DESIGN DOCUMENT

POLITECNICO
MILANO 1863

Software Engineering for Geoinformatics

A. A. 2024 / 2025

Davide Galluzzo, Silvia Macchi, Giovanni Pasut, Sveva Zanetti

SUMMARY

1. INTRODUCTION

1.1 CONTEX AND MOTIVATIONS:

In recent years, growing awareness of pollution and its profound implications on public health has underscored the urgent need for accessible and transparent environmental data. Among the various environmental factors, air quality stands out as a critical point for both individual well-being and ecosystem health.
Regional and local authorities, public health agencies, and citizens increasingly require timely and accurate information about air pollution to take preventive actions and shape effective policies. So, the aim of the project is to develop a client-server application that analyses and visualizes air quality data retrieved from different regional sources:

- Dati sensori aria dal 2018
- Stazioni qualità dell'aria

By offering an interactive and intuitive dashboard, the application empowers users to monitor patterns, interpret critical information, and provide a better understanding of pollution trends and exposure risks.

1.2 PURPOSE:

This document defines the design objectives and technical direction of the project, acting as a bridge between the Requirements Analysis and Specification Document (RASD) and the implementation phase. It provides the foundation upon which the project has been built, guiding development while ensuring that both client needs and engineering best practices are met.

1.3 SCOPE:

The project involves the development of a client-server application designed to support users in querying and visualizing air quality data retrieved from *Dati Lombardia* sensor dataset. The system will process historical measurements to deliver meaningful insights into pollution distribution, temporal trends, and potential exposure risks. The application will serve as a visualization and analysis platform that will enable users to:

- Visualize and interact with air quality data through an intuitive map interface
- Analize pollution trends over time and across different areas
- Create customized data views based on selected parameters and timeframes
- Enhance map-based visualizations with additional data layers

1.4 INTENDED AUDIENCE:

The main audience we expect to use the site will be:

- **Environmental Agencies and Public Health Organizations**: the application will support decision-makers by providing accessible insights into air quality trends and pollution distribution, helping policy-making, health advisories, and environmental planning.
- **Regional and Local Administrations**: local authorities can use the platform to monitor air quality in their territories, assess environmental impact, and develop targeted interventions.

- **Citizens**: Individuals interested in understanding the air quality can use the application to stay informed and make decisions related to outdoor activities.
- **Students and Educators**: the platform can be a valuable educational tool for learning about pollution patterns, data interpretation, supporting classroom work and independent research.

1.5 CORE FUNCTIONALITY OF THE SYSTEM:

Data Collection and Integration

- The system retrieves and processes air quality data: it integrates historical measurements to build a comprehensive dataset that includes pollution levels, pollutant types, and sensor locations.

Visualization

- Air quality data is presented through an interactive and intuitive dashboard that allows users to explore pollution levels over time and space. Data can be visualized in different map-based views, showing the geographic distribution of sensors and pollutant levels, and graph-based visualizations to highlight trends and comparisons.

Data Exploration and Customization

- The system incorporates user role differentiation, distinguishing between *standard* and *expert* users at the beginning of the client application interface: these roles determine the level of interaction available throughout the system. Both standard and expert users are allowed to select parameters as pollutants, time windows, provinces, and municipalities to visualize time series, average concentrations, and sensor data. However, expert users gain extended capabilities, including defining custom analysis areas, setting pollutant-specific thresholds and generating detailed time series across years.

2. COMPONENT

2.1 COMPONENT DESCRIPTION:

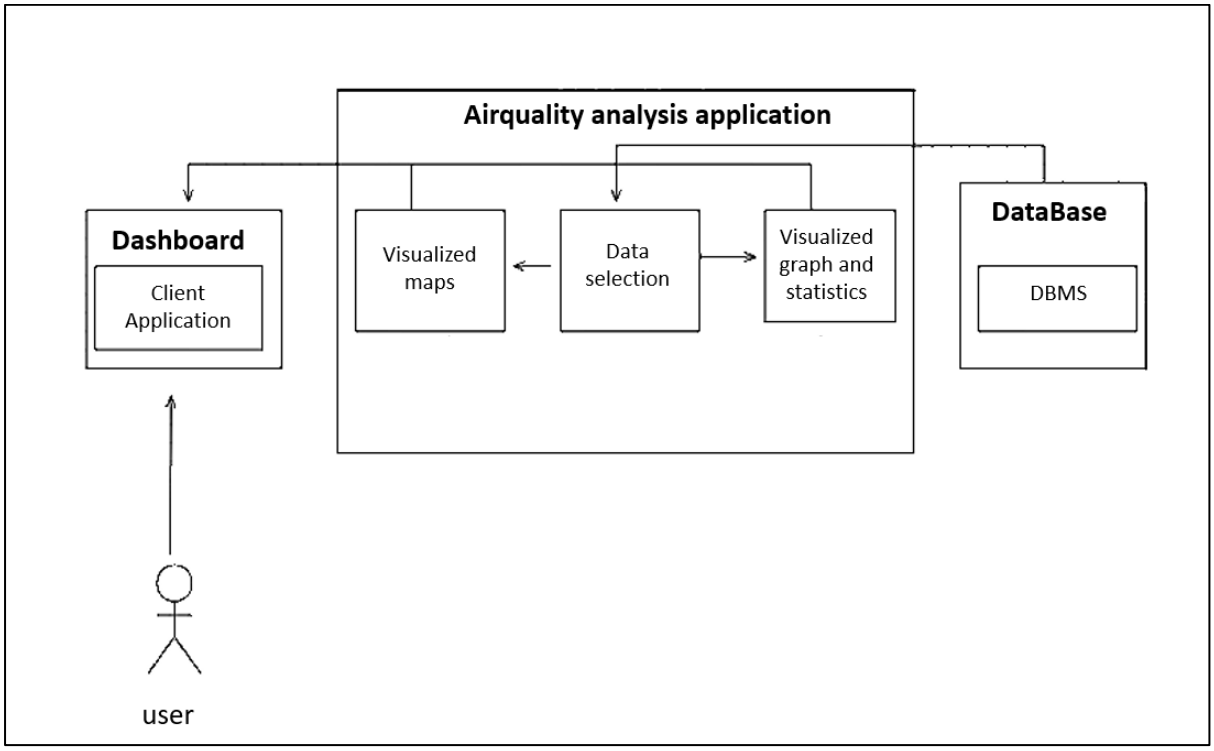

Figure 1: application diagram

- **Dashboard:** the visualization interface is built using the Dash framework and runs directly within Jupyter Notebooks. It is primarily designed for data exploration, offering visual access to results such as air quality maps, charts, and statistics. The interface manages user interactions, handles input parameters, and communicates with the backend by sending HTTP requests. It does not carry out any data processing itself; instead, it focuses on displaying the processed results returned by the server.
- **Server:** the server forms the backbone of the system: it handles incoming client requests, processes them, interacts with the database, and delivers appropriate results. The server is responsible for managing both data storage and all processing tasks.
- **Database:** the database is responsible for the storage of the system. It hosts all the air quality data and responds to queries made by the server, returning structured data used for visualization and analysis
- **User**: two types of users are designated
    - **Standard User**: A non-technical user, such as a citizen or student who interacts with an essential dashboard to view general air quality information.
    - **Expert User**: A technical or professional user, such as a researcher or public administrator who needs access to customizable analysis tools.

2.2 MAPPING COMPONET AND REQUIRMENT:

Data Integration Component:

Req	Title
DI-1	Store air quality data from Dati Lombardia
DI-2	Store sensor data from Sensor Data Table
DI-3	Combine air quality data with sensor data using the sensor ID
DI-4	Filter out invalid data to ensure data quality
DI-5	Update the database when new data is uploaded to Dati Lombardia and Sensor Data
DI-6	Integrate geometric data (shape of province/municipality and contour lines)
TDV-1	Use PostgreSQL/PostGIS for spatial database management

Table 1: Data Integration Requirements

Data Processing Component:

Req	Title
DP-1	Query the database to return only the requested data
DP-2	Perform data aggregation and statistical analysis

Table 2: Data Processing Requirements

Data visualization component:

Req	Title	Standard User	Expert User
DV-1	Allow users to choose between standard and expert profiles	**x**	**x**
DV-2	Allow standard users eventually to upgrade to expert profiles	**x**	
DV-3	Enable selection between different basemaps and layers of interest	**x**	**x**
DV-4	Notify users if no data is available for their selected visualization	**x**	**x**
DV-5	Allow visualization of monthly time series for specific pollutants measured by a selected sensor	**x**	**x**
DV-6	Display locations of sensors measuring selected pollutants on a map	**x**	**x**
DV-7	Visualize monthly time series of each pollutant for a selected province or municipality	**x**	**x**
DV-8	Display histograms of average pollutant concentrations per province, including global averages	**x**	**x**
DV-9	Show average pollutant concentrations over a selected time on a map, differentiated by area and polluted level	**x**	**x**
DV-10	Visualize topographic maps showing pollutant concentrations correlated with altitude	**x**	**x**
DV-11	Graph the correlation between monthly pollutant concentrations and sensor altitudes	**x**	**x**
EU-DV-1	Allow user to define polygons and analyze contained sensor data		**x**
EU-DV-2	Enable definition of thresholds to analyze pollutant data over time and province		**x**

Table 3: Data visualization Requirements

Client-Server component:

Req	Title
TCS-1	Provide data from database
TCS-2	Return data in JSON format to the client
TCS-3	Develop an interactive dashboard using Jupyter Notebook

Table 4: Client-Server Requirements

3. SOFTWARE STRUCTURE

The system adopts a client-server architecture and consists of three main layers: the presentation layer, the application layer, and the data layer. Each layer has a specific role in the processing, management, and delivery of air quality data.

1. **Client Side / Dashboard**
   This layer is composed of the dashboard and is responsible for interacting with the user, collecting input parameters, and displaying the output (maps, graphs, statistics) dynamically as the button is clicked.

2. **Server Side**
   The backend logic is handled by a Flask application that exposes several REST API endpoints. These endpoints process incoming HTTP requests from the dashboard, interpret the parameters, and construct SQL queries accordingly. The Flask app handles both simple queries and complex ones.
   The application layer is composed of several modules:
   - *Data Selection*: constructs and executes queries to extract relevant data based on the selected parameters.
   - *Visualized Maps and Statistics***:** returned geospatial visualizations of air quality indicators along with analytical outputs such as time series, average pollution levels, and other statistical summaries.

3. **Database**
   The system is built around a PostgreSQL database that organizes and manages structured air quality data. This includes tables for sensors, stations, measurements, and geographical information about municipalities and provinces. The application layer interacts with this database to retrieve and aggregate relevant data based on the parameters provided by the client.

4. DATABASE

4.1 INTRODUCTION:

The database system plays a central role in the client-server application, acting as the backbone for integrating, storing, and managing air quality and sensor data. It processes data from administrative sources, merging them into a unified dataset that includes hourly pollutant measurements, sensor metadata, and geolocation information. This relational structure ensures data integrity and allows for trend analysis and pollution exposure assessments.
The data for this project, including information about sensor locations and air pollution levels, are taken from the following providers, associated to the Lombardy Region Administration:

- Dati sensori aria dal 2018
- Stazioni qualità dell'aria

The dashboard interacts with the server, which performs operations on a PostgreSQL database. Thanks to the integration of PostgreSQL and its PostGIS extension, the application can efficiently handle and analyze advanced spatial data.

4.2 DATABASE SCHEMA:

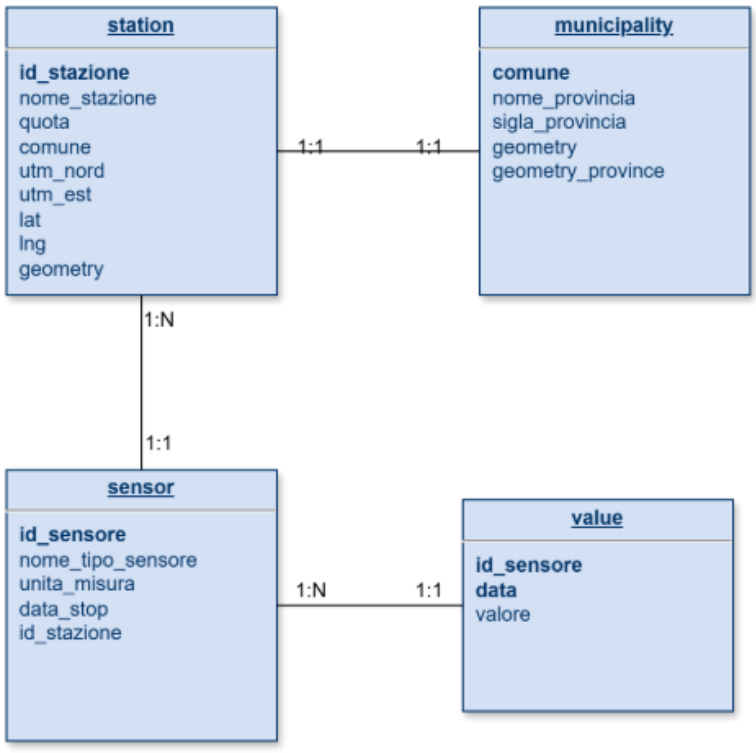

Figure 2: Database schema

4.3 DATABASE TABLE DESCRIPTION:

A common solution for managing structured data is a relational database, which organizes information into tables. Each table contains rows (also known as tuples) that follow a consistent format defined by columns (or attributes). The overall structure (defining the tables, their columns, and the relationships between them) is known as the database schema.
In particular, the database is composed of four tables: Value, Sensor, Station, and Municipality.
To ensure only relevant and necessary information are collected inside the schemas, only some specific columns are considered from the original providers: considering the Value table for instance, the columns "idOperatore" and "Stato" were excluded, as the dataset was filtered to retain only records with a valid status ("VA").
The filtering process includes the following steps:

- Access the Query section from Dati Sensore aria dal 2018.
- Under “filtra”, “seleziona stato è VA".
- Select only the columns: *id_sensore*, *data* and *valore*.

The resulting data was downloaded in CSV format and used to populate the Value table.

Value

The Value table stores environmental measurements recorded by sensors. Each record includes the primary keys: *id_sensore* (identifying the sensor) and *data* (the date of the measurement), along with *valore*, a numeric field representing the measured concentration of a pollutant.
The attribute *valore* is associated with the type of pollutant being measured through a relationship with the *Sensor table*. Inside the *Sensor table*, each sensor is uniquely identified by *id_sensore*, and the type of pollutant it measures is specified in the *nome_tipo_sensore* column.

Attribute	Type	Description
id_sensore	Numeric	Unique identifier for the sensor device
data	Timestamp	date and hour of the measurement (e.g., 2024-12-19 18:00:00)
valore	Numeric	Value of the measurement

Table 5: Value table description

Sensor

The Sensor table contains information about the type of pollutant from fixed monitoring stations in the ARPA Lombardia network. Each record includes a unique sensor identifier as a primary key *id_sensore,* the specific type of sensor *nome_tipo_sensore* and other meaningful attributes. This table is linked to additional sensor metadata such as the attribute *data_stop,* which indicates the last date recorded by inactive sensors, and the attribute *unita_misura* referring to the unit of measurement, making them relevant fields for analyzing different pollutant agents.

The filtering process for this table involves:

- Select and filter only the column from the provider: *id_sensore*, *nome_tipo_sensore*, *unita_misura*, *id_stazione*, and *data_stop*.

The selected data is downloaded in CSV format to populate the Sensor table.

Attribute	Type	Description
id_sensore	numeric	Unique identifier for the sensor device
nome_tipo_sensore	text	The type of measured pollutant (e.g., NO2)
unita_misura	text	Unit of value of the pollutant (if provided)
data_stop	text	If provided, it indicates inactive sensors
id_stazione	numeric	Unique identifier for the stations

Table 6: Sensor table description

Station

The Station table provides detailed spatial information about the location of the air quality monitoring stations. Each station is uniquely identified by the primary key *id_stazione* and described by general attributes as *nome_stazione*, elevation above sea level *quota*, and the municipality column *comune*. Spatial location is specified through UTM coordinates (*Utm_Nord*, *UTM_Est*), geographic coordinates *lat*, *lng* and a geometric point *geometry* that represents the specific location of the station. This table is essential for mapping stations, associating them with sensors, and performing spatial analyses on air quality data.
The filtering process for this table involves:

- Select and filter only the column from the provider: *id_stazione, nome_stazione, quota, comune, utm_nord, utm_est, lat,* and *lng*.

The grouped data was downloaded in CSV format to populate the Station table.

Attribute	Type	Description
id_stazione	numeric	Unique identifier for the stations
nome_stazione	text	It indicates the name of the specific station
quota	numeric	Indicates the elevation of the sensor's physical location meters above sea level.
comune	text	It indicates the Municipality in which the specific station belongs.
utm_nord	numeric	UTM coordinate (northing) of the specific station
utm_Est	numeric	UTM coordinate (easting) of the specific station
lat	double precision	Latitude value of the specific station
lng	double precision	Longitude value of the specific station
geometry	geometry	Indicates the spatial location of each station as a geometric point

Table 7: Station table description

Municipality

The Municipality table contains geographical and administrative information about municipalities and provinces within the Lombardy region. Each entry includes the municipality name *comune* as a primary key, the corresponding province name *nome_provincia*, and its abbreviation *sigla_provincia*. The spatial extent of each municipality is defined by the *geometry* attribute, which stores polygon shapes representing boundaries (reprojected to EPSG:4326), while the boundaries of each province are defined by the *geometry_province* attribute.
This table is used to spatially contextualize sensor and station data within their municipal and provincial boundaries.
The municipal boundaries considered are obtained separately by downloading the corresponding shapefile from the Lombardy Region dataset (initially as a CSV format). This dataset is imported into QGIS software, where it is subsequently exported as a shapefile that includes the required geometry attributes.
Similarly, the provincial boundaries are retrieved from the Geoportal of the Lombardy Region.
These two boundaries are then combined and displayed on a base map to improve the visualization of the analysis results. They include both administrative boundaries and polygon geometries.
The shapefile containing provincial boundaries was originally defined in the EPSG:32632 coordinate reference system (WGS 84 / UTM zone 32N), which uses projected metric coordinates.

To ensure consistency with the Station table, whose coordinates are expressed in geographic format (latitude and longitude) using EPSG:4326 (WGS 84); the dataset has been reprojected accordingly.
This reprojection enables accurate spatial operations and alignment between the two tables.

Attribute	Type	Description
comune	text	name of the municipality considered
nome_provincia	text	name of the province considered
sigla_provincia	text	the initials of the province considered (e.g.Milano = MI)
geometry	geometry	Indicates the boundary of the Lombardy municipalities as a geometric polygon (Reproject to epsg 4326)
geometry_province	geometry	Indicates the boundary of the Lombardy province as a geometric polygon (Reproject to epsg 4326)

Table 8: Municipality table description

4.4 SQL AND JOIN OPERATIONS:

SQL, or Structured Query Language, is the standard language used to communicate with relational databases. It enables users to define the structure of a database through Data Definition Language (DDL) and to manipulate the stored data using Data Manipulation Language (DML).
To combine data from multiple tables, the relations of the web application here considered are handled accordingly through the JOIN command, which performs operations to return only the rows where there is a match in both tables based on a specified column (INNER JOIN).
For instance, in the *DV-5 Requirement* query function the SQL JOIN operation is used to merge information from the two tables of *sensor* and *value,* by matching rows where the primary key *id_sensore* is the same in both tables.
The purpose of this join is to retrieve pollutant measurements from the *value* table that correspond to a specific sensor defined in the *sensor* table.

```
query = f"""
SELECT ST.geometry, ST.nome_stazione, S.id_sensore, S.data_stop
FROM sensor AS S
JOIN station AS ST ON ST.id_stazione = S.id_stazione
WHERE S.nome_tipo_sensore = :pollutant
"""
```

Figure 3: Example of SQL query to get station location, name, sensor ID, and stop date for specific sensors

This approach allows the application to efficiently fetch only the relevant environmental data tied to a specific sensor and pollutant, combining data from different sources into a coherent result set that is then returned to the client in JSON format.
In addition to combining data, the requirements specified in the Design Document need aggregate functions such as AVG, which are used to perform calculations across rows of data.
In the *DV-8 Requirement* query function, the AVG function is used to calculate the average value of pollutant measurements across a specified time range and for a specific pollutant type. The query retrieves data from multiple related tables, joining *sensor*, *value*, *station* and *municipality* to associate each measurement with its corresponding geographic location.

```
query = """
    SELECT M.nome_provincia AS province, AVG(V.valore) AS average_value
    FROM sensor AS S
    JOIN value AS V ON V.id_sensore = S.id_sensore
    JOIN station AS ST ON ST.id_stazione = S.id_stazione
    JOIN municipality AS M ON M.comune = ST.comune
    WHERE S.nome_tipo_sensore = :pollutant
      AND V.data BETWEEN :start_date AND :end_date
    GROUP BY M.nome_provincia
"""
```

Figure 4: Example of SQL query to compute average pollutant levels by province

When these aggregate functions are used together with the GROUP BY clause, they allow for grouping data by a particular attribute and performing aggregations within each group.
Together, the operations of projection (selecting specific columns), selection (filtering rows), joining tables, and aggregation (summarizing data) form the essential tools for querying and analyzing relational data effectively.

5. TECHNOLOGY STACK SELECTION

The following technologies have been implemented for the development of the web Platform:

Visual Studio Code
A lightweight, open-source code editor developed by Microsoft. It supports multiple programming languages and offers features like syntax highlighting, debugging, terminal integration, and an extensive ecosystem of extensions.

PostgreSQL
PostgreSQL is a powerful, open-source relational database system known for its reliability and rich features.

PgAdmin 4
A web-based administration tool for PostgreSQL databases. It provides a graphical interface to perform tasks such as writing and executing SQL queries, managing tables and schemas, monitoring performance, and visualizing data.

PostGIS
An open-source spatial database extension for PostgreSQL. It adds support for geographic and geometric data types, allowing advanced spatial queries such as distance calculations and intersections. Widely used in applications involving maps, geolocation, and geographic data analysis.

Flask
A micro web framework written in Python, used to build web applications and RESTful APIs. Flask is lightweight, easy to use, and highly customizable, making it well-suited for both simple prototypes and more complex applications.

QGIS
It is a powerful mapping and spatial analysis program that allows developers to work with geographic information to explore spatial patterns, create custom maps, and perform spatial analysis.

GitHub
A cloud-based platform for cross-collaboration using Git. It allows developers to host code repositories, track changes, collaborate on code with pull requests, manage issues, and the team's workflow.

6. CONCLUSION

This document has outlined the comprehensive design and structure of a client-server side application developed to visualize and analyze air quality data from the Lombardy Region. Motivated by the idea of developing a user-friendly app for air quality data, the system provides a centralized platform that integrates, processes, and displays pollution data collected from regional monitoring stations.
The purpose of the system is to empower various stakeholders including citizens, public agencies, and researchers, with the tools to explore spatial and temporal patterns of air pollution. This is achieved through a multi-layered architecture composed of:

- A **client-side interface** for interactive data visualization;
- A **server-side application** that handles data requests, filtering, aggregation, and returns results;
- A **PostgreSQL/PostGIS database**, which stores environmental measurements, sensor data, and geographical boundaries.

The database design, centered on four key tables (Value, Sensor, Station, Municipality), supports efficient querying and geospatial operations, crucial for contextualizing pollution data within administrative boundaries. The system applies SQL joins and spatial functions to create customized configurations based on user input.
This platform provides a valuable tool for monitoring environmental conditions, visualizing air quality data and supporting public administration data requests.