Design Document

GeoAir

POLITECNICO
MILANO 1863

Gianluca Bettoni
Mobina Faraji
Alessia Ippolito
Edoardo Pessina

INDEX

1. Introduction **2**
1.1 Context and Motivations 2
1.2 Purpose 2
1.3 Scope and Limitations 2
1.4 Definitions, Acronyms, and Abbreviations 3
2. Architectural design **4**
2.1 Overview 4
2.2 Component diagram 5
2.3 Technology stack 5
3. Software functionalities **6**
3.1 Accessibility 6
3.2 Homepage 6
3.3 Map page 6
3.4 Trend page 6
3.5 Download 7
4. Implementation **7**
4.1 API implementation 7
4.2 API Design 8
5. Database **8**
5.1 Data Source 8
5.2 PostGRE database 8
6. Deployment **9**
7. User interface design **10**
8. Bibliography **14**

document version	data	group members
v1	17/05/2025	Gianluca Bettoni Edoardo Pessina
v2	28/05/2025	Monìbina Faraji Alessia Ippolito
v3 final	04/07/2025	Gianluca Bettoni Monìbina Faraji Alessia Ippolito Edoardo Pessina

1. Introduction

1.1 Context and Motivations

This project focuses on Air quality monitoring which has become a critical public health concern, particularly in densely populated and industrial regions and municipalities like Lombardy (area of interest) where natural air recirculation is limited. Pollution from traffic, industrial activities, and heating systems contributes to high concentrations of particulate matter and harmful gases.

To address the limitation of accessible data for users, the project is set as a smart, interactive web-based platform to make air quality data accessible and understandable for everyone who has even limited knowledge about this concept by using graphs, maps and diagrams which make understanding clearer. The project enables users to explore historical pollution data, visualize spatial and temporal trends, and export insights through an intuitive dashboard.

1.2 Purpose

This Design Document outlines the architectural and technical blueprint of the **GeoAir** web-app. It translates the functional and non-functional requirements defined in the **RASD** into a structured design that guides system development. The document defines how the system will be constructed to fulfill the specified requirements by detailing:

- The overall system architecture project.
- The individual components and the interactions between them.
- The selected technologies, including Flask, PostgreSQL/PostGIS, and Jupyter Notebook, Dash, Python.
- The data structures, communication protocols, and API access patterns.

1.3 Scope and Limitations

GeoAir is a client-server application developed to support air quality analysis and visualization in the Lombardy region. The system is composed of:

- A PostgreSQL/PostGIS database that stores pollutants, sensor data and user credentials.
- A Flask backend server that exposes a REST API for querying the data.
- A Python, with Dash for frontend dashboard for data visualization and user interaction.

Core functionalities include:

- Querying pollution data - Interactive visualizations using maps, graphs.

- Data export and simple analytics by image.
- User authentication to access advanced functionalities (login functionality of the app)

Limitations:

- The system is limited to the Lombardy region and does not support data from other areas.
- Features like predictive analytics, mobile support, and multilingual UI are not part of the initial version.
- Data accuracy is dependent on external sources from Dati Lombardia.

1.4 Definitions, Acronyms, and Abbreviations

Term / Acronym	Definition
API	Application Programming Interface – a way for different parts of the system to communicate
Dati Lombardia	Open data portal from the Lombardy Region government providing air quality and sensor data
RASD	Requirements Analysis and Specification Document – outlines what the system must do
SRD	Software Release Document – final deliverable describing how to install and run the application
Station	Database that contains all the info of a specific station sensor like the unit of measurement and the position.
Measurement	Database that contains the measurement of all the station sensors based on datetime.
Users	Database with accounts data

PM10		Concentration of PM10
PM2.5		Concentration of PM 2.5
	NO_x	Concentration of nitrogen oxide
	O_3	Concentration of Ozone
Pollutant group		Groups of similar pollutant used for select
		Particulate matter (PM10, PM2.5, other)
		Nitrogen compound (NO, NO2, NOx)
		Sulfur and Carbon compound (CO, SO2)
		Heavy metals (As, Cd, Ni, Pb)
		Others (C6H6, C20H12, O3)

2. Architectural design

2.1 Overview

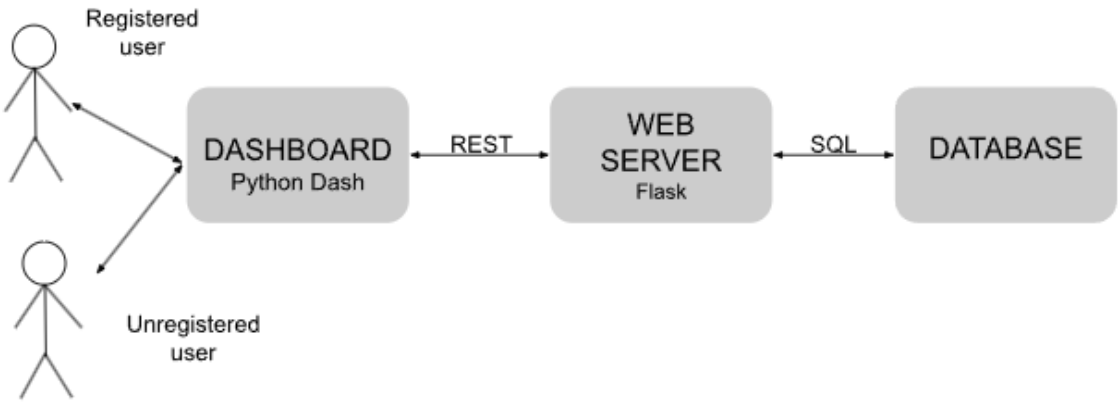
The design of the software architecture can be divided into three principal components:

The **Dashboard** serves as the frontend of the application, developed in Python using the Dash library. It provides an interactive web interface where users can visualize air quality maps, graphs related to the Lombardy region. Registered users have access to additional features, allowing them to analyze and download more advanced air quality data.

The **Web server** receives the requests of the users from the dashboard, elaborates them to make requests in SQL and retrieve the data from the database; by the API exposes the data to the dashboard. Moreover, permits the maps, graphs and data elaboration.

The **Database** stores all air quality data from Regione Lombardia, the relative sensors and user account information. The database is accessible to administrators for management and maintenance purposes.

2.2 Component diagram



2.3 Technology stack

This segment contains the instruments used in the creation of the web software, divided by component.

- Dashboard tools:
  - *Plotly.express*: for the implementation of the bar and line graphs.
  - *Dash* with *html, dcc, Output, Input, State, callback, no_update, page_container*: to manage the pages layout, interaction and the callbacks.
  - *Dash_leaflet*: for the creation of the maps and layers.
  - *Pandas*: to manage the data in the dataframes.
  - *GeoPandas*: to read the input file for a map.
  - *Psycopg2*: for the connection with the database.
  - *Requests*: to make HTTP requests.
  - *Datetime*: to manage the dates selection from the users.
  - *uuid, os, zipfile, io*: to manage the download of the shapefile.
- Web server tools:
  - *Flask*: to manage the server and the APIs.
  - *Pandas*: to manage data in the dataframes.
  - *Psycopg2*: to connect the database to the web server.
  - *Warnings*: filtering the warnings due to pandas in psycopg2.
- Database tools:
  - *Jupyter notebook*: used for the database code.

- *PostgreSQL*: An open-source relational database used to store structured air quality data, such as pollutant levels, timestamps, and sensor metadata.
- *PostGIS*: A spatial database extension for PostgreSQL that enables storage and querying of geographic information, such as coordinates of sensor stations or city boundaries.
- *Urllib.parse* with *urlencode*: to manage the parameters in the URL data request to dati Lombardia.
- *Psycopg2.extras* with *execute_values*: to insert multiple values in the PostgreSQL tables.

3. Software functionalities

3.1 Accessibility

The user can register, login and logout to access advanced operations on the web-app.

- User registration
- User login
- User logout

3.2 Homepage

Dashboard that show a general overview of the data:

- Map view of all stations filtered per pollutant to specify the group used and provinces
- Histogram with the number of sensor per province
- Histogram representing the number of sensor per pollutant in a provinces with the possibility to filter by pollutant group

3.3 Map page

- Choropleth map of mean value of concentration for specific province, pollutant and time period.
- Histogram of mean value of concentration for a specific province, pollutant and time period.

3.4 Trend page

General pollutant analysis:

- Trend graph of a specific pollutant in a single station

- Cards with highlighted information about average, maximum, minimum and total data points

Specialized analysis:

- Graph of multiple pollutants filtered by time period and province with the possibility to visualize it with a moving average of 7 or 14 days.
- Cards with highlighted information about average, maximum, minimum and total data points

3.5 Download

- Download the graph as an image
- Download the maps as a shapefile

4. API Implementation

To implement the functionalities of the web app we define the following APIs:

Endpoint	Method	Description
/api/login	POST	Sends username and password, returns success or redirect to sign in.
/api/signin	POST	Sends login fields, email and type ‘user’ as default, return success or not
/api/provinces	GET	Fetch distinct provinces from the database
/api/stations	GET	Fetch the stations based on pollutant, if not pollutant fetch all
/api/measurements	GET	Fetch the pollutant measurements based on ID sensor
/api/measurements_filters	GET	Fetch measurements average for a certain province, pollutant, date

/api/avg_province_time	GET	Fetch measurements average for a certain pollutant and date, if not date last 7 days average

5. Database

5.1 Data Source

Sensor and measurement data were obtained through the following Lombardy region data APIs and stored in local databases used for processing:

Measurement from: https://www.dati.lombardia.it/Ambiente/Dati-sensori-aria-dal-2018/g2hp-ar79/about_data thought the API https://www.dati.lombardia.it/resource/g2hp-ar79.json

Station from: https://www.dati.lombardia.it/Ambiente/Stazioni-qualit-dell-aria/ib47-atvt/about_data thought the API https://www.dati.lombardia.it/resource/ib47-atvt.json

5.2 PostGRE database

Table	Usage	Fields
measurement	For storing data about pollutants measurements done by each sensor	sensor ID (PK), data (PK), value, state, operated ID
station	For storing data about stations, location and typology	sensor ID (PK), sensor type name, measurement unit, station ID, station name quota, province, comune, storico, datastart, datastop, utm nord, utm est, lat, lng, location txt, location.
user	For keep track of all the users who	id (PK),

	want access advanced functionalities	user, email, password, type, data

The “sensor ID” field connect the tables measurement and station.

6. Application Deployment and Execution

To execute the Air Quality Analysis web application, the following steps must be completed. First, ensure that Python 3.8 or later, PostgreSQL, and the PostGIS spatial extension are properly installed. Clone the project repository and install the required Python dependencies by executing pip install -r requirements.txt from the project root directory. Next, create a PostgreSQL database named lombardia_air_quality and configure a user with appropriate privileges. Navigate to the database/ directory and sequentially run the Jupyter notebooks (database_station.ipynb, database_measurement.ipynb, and database_user.ipynb) to enable the PostGIS extension, retrieve air quality and station data from the official Lombardia regional APIs, and populate the database accordingly. Once the database is prepared, start the backend API service by running python server.py (which will be accessible at http://localhost:5001), and launch the Dash frontend interface by running python app.py (available at http://localhost:8000). Both the backend and frontend must remain active concurrently for the application to function correctly.

7. User interface design

7.1 Home Page

The main page serves as the welcoming landing interface of the GeoAir web application. It features a clean, user-friendly design that provides a concise overview of the platform’s purpose. Prominently displayed are two central buttons, "Explore Map" and "View Trends", which direct users to the corresponding analysis pages. Additionally, a navigation bar at the top grants access to different sections of the application, including Home, Map, Trends, and Login, ensuring smooth and intuitive navigation across the platform.

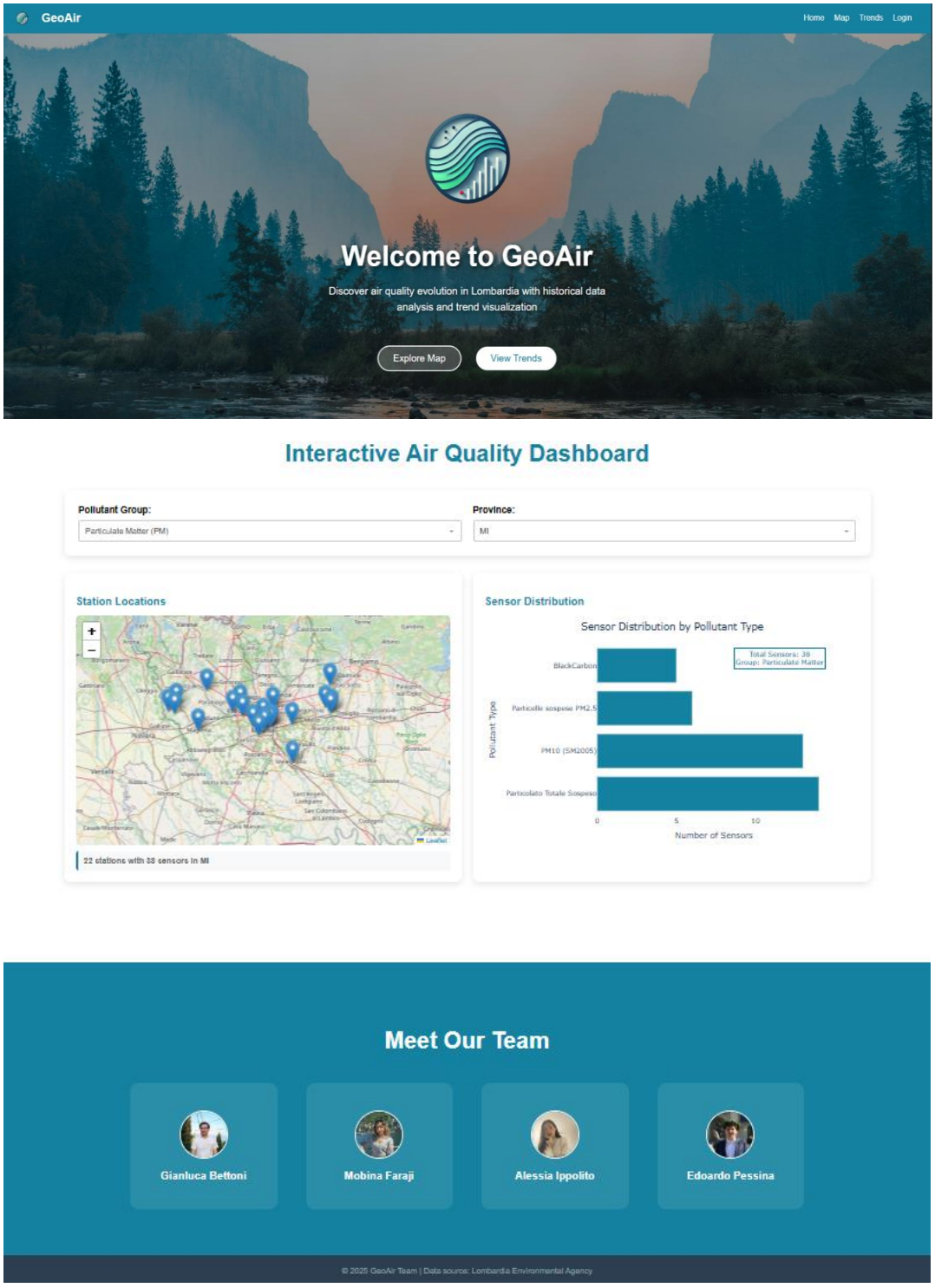


Figure 1: Home Page

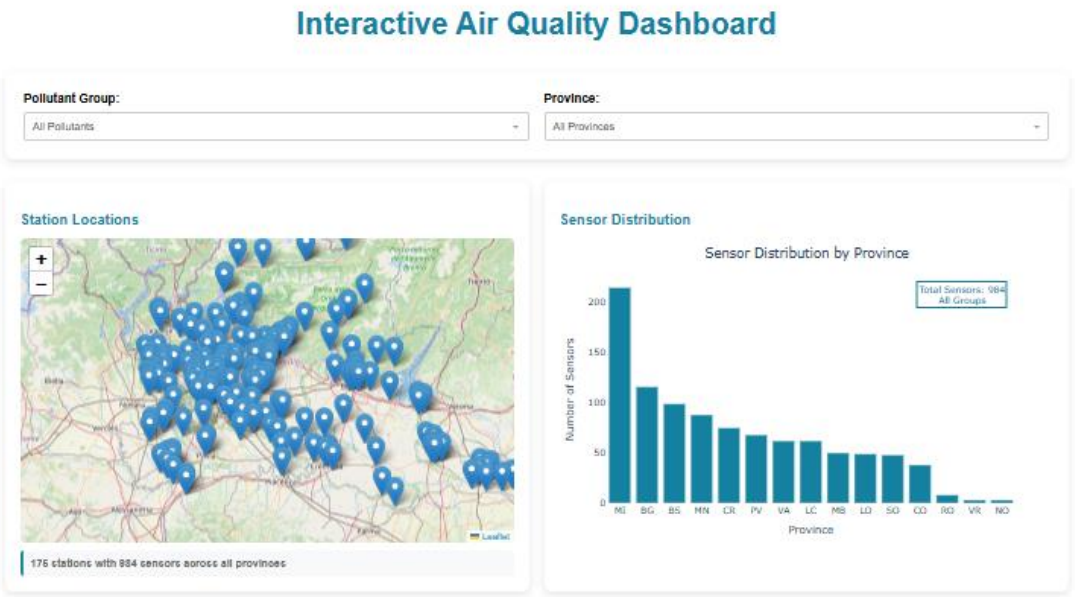


Figure 2: Map Page

The lower section of the main page presents an interactive station map alongside a sensor distribution bar chart. This interface allows users to explore sensor availability across provinces and pollutant groups. Filters enable selection by both pollutant category and geographic region, offering an intuitive overview of station coverage and pollutant monitoring density.

7.2 Login / Registration

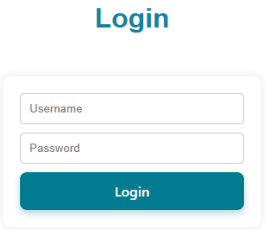


Figure 3: Login

The login page allows users to access their account if they are already registered. When the user exists in the database, they can enter their credentials to log in. If any required fields are missing or incorrect, an error message will appear in red. Once the login is successful, the page will remember the user's current account.

If the user is not found in the database, the page switches to sign-in mode by showing additional fields, such as email and confirm password, so the user can complete the registration process and create a new account.

Once the user has successfully logged in, the login page will show a personalized greeting indicating the current account and prompt the user to log out if they wish to switch to a different account.

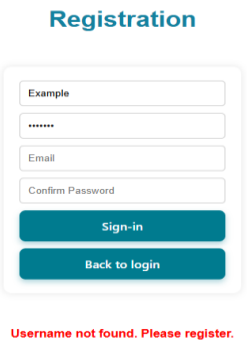


Figure 4: Registration

Logging out ends the current session and clears the user's account information from the page, requiring the user to log in again to access protected features such as the map and trend pages.

7.3 Map Page

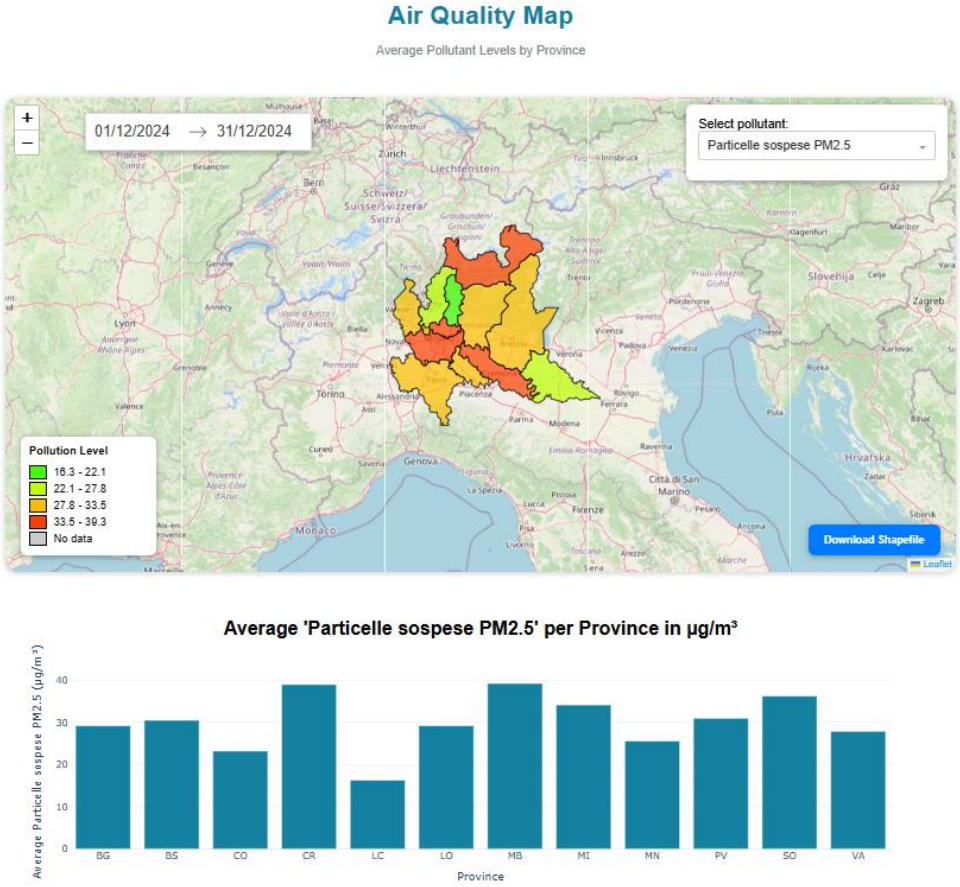


Figure 5: Map Page

The Map View page features a color-coded choropleth map that enables users to explore spatial variations in air quality across provinces. Through an interactive interface, users can:

- Select a specific pollutant type
- Define a custom date range
- Download the corresponding shapefile for further geospatial analysis

Complementing the map, a bar chart below provides a numerical summary of average pollutant concentrations per province.

7.4 Trend Page

The **Trend Page** is organized into two main sections: **General Pollutant Analysis** and **Specialized Analysis**.

1. **General Pollutant Analysis**
   Users can select a specific pollutant and a monitoring station. The application returns a time series plot showing the concentration levels of the selected pollutant over time, starting from the earliest available data. In addition, **informative summary cards** show average minimum, maximum, concentration and total number of data points within the selected time frame.

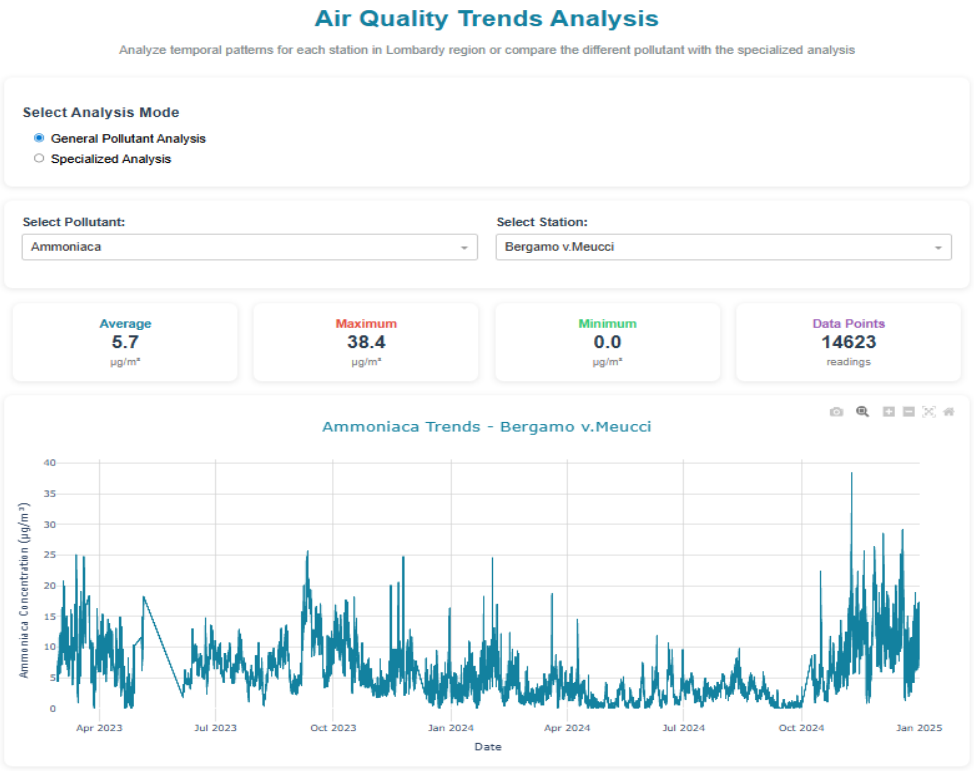


Figure 6: General Pollutant Analysis

2. **Specialized Analysis**
   Users can select multiple pollutants, a province, the desired smoothing option (raw data, 7-day, or 14-day moving average), and a custom date range for the analysis. The output includes:

   - A **dynamic multi-line chart** showing the time series of each selected pollutant.

   - **Informative summary cards** for each pollutant, presenting key statistics: average concentration, minimum, maximum, and total number of data points within the selected time frame.

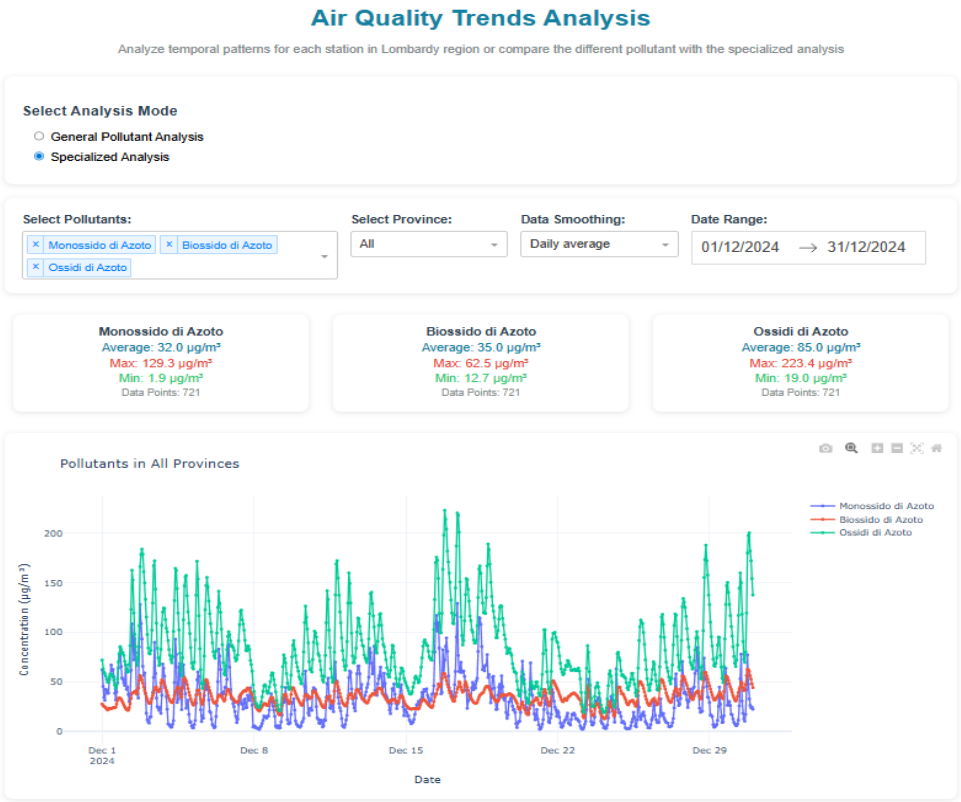


Figure 7: Specialized Analysis

This page allows for both broad and in-depth exploration of air quality trends, enabling users to visualize and compare pollutant behavior across time, locations, and smoothing methods.

8. Bibliography

Regione Lombardia – *Dati sensori aria dal 2018*. Retrieved from: https://www.dati.lombardia.it/Ambiente/Dati-sensori-aria-dal-2018/g2hp-ar79

Regione Lombardia – *Stazioni qualità dell'aria*. Retrieved from: https://www.dati.lombardia.it/Ambiente/Stazioni-qualit-dell-aria/ib47-atvt

Plotly Technologies Inc. (2024). *Dash Documentation*. Available at: https://dash.plotly.com/

Flask Documentation. (2024). *Flask Web Framework*. Available at: https://flask.palletsprojects.com/

PostgreSQL Global Development Group. (2024). *PostgreSQL Database Documentation*. Available at: https://www.postgresql.org/docs/

Python Software Foundation. (2024). *Official Python Documentation*. Available at: https://docs.python.org/3/

GeoPandas Development Team. (2024). *GeoPandas Documentation*. Available at: https://geopandas.org/

Shapely Developers. (2024). *Shapely Documentation*. Available at: https://shapely.readthedocs.io/