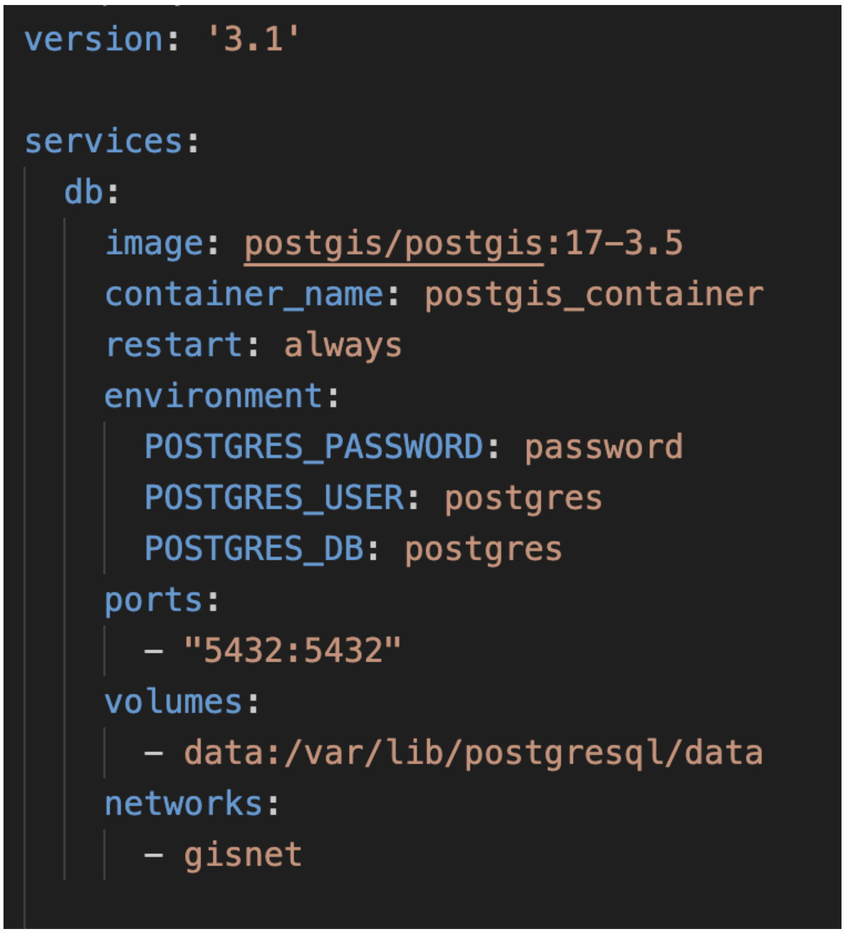
How each service is working in the system (PostgreSQL, pgAdmin, and GeoServer) is run inside its own container. This architecture is called microservices based, because each tool runs independently, but they are connected and communicate with each other via a Docker internal network. This makes the system modular, scalable, and easier to maintain.

## 3.2 Three main containers

### 3.2.1 PostgreSQL with PostGIS container

The DB service provides the PostgreSQL container with the PostGIS extension, which provides spatial capabilities for geospatial data processing. The image used is postgis/postgis:17-3.5, which bundles PostgreSQL and PostGIS in a single ready to use package. Environment variables are set to define the database name (POSTGRES DB=postgres), the default user (postgres) and the password (password).

The container is named postgis container, and the service is configured to restart automatically in case of failure (restart: always). It listens on the default PostgreSQL port 5432, which is exposed to the host system. A Docker volume (data) is mounted to /var/lib/postgresql/data to persist database information across restarts. This service is connected to a custom Docker network (gisnet) to allow secure communication with other services like GeoServer and pgAdmin.



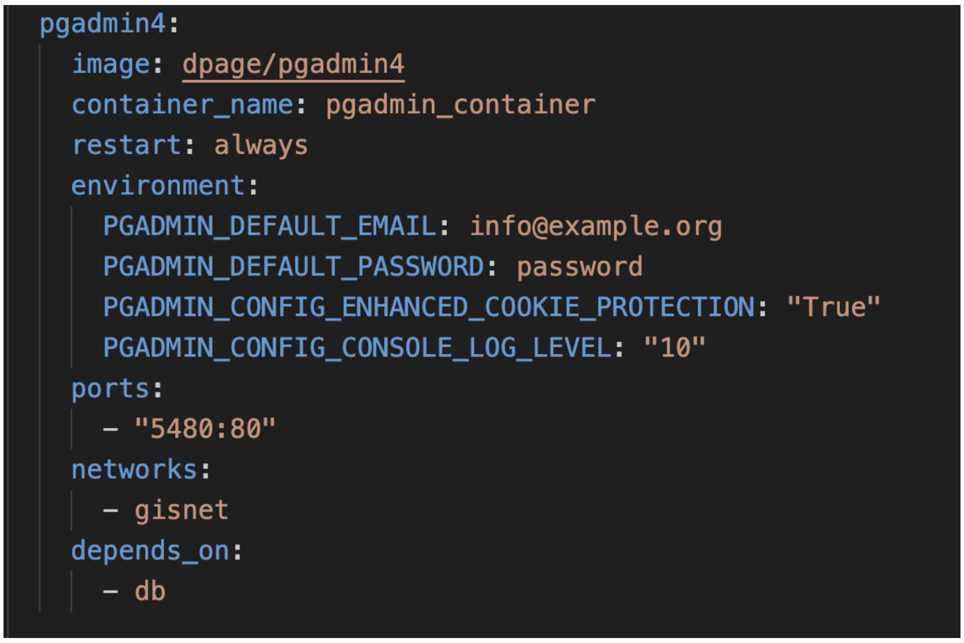
```
version: '3.1'

services:
  db:
    image: postgis/postgis:17-3.5
    container_name: postgis_container
    restart: always
    environment:
      POSTGRES_PASSWORD: password
      POSTGRES_USER: postgres
      POSTGRES_DB: postgres
    ports:
      - "5432:5432"
    volumes:
      - data:/var/lib/postgresql/data
    networks:
      - gisnet
```

Figure 3.1: Enter Caption

### 3.2.2 pgAdmin4

The pgadmin4 service provides a web-based interface for managing the PostgreSQL database. This is especially useful for visually exploring tables, running SQL queries, and monitoring database activity. The container is built using the dpage/pgadmin4 image and named pgadmin container. It is configured with admin credentials (PGADMIN DEFAULT EMAIL and PGADMIN DEFAULT PASSWORD) and logs at a detailed level. The service exposes port 5480 on the host, allowing the user to access pgAdmin from a browser via http://localhost:5480.



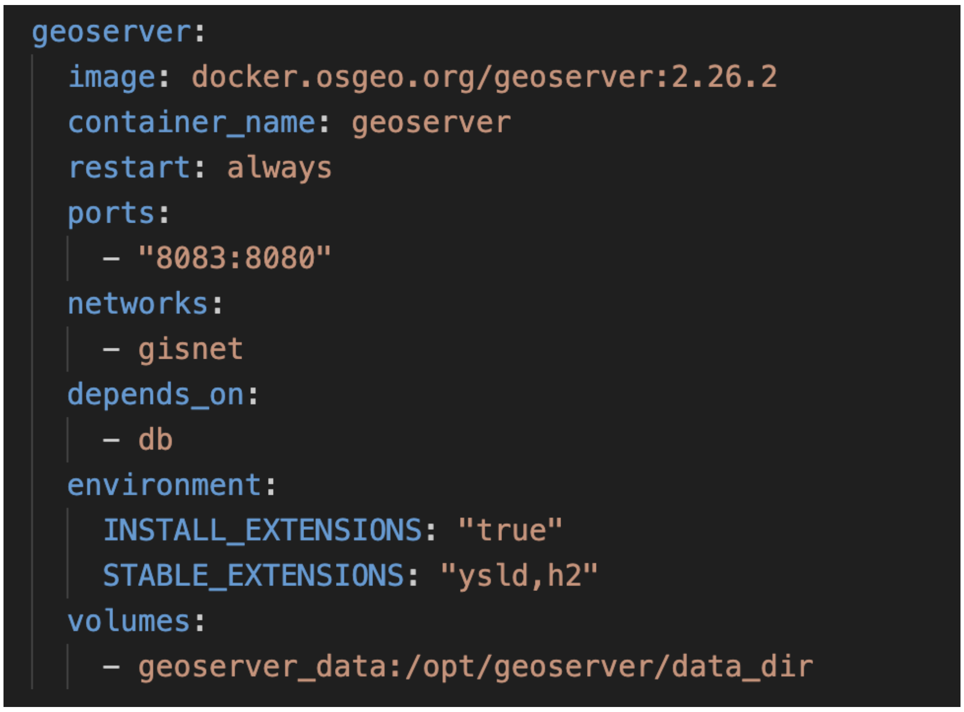
```
pgadmin4:
  image: dpage/pgadmin4
  container_name: pgadmin_container
  restart: always
  environment:
    PGADMIN_DEFAULT_EMAIL: info@example.org
    PGADMIN_DEFAULT_PASSWORD: password
    PGADMIN_CONFIG_ENHANCED_COOKIE_PROTECTION: "True"
    PGADMIN_CONFIG_CONSOLE_LOG_LEVEL: "10"
  ports:
    - "5480:80"
  networks:
    - gisnet
  depends_on:
    - db
```

Figure 3.2: pgadmin4 service

This service depends on the db service, meaning it starts only after the database is available. Like the database, it is also connected to the gisnet network to ensure secure communication.

### 3.2.3 GeoServer

The geoserver service is publishing geospatial data through web services such as WMS (Web Map Service) and WFS (Web Feature Service). It uses the image docker.osgeo.org/geoserver:2.26.2 which comes with the essential tools to run GeoServer. The container is named geoserver. GeoServer runs on port 8080 inside the container, which is mapped to port 8083 on the host, making it accessible at http://localhost:8083. It includes environment variables to install



```yaml
geoserver:
  image: docker.osgeo.org/geoserver:2.26.2
  container_name: geoserver
  restart: always
  ports:
    - "8083:8080"
  networks:
    - gisnet
  depends_on:
    - db
  environment:
    INSTALL_EXTENSIONS: "true"
    STABLE_EXTENSIONS: "ysld,h2"
  volumes:
    - geoserver_data:/opt/geoserver/data_dir
```

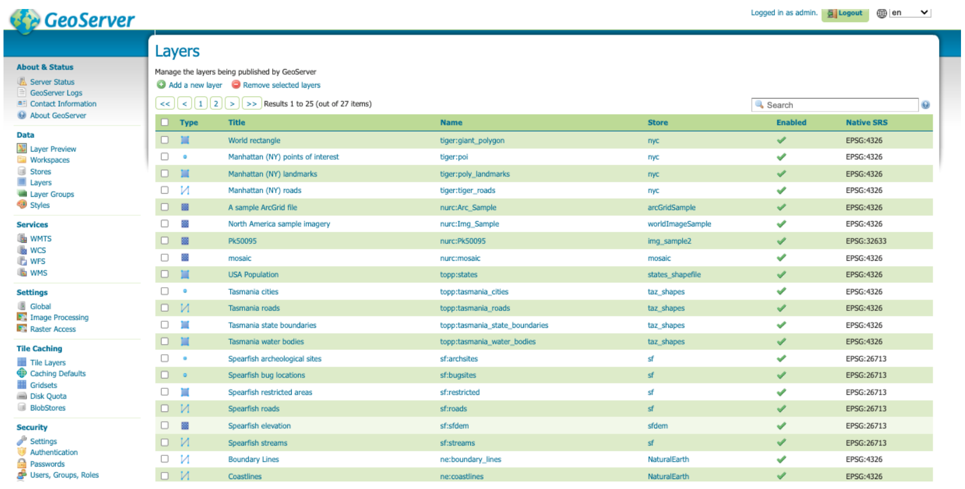
Figure 3.3: Enter Caption

GeoServer is designed to handle, publish, and share geospatial data. And it easily can be integrated with PostgreSQL, especially when the PostGIS extension is enabled. The integration allows GeoServer to connect directly to the database and serve by providing the content as web accessible layers.

The connection between the database is achieved using the standard credentials like host, port, database name, username, and password. Once connected, GeoServer scans the database and detects spatial tables. These tables can then be published as layers in GeoServer, making them accessible over the web through mapping services.

### Layers

In GeoServer, the important concept is the layer. A layer is represent a spatial dataset from the database. When we configure a new layer in GeoServer, and it chooses a spatial table from the PostGIS database. GeoServer then uses that table's geometry column to render the geographic features.



Layers

Manage the layers being published by GeoServer

Add a new layer  Remove selected layers

<< < 1 2 > >> Results 1 to 25 (out of 27 items)

| Type | Title | Name | Store | Enabled | Native SRS |
|---|---|---|---|---|---|
| ▦ | World rectangle | tiger:giant_polygon | nyc | ✓ | EPSG:4326 |
| ● | Manhattan (NY) points of interest | tiger:poi | nyc | ✓ | EPSG:4326 |
| ▦ | Manhattan (NY) landmarks | tiger:poly_landmarks | nyc | ✓ | EPSG:4326 |
| ⟋ | Manhattan (NY) roads | tiger:tiger_roads | nyc | ✓ | EPSG:4326 |
| ▩ | A sample ArcGrid file | nurc:Arc_Sample | arcGridSample | ✓ | EPSG:4326 |
| ▩ | North America sample imagery | nurc:Img_Sample | worldImageSample | ✓ | EPSG:4326 |
| ▩ | Pk50095 | nurc:Pk50095 | img_sample2 | ✓ | EPSG:32633 |
| ▩ | mosaic | nurc:mosaic | mosaic | ✓ | EPSG:4326 |
| ▦ | USA Population | topp:states | states_shapefile | ✓ | EPSG:4326 |
| ● | Tasmania cities | topp:tasmania_cities | taz_shapes | ✓ | EPSG:4326 |
| ⟋ | Tasmania roads | topp:tasmania_roads | taz_shapes | ✓ | EPSG:4326 |
| ▦ | Tasmania state boundaries | topp:tasmania_state_boundaries | taz_shapes | ✓ | EPSG:4326 |
| ▦ | Tasmania water bodies | topp:tasmania_water_bodies | taz_shapes | ✓ | EPSG:4326 |
| ● | Spearfish archeological sites | sf:archsites | sf | ✓ | EPSG:26713 |
| ● | Spearfish bug locations | sf:bugsites | sf | ✓ | EPSG:26713 |
| ▦ | Spearfish restricted areas | sf:restricted | sf | ✓ | EPSG:26713 |
| ⟋ | Spearfish roads | sf:roads | sf | ✓ | EPSG:26713 |
| ▩ | Spearfish elevation | sf:sfdem | sfdem | ✓ | EPSG:26713 |
| ⟋ | Spearfish streams | sf:streams | sf | ✓ | EPSG:26713 |
| ⟋ | Boundary Lines | ne:boundary_lines | NaturalEarth | ✓ | EPSG:4326 |
| ⟋ | Coastlines | ne:coastlines | NaturalEarth | ✓ | EPSG:4326 |

Figure 3.4: Geoserver Dashboard

Each layer in GeoServer includes its own settings for coordinate reference system (CRS), and attribute filters. This means we can control how the data appears on the map and how it's filtered or displayed, even when it's all coming from the same database. For example, I used a single dataset containing earthquakes from all over the world and then created separate layers for each country.

This approach allows users to focus on a specific region and understand what types of hazards are possible in that location. It supports better risk awareness and can guide prevention or preparedness actions more effectively.

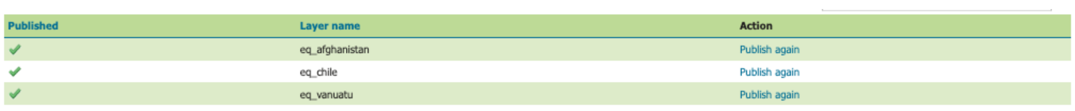
**Layers work as follows:**

1. First we have to connect GeoServer to the PostGIS database by creating a Data Store. Then the GeoServer look for spatial data within the database.

2. Select the specific table or view in the database that contains the data to publish. In my case, I used views that I created with a query and ran it in the query tool of the pgAdmin.

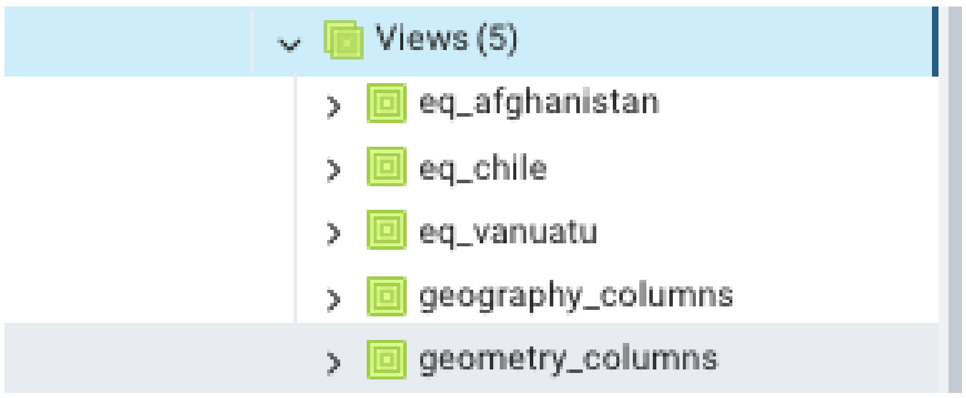
3. GeoServer then publishes the data as a Layer, which becomes available via WMS (Web Map Service) or WFS (Web Feature Service). These services can be accessed by tools like Leaflet or any other web mapping client.

This method provide flexibility and modularity to the system each layer can be updated, styled, or analysed independently. It also makes it easy to compare risks between different areas or within the same area.



| Published | Layer name | Action |
|---|---|---|
| ✓ | eq_afghanistan | Publish again |
| ✓ | eq_chile | Publish again |
| ✓ | eq_vanuatu | Publish again |

Figure 3.5: Available layers in the database



Views (5)
- eq_afghanistan
- eq_chile
- eq_vanuatu
- geography_columns
- geometry_columns

Figure 3.6: View list

### 3.2.4 Web Map Service

WMS is the most commonly used service in GeoServer. It allows the server to render map images to the client, based on the data stored in the database. These images are generated dynamically and can be displayed on web maps like Leaflet. When a client makes a WMS request, it includes parameters like the desired layer, geographic bounding box, image size, and format. GeoServer receives this, fetches the relevant data from PostGIS, styles it according to predefined rules, and returns an image of the map. This is very useful for visualisations because it's fast, simple, and doesn't require the client to handle raw geometry data.

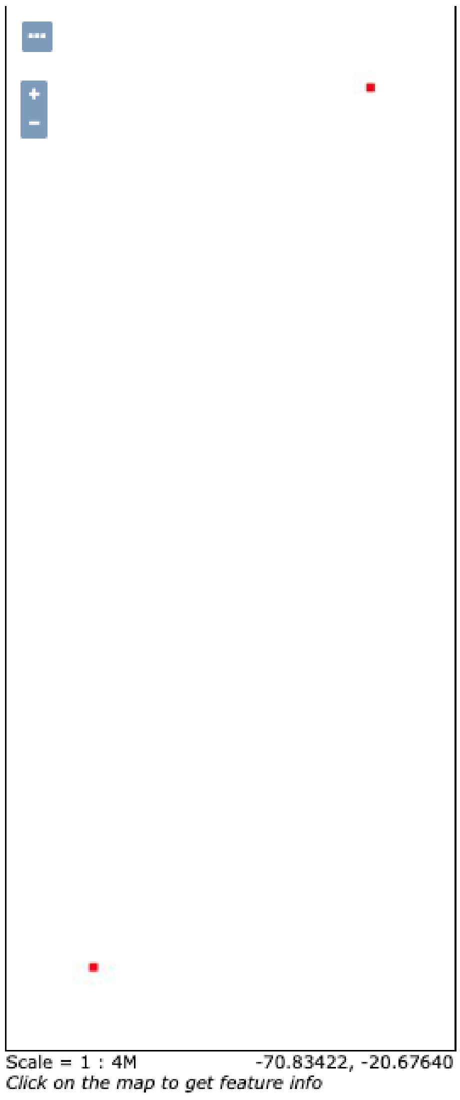
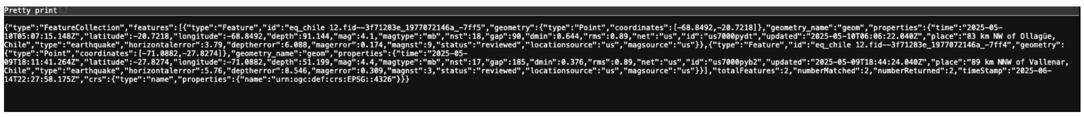


Figure 3.7: WMS points(just 2 points available on this layer)

### 3.2.5 Web Feature Service

WFS is more advanced and allows direct access to the features themselves, not just images. With WFS, clients can request the actual data behind the map, like points, lines, polygons, and their attributes



```
{"type":"FeatureCollection","features":[{"type":"Feature","id":"eq_chile 12.fid--3f71283e_1977072146a_-7ff5","geometry":{"type":"Point","coordinates":[-68.8492,-20.7218]},"geometry_name":"geom","properties":{"time":"2025-05-
10T05:07:15.148Z","latitude":-20.7218,"longitude":-68.8492,"depth":91.144,"mag":4.1,"magtype":"mb","nst":18,"gap":90,"dmin":0.644,"rms":0.89,"net":"us","id":"us7000pydt","updated":"2025-05-10T06:06:22.040Z","place":"83 km NW of Ollagüe,
Chile","type":"earthquake","horizontalerror":3.79,"deptherror":6.088,"magerror":0.174,"magnst":9,"status":"reviewed","locationsource":"us","magsource":"us"}},{"type":"Feature","id":"eq_chile 12.fid--3f71283e_1977072146a_-7ff4","geometry":
{"type":"Point","coordinates":[-71.0882,-27.8274]},"geometry_name":"geom","properties":{"time":"2025-05-
09T18:11:41.264Z","latitude":-27.8274,"longitude":-71.0882,"depth":51.199,"mag":4.4,"magtype":"mb","nst":17,"gap":185,"dmin":0.376,"rms":0.89,"net":"us","id":"us7000pyb2","updated":"2025-05-09T18:44:24.040Z","place":"89 km NNW of Vallenar,
Chile","type":"earthquake","horizontalerror":5.76,"deptherror":8.546,"magerror":0.309,"magnst":3,"status":"reviewed","locationsource":"us","magsource":"us"}}],"totalFeatures":2,"numberMatched":2,"numberReturned":2,"timeStamp":"2025-06-
14T22:27:58.175Z","crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:EPSG::4326"}}}
```

Figure 3.8: WFS service

## 3.3 Web Map

### 3.3.1 Leaflet and Its Role in the system

Leaflet is a JavaScript library used to build interactive maps that run directly in the web browser. It plays a key role in creating WebGIS applications, which are Geographic Information Systems made accessible and usable through a web interface.

WebGIS using Leaflet allows users to visualize, explore, and interact with spatial data without installing any desktop GIS software. This spatial information more accessible and useful, especially for decision makers and the general public.

Figure 3.9: Leaflet map

### 3.3.2 Integrating Leaflet with GeoServer

The leaflet doesn't connect directly to a database. Instead, it connects to GeoServer using standard web services like Web Map Service and Web Feature Service. Leaflet has built in support for WMS, so it's easy to display map layers published in GeoServer by simply pointing Leaflet to the GeoServer WMS URL and choosing the right layer name.

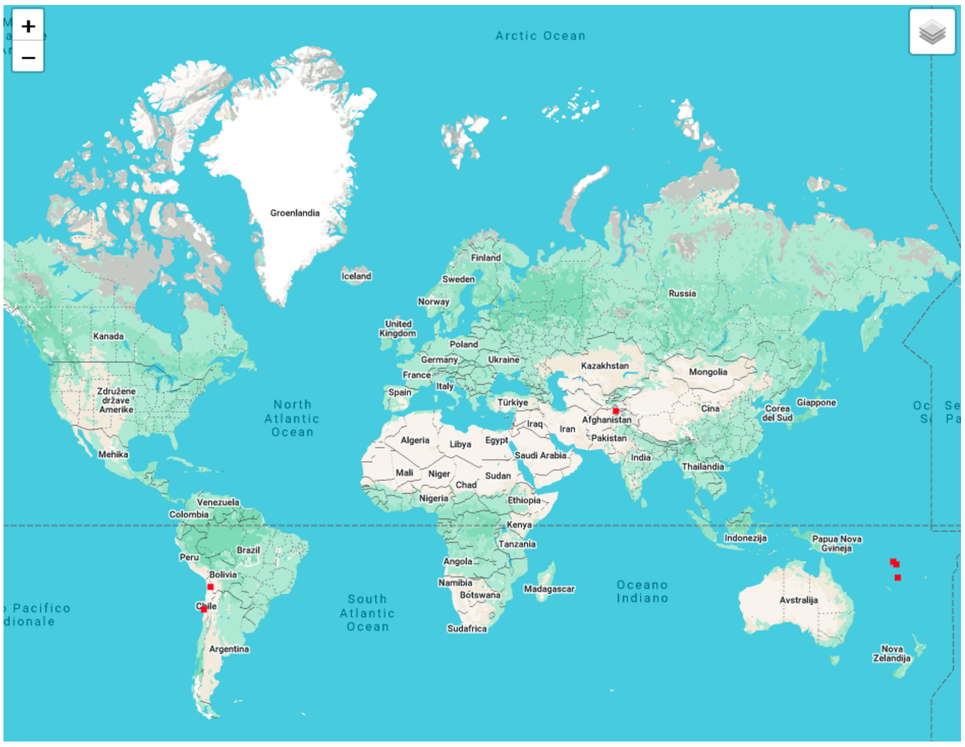


Figure 3.10: Map with WMS

For WFS, which returns vector data, the connection is a bit more complex. Leaflet doesn't support WFS natively, but there are ways to make it work. For example, we had to adjust some GeoServer settings to allow WFS responses in GeoJSON format, which Leaflet can easily handle. Then, in the Leaflet script, we made a custom function to fetch the WFS data and display it on the map. This part gave us some technical issues, especially with CORS and incorrect for- mats, but after correcting the request parameters and enabling the GeoJSON output in GeoServer, we managed to load WFS features in Leaflet.

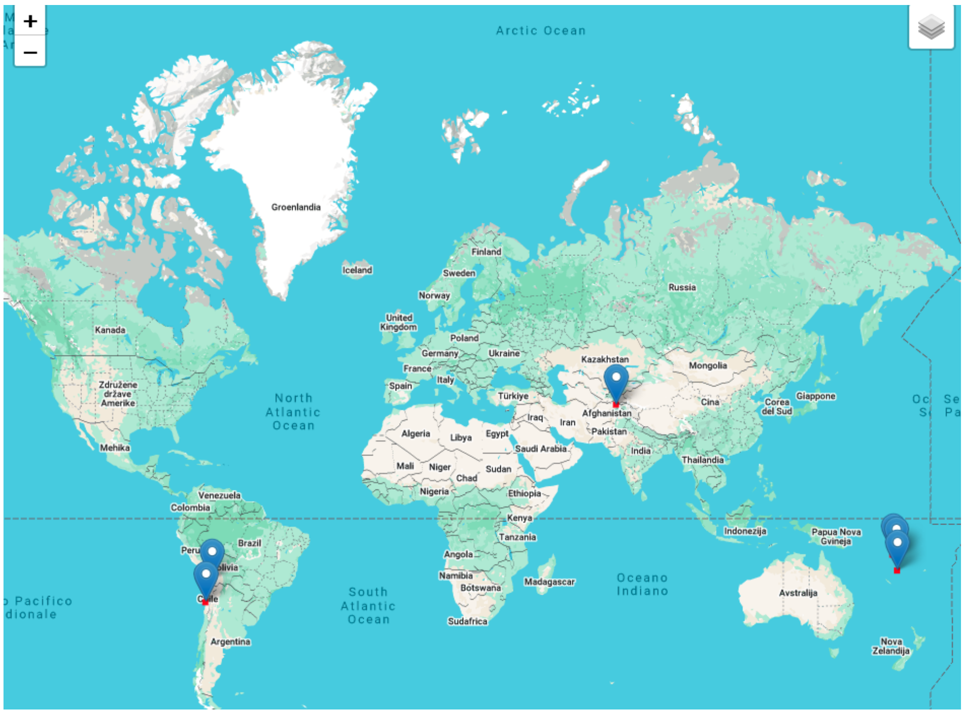
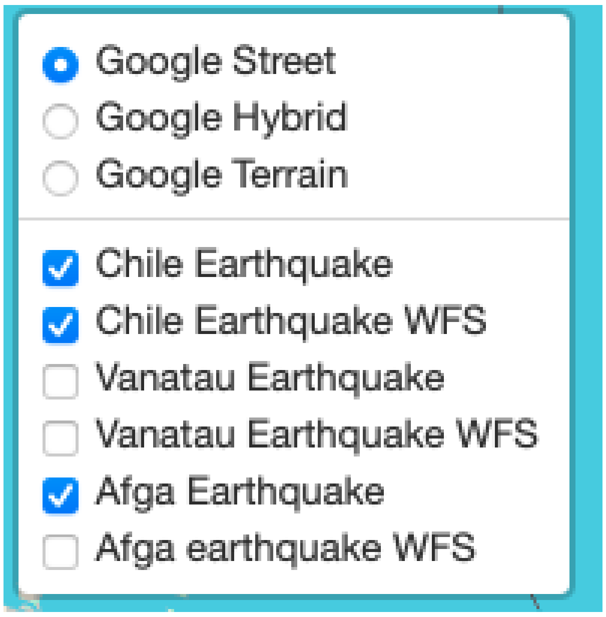


Figure 3.11: Map with WFS

### 3.3.3 Map Layers and Layer Control

Leaflet uses the concept of layers to organize different types of map content. For example, a WMS base map can be one layers, and a WFS vector layers can be another. These layers can be added, removed, or styled dynamically.



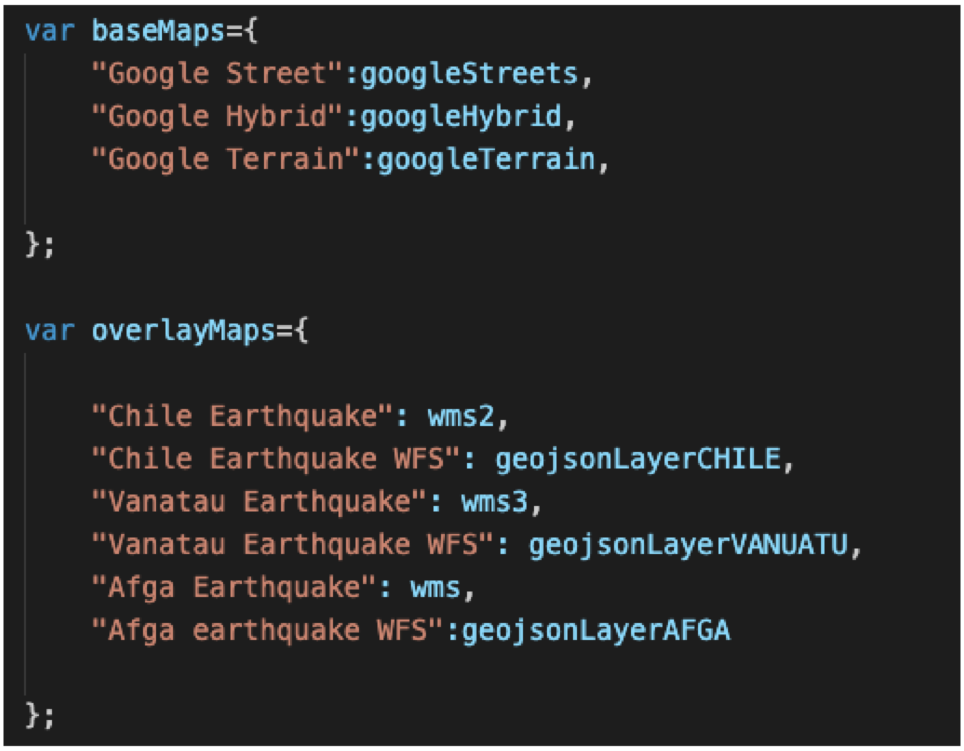
● Google Street
○ Google Hybrid
○ Google Terrain

[✓] Chile Earthquake
[✓] Chile Earthquake WFS
[ ] Vanatau Earthquake
[ ] Vanatau Earthquake WFS
[✓] Afga Earthquake
[ ] Afga earthquake WFS

Figure 3.12: Layers selection

We also used the Layer Control feature in Leaflet to allow users to switch between different base maps (like satellite view or street map) and to toggle overlays. This makes the map flexible and interactive.

Each layer is defined in the JavaScript code and linked to a specific URL from GeoServer. The Layer Control is automatically updated to reflect these choices, making it easier for users to explore different types of data.



```
var baseMaps={
    "Google Street":googleStreets,
    "Google Hybrid":googleHybrid,
    "Google Terrain":googleTerrain,

};

var overlayMaps={

    "Chile Earthquake": wms2,
    "Chile Earthquake WFS": geojsonLayerCHILE,
    "Vanatau Earthquake": wms3,
    "Vanatau Earthquake WFS": geojsonLayerVANUATU,
    "Afga Earthquake": wms,
    "Afga earthquake WFS":geojsonLayerAFGA

};
```

Figure 3.13: Layer control code

### 3.3.4 Alignment with Deliverable 7.2.2

#### Interoperable and Modular Spatial Infrastructure

Deliverable 7.2.2 state the need for a spatially enabled system that supports modularity and their connection with each other. The architecture, built with Docker and Docker Compose, allows each component (database, GeoServer, frontend) to operate as an independent service, while being fully integrated through a shared internal network (gisnet). This provide flexibility, easy updates, and clear separation.

### Dynamic Visualization and Interaction

This part highlights that importance of user friendly tools that can visualize data across sectors and hazard types. So I use GeoServer to publish WMS/WFS layers, and Leaflet to display them interactively on the web, directly addresses this. Users can view, toggle, and interact with multiple hazard and opens multiple layers in real time, satisfying the need for a dynamic and accessible interface.

### Dynamic Visualization and Interaction

This part highlights that importance of user friendly tools that can visualize data across sectors and hazard types. So I use GeoServer to publish WMS/WFS layers, and Leaflet to display them interactively on the web, directly addresses this. Users can view, toggle, and interact with multiple hazard and opens multiple layers in real time, satisfying the need for a dynamic and accessible interface.

### Support for Multi Hazard, Multi Domain Representation

Also deliverable requires a system that can display and assess various hazards across multiple thematic domains (e.g., infrastructure, communities, environment). By publishing multiple layers in GeoServer each of them linked to in the database, this system supports layered analysis of overlapping risks, as prescribed in the deliverable.

### Scalability and Thematic Expansion

Deliverable provide the need for systems that can grow as new hazards, territories, or indicators are added. This design supports adding new WMS/WFS layers at any time without changing the core system. Each layer is combined to form a view of dataset within PostgreSQL

# Chapter 4

# Data Population

The next step was to populate the data with multiple connected tables. This process ensuring each value were uniquely and correctly identified using UUIDs (Universally Unique Identifiers), while maintaining proper relationships through foreign keys to ensure data consistency and integrity.

## 4.1 Populating the dimensione Table

We began by inserting values into the dimensione table, which defines the seven societal sectors used for multi-risk impact assessment:

### 4.1.1 Built Environment

This dimension includes all physical and man made structures such as residential buildings, commercial complexes, public infrastructure, transport networks, and industrial zones. The indicators under this dimension help evaluate the structural vulnerability and potential economic losses to critical assets.

### 4.1.2 Business Activities

This category focuses on the economic sectors of a region, including primary, secondary, and tertiary industries. It helps capture the disruption of productive capacity due to natural hazards and assesses how disasters may affect employment, market operations, and overall economic output.

### 4.1.3 Communities

The Communities dimension is focusing on the social and cultural aspects of a population. It includes factors like cultural heritage, education level, and social cohesion. This dimension is important to understanding how hazards impact human capital, cultural identity, and the resilience of local communities, especially marginalized or high risk groups.

### 4.1.4 Environmental Systems

This dimension includes natural ecosystems and biophysical components such as the atmosphere, lithosphere (soil and geological structures), biosphere (plants and animals), and hydrosphere (surface and groundwater systems). Environmental Systems are studied to understand how nature is harmed by things like pollution, loss of animals and plants, or damage caused by natural events or human activities

### 4.1.5 Financial System

This dimension involves on the institutional and market elements of the economy, such as banks, insurance companies, and financial stability indicators. It is used to measure the vulnerabilities, the capacity to recover post disaster, and the financial ripple effects of risk events on investments, lending, and economic planning.

### 4.1.6 Individual Well-being

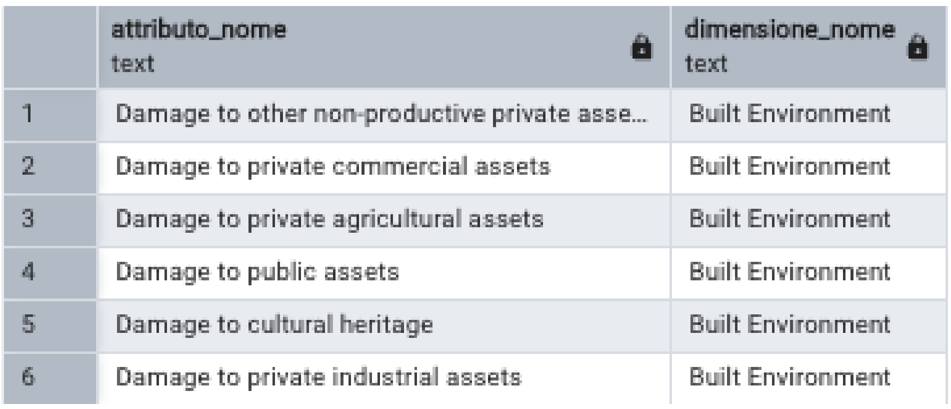
This dimension focuses on direct and indirect effects of hazards on individuals, including injuries, deaths, displacement, and the loss of livelihoods. It covers both the physical and economic situation of people, enabling targeted responses that prioritize health, housing, and income recovery for affected populations.

### 4.1.7 Public Services

Public Services include utilities and emergency response systems such as water supply, electricity, waste management, healthcare, education, and civil protection. This helps us understand how much important services are affected and how well institutions can keep working during and after disasters

## 4.2 Dimensions with Their Attributes

To support multi risk evaluation of society, the dimensione table includes seven thematic areas. Each of these areas is enriched by a set of attributo values that describe specific impacts relevant to risk modeling. These indicators were derived from the Excel matrices and are now structured in the database for analysis and visualization.



| | attributo_nome text | dimensione_nome text |
|---|---|---|
| 1 | Damage to other non-productive private asse... | Built Environment |
| 2 | Damage to private commercial assets | Built Environment |
| 3 | Damage to private agricultural assets | Built Environment |
| 4 | Damage to public assets | Built Environment |
| 5 | Damage to cultural heritage | Built Environment |
| 6 | Damage to private industrial assets | Built Environment |

Figure 4.1: Relationship between the Built Environment dimension and it’s attributes

## 4.3 UUID and its extension

To make sure each data in the database has a unique and reliable identifier, we used something called UUID, which stands for Universally Unique Identifier.Instade of the auto incremental like SERIAL, which increase one by one (like 1, 2, 3. . . ), a UUID is a long combination of letters and numbers that is almost impossible to repeat. This is very useful when many people are working on the same system or when data might come from different sources, because it avoids the risk of having duplicate IDs.

In PostgreSQL, we used a special extension called uuid-ossp that lets the database automatically create these UUIDs. Once we activated the extension using the command CREATE EXTENSION IF NOT EXISTS "uuid-ossp"; we could use functions like uuid-generate-v4() to generate a random UUID for each new entry. This way, all row in the tables like dimensione, attributo, and and other tables gets a unique ID that makes the data easy to manage and connect without confusion.

# Chapter 5

# Conclusion and Future Works

In these past three months, the work on the PIGRECO project has led to the creation of a robust system for multi risk assessment. We began by designing and implementing a relational spatial database using PostgreSQL and PostGIS, building the architecture and then the tables of the database needed to handle hazard data, exposure, vulnerability, proxies, indicators, and measures.

This structure was carefully designed with clearly defined relationships between elements. This database is connected to the external tools using Docker, GeoServer, and Leaflet, allowing spatial data to be published and visualized on interactive web maps. This provide the possibility for stakeholders to explore multiple hazard layers, view real time risk data, and interact with features at various geographic levels.

In the third phase focus was on the population of key tables, including dimensione, attributo, and attributocomposto. Each entry was associated with a unique UUID using the uuid-ossp extension to ensure reliable identification and future scalability. The attribute data was inserted based on thematic matrices provided in the deliverable, covering all seven societal dimensions (e.g., Built Environment, Financial System, and Public Services). Proper foreign key constraints were applied to preserve data integrity, and composite attributes were added to support advanced modeling.

Looking ahead, the next steps will involve continuing the population of other core tables such as esposizione, vulnerabilita, and hazard, using real regional datasets. We will also begin testing risk model outputs, enhancing the map interface with filtering and querying features, and aligning the system with evolving stakeholder feedback. As the project progresses, new spatial layers and modeling methods will be gradually integrated to support more detailed and dynamic risk evaluations.