

# **Workshop 1 - Project**

## **Air Quality Analysis Platform**

### **Members:**

Jose Alejandro Cortazar – Cod.: 20181020022

Stivel Pinilla Puerta – Cod.: 20191020024

Johan Castaño Martinez – Cod.: 20191020029

### **Professor:**

Carlos Andrés Sierra

**Universidad Distrital Francisco José de Caldas**

School of Engineering

Computer Engineering Program

Databases II

September 2025, Bogotá D.C.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Business Model</b>	<b>2</b>
<b>3</b>	<b>Requirements</b>	<b>4</b>
<b>4</b>	<b>User Stories</b>	<b>6</b>
<b>5</b>	<b>Initial Database Architecture</b>	<b>8</b>

# 1 Introduction

It is a web and mobile application that collects, analyzes, and displays real-time and historical air quality data. It is designed to help citizens, governments, and businesses make informed decisions about health and the environment.

# 2 Business Model

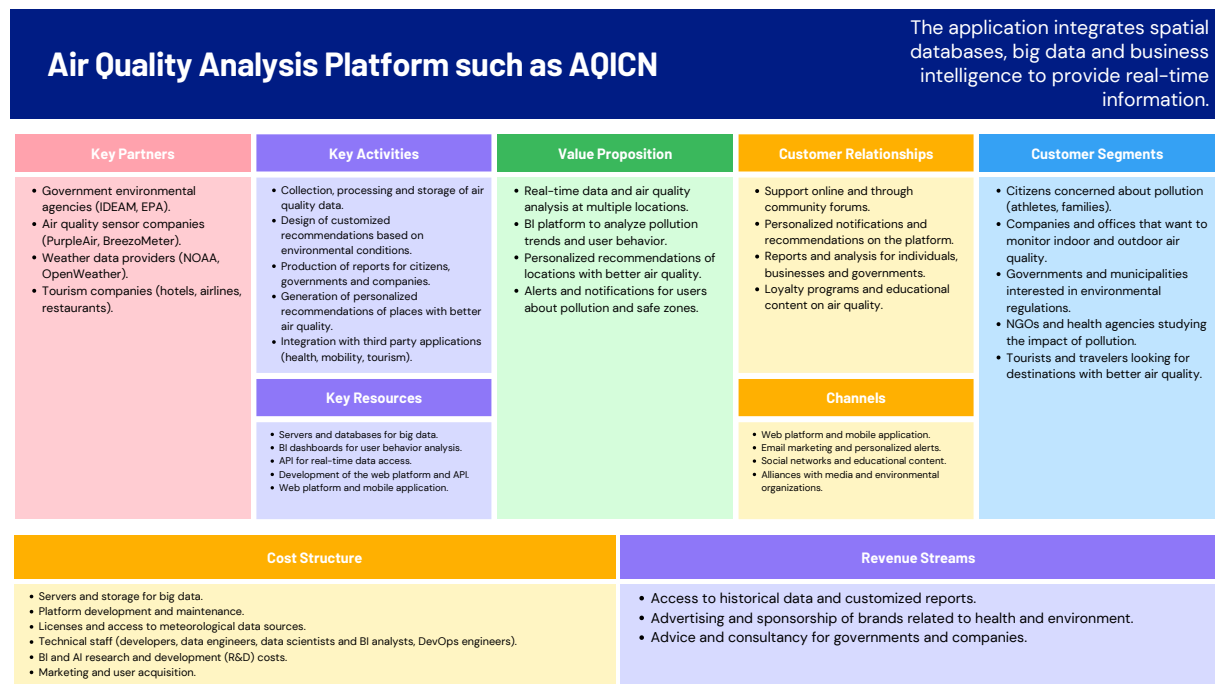


Figure 1: Business Model

## How Does It Work?

The platform operates through the following process:

- It collects data from environmental sensors and third-party APIs (such as meteorological stations and air quality sensors).
- It processes the data to clean, transform, and store it in both relational and big data databases.
- It displays useful information to users through:
  - Real-time dashboards.
  - Personalized recommendations.
  - Downloadable reports.
  - Interactive maps and graphs.

## **Who Uses the Platform?**

- Citizens who want to know if it is safe to exercise or go outdoors during high pollution periods.
- Governments that issue alerts and design public policies.
- Companies and NGOs interested in analyzing environmental impact.

## **How Does It Generate Revenue?**

- Through advertising from brands related to health and the environment.

## **What Technologies Does It Use?**

- Relational databases (SQL) and NoSQL (for real-time data).
- Business Intelligence (BI) modules.
- APIs to allow other systems to access the data.
- Architecture designed for high availability, scalability, and multi-region/multi-device access.

### 3 Requirements

ID	Type	Requirement	Associated User Stories
FR1	Functional	The system must collect real-time air quality data from multiple external sources (e.g., APIs, stations) and store it for further processing.	US1, US13, US14
FR2	Functional	The system must allow users to query and visualize historical air quality data filtered by date, location, and pollutant type.	US2, US7
FR3	Functional	The system must display air quality information in a uniform and clear manner. The origin of the data is irrelevant.	US1, US3
FR4	Functional	The system must display real-time dashboards with key performance indicators (KPIs) on air quality.	US4
FR5	Functional	The system must generate customized reports with filters for date, location, and indicators, available for download or email.	US5
FR6	Functional	The system must present interactive graphs showing air quality evolution over time.	US6
FR7	Functional	The system must allow users to export historical data in standard formats like CSV and JSON.	US7
FR8	Functional	The system must provide personalized recommendations based on user location and air quality conditions.	US8
FR9	Functional	The system must send customizable air quality alerts when thresholds are exceeded.	US9
FR10	Functional	The system must suggest certified protective products during high pollution periods.	US10
FR11	Functional	The system must display maps highlighting areas with better air quality for navigation.	US11
FR12	Functional	The system must support geographic search for air quality data by country, city, or region.	US15
FR13	Functional	The system must be responsive and compatible with mobile, tablet, and desktop devices.	US16
FR14	Functional	The system must allow users to share air quality data and reports on social media with pre-generated links and previews.	US17
NFR1	Non-Functional	Queries on large datasets ( $\geq 1$ million records) must execute in under 2 seconds 95% of the time.	US3, US12

ID	Type	Requirement	Associated User Stories
NFR2	Non-Functional	The system must support continuous streaming data ingestion 24/7 without manual intervention.	US1, US14
NFR3	Non-Functional	Data storage must be distributed and optimized for big data processing.	US1, US3
NFR4	Non-Functional	Customized reports must be generated in under 10 seconds.	US5
NFR5	Non-Functional	The recommendation engine should be updated every 10 minutes throughout the day with air quality data.	US8, US10
NFR6	Non-Functional	Air quality data and visualizations must load in less than 2 seconds for 95% of user requests.	US12
NFR7	Non-Functional	The system architecture must include fault tolerance and geographic redundancy.	US14
NFR8	Non-Functional	The system must scale horizontally to support growth in data volume and users.	US13, US14
NFR9	Non-Functional	The system UI must function correctly on all major browsers and operating systems without errors.	US16
NFR10	Non-Functional	Data consistency and user personalization must be preserved across all user devices.	US16

Table 1: Requirements

### Non-Functional Requirements Justification

**NFR2 (24/7 streaming ingestion):** Air quality monitoring requires continuous data collection since pollution levels fluctuate throughout the day and immediate detection of hazardous conditions is critical for public health.

**NFR4 (10 seconds report generation):** Complex reports with multiple filters and large datasets require processing time, but 10 seconds maintains user engagement while allowing for comprehensive data analysis.

**NFR5 (10 minutes update frequency):** Air quality APIs from external sources typically update their information every 10 minutes to 1 hour, making more frequent updates unnecessary and resource-intensive.

**NFR6 (2 seconds data loading):** Critical safety information like air quality alerts must be delivered quickly to enable timely user decisions about outdoor activities and health precautions.

## 4 User Stories

User Story ID	Role and Need	Acceptance Criteria
US1	As a Technical Administrator, I want to collect real-time data from multiple APIs so that I can provide accurate and up-to-date information on air quality.	The system receives updates at least every 10 minutes from configured sources. Processes at least 1000 data points per minute without loss. Logs successful and failed ingestions with error codes.
US2	As a Researcher/Analyst, I want to access historical data by date and location so that I can perform longitudinal analysis and scientific research.	User can filter by city, date, and pollutant type. Data export available in CSV and JSON. Historical records available for up to 3 years.
US3	As a Technical Administrator, I want to run queries over large volumes of data quickly so that I can avoid delays when searching for information.	Queries over 1 million records return in under 3 seconds. Indexes and partitions are used for performance.
US4	As a Public Policy Manager, I want to access dashboards with real-time analysis so that I can issue alerts or recommendations to the public.	Dashboards include real-time maps, graphs, and alerts. Automatic refresh without manual reload. Critical thresholds can be configured for alerts.
US5	As a Public Policy Manager, I want to generate customized reports on pollution trends so that I can design evidence-based public policies.	User can choose indicators, date ranges, and export formats. Reports are downloadable in PDF or sent via email. Graphs are auto-generated from selected data.
US6	As a Citizen, I want to view interactive graphs about air quality evolution so that I can understand changes and make informed decisions.	Filters for location, date, and pollutant available. Charts update dynamically with parameter changes. Visualizations load in under 2 seconds.
US7	As a Researcher/Analyst, I want to export historical data in standard formats so that I can analyze it with my own statistical tools.	Interface allows selection of location, date, and format. Files download successfully without errors. Download limits prevent system overload.
US8	As a Citizen, I want to receive personalized recommendations based on air quality so that I can know if it is safe to do outdoor activities.	Location and user preferences are considered. Alert color coding (green, yellow, red) is used. Suggested actions are clearly displayed.

US9	As a Citizen, I want to receive early warnings when air quality is harmful so that I can take precautions in time.	User can set alerts by pollutant and critical threshold. Notifications sent via email. Alert triggers when AQI surpasses configured limits.
US10	As a Citizen, I want to receive suggestions for protective products so that I can protect my health during high pollution levels.	Suggestions appear only during high pollution periods. Products are certified and include links. User can disable this feature in preferences.
US11	As a Citizen, I want to see areas with better air quality so that I can avoid the most polluted zones when moving.	System displays a map with AQI levels per area. Users can compare different city places.
US12	As a Citizen, I want to load air quality data quickly so that I can access information without delay.	Air quality data loads in under 2 seconds normally. System handles 100 concurrent users without slowdown. Cache is used for frequently accessed queries.
US13	As a Technical Administrator, I want to handle traffic peaks without failure so that I can maintain user experience during high demand.	Stress test simulates 1000 concurrent users. System remains responsive under load. Latency remains below 5 seconds per request.
US14	As a Technical Administrator, I want to keep the platform available 24/7 so that I can ensure constant access to information.	Platform includes geographic redundancy. Uptime monitoring with automated alerts is enabled. Monthly availability is $\geq 99.9\%$ .
US15	As a Citizen, I want to check air quality in different regions so that I can plan trips and activities.	Users can switch between countries/regions easily. Information is shown in local language where possible.
US16	As a Citizen, I want to access the platform from various devices so that I can always access data regardless of the device.	Responsive design across mobile, tablet, and desktop. Core functions work on all devices.
US17	As a Citizen, I want to share air quality data on social media so that I can inform and raise awareness among others.	Sharing available on X, Facebook, Instagram, WhatsApp. Preview includes summary and image. Links and QR codes are auto-generated for sharing.

Table 2: User Stories

**Corrections** Based on instructor feedback, one improvement was the inclusion of roles. New roles and acceptance criteria were added.



## 5 Initial Database Architecture

### Architecture

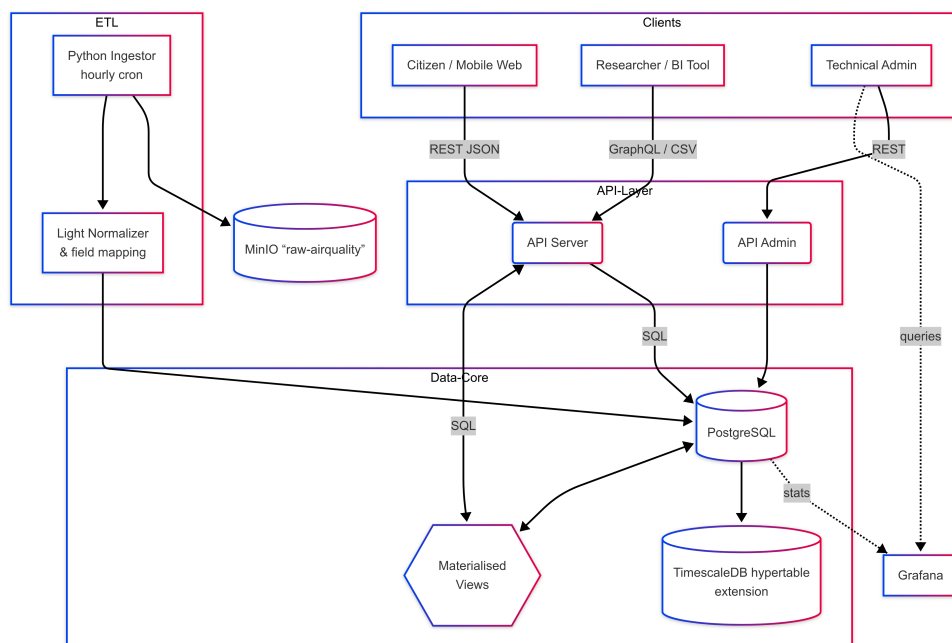


Figure 2: High-level architecture diagram

### 1. Ingestion and Raw Storage

A lightweight Python service (**Ingestor**) polls air quality endpoints at regular intervals. Each JSON payload is first persisted—unchanged—in a MinIO bucket (**raw-airquality**) to ensure auditability and allow for future replay.

### 2. Normalization and Relational Store

A mapping layer converts heterogeneous field names, units, and AQI scales into a unified schema before inserting the data into a partitioned PostgreSQL database. Monthly partitions, created using TimescaleDB, reduce index scan sizes and ensure sub-second query performance for time-localized filters.

### 3. Query Acceleration

Materialized views are refreshed concurrently to avoid blocking read operations, providing fast access to aggregated insights.

### 4. API and User Access

An API layer exposes REST and GraphQL endpoints for citizens, researchers, and BI tools. Administrative users access the database through a dedicated API, while query latency and slow statements are monitored via Grafana dashboards.

## Diagram ER

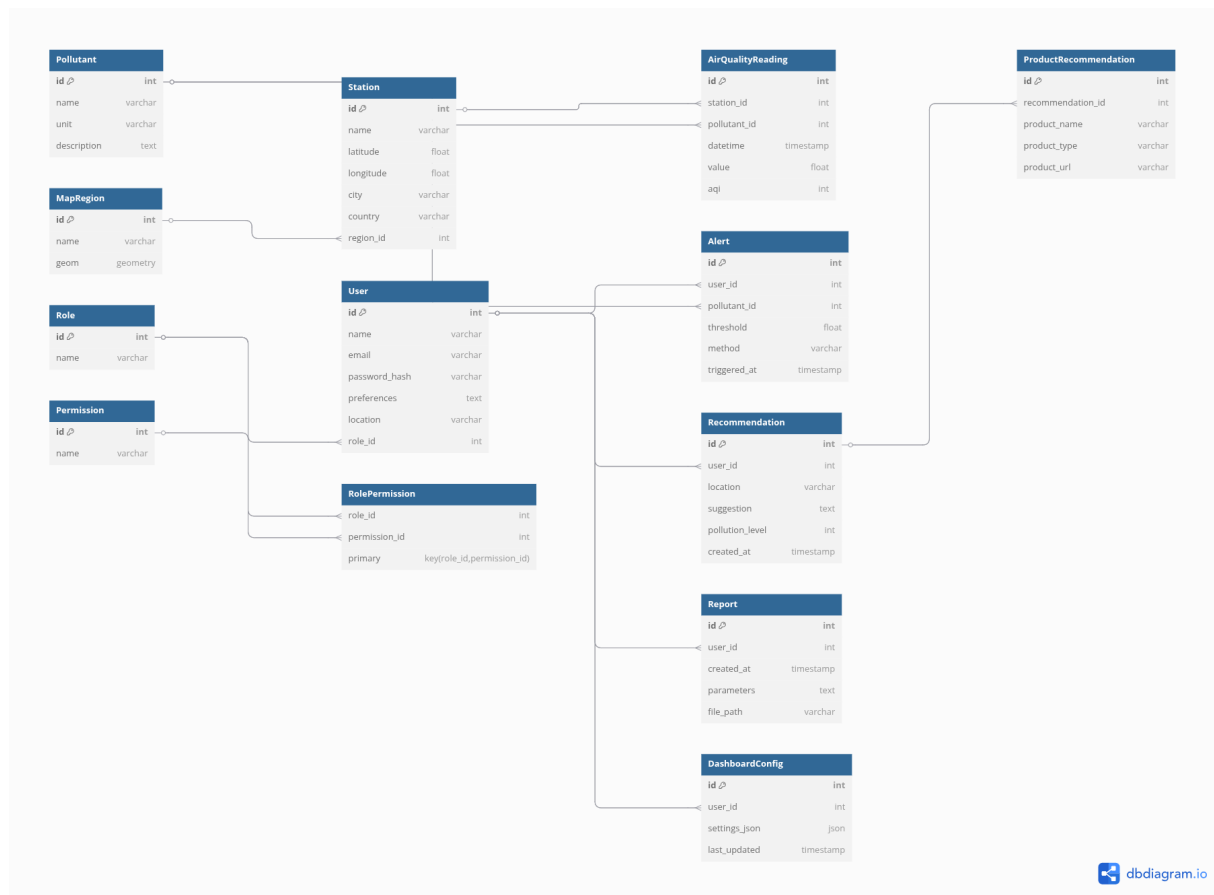


Figure 3: ER Diagram

Component	Entity Name	Description	Attributes
Geospatial Layer	Station	Monitoring station metadata and location.	id (PK), name, latitude, longitude, city, country
Geospatial Layer	AirQuality Reading	Air quality measurements from stations.	id (PK), station_id (FK), datetime, pollutant_type, value, aqi
Geospatial Layer	Pollutant	Catalog of pollutants and their units.	id (PK), name, unit, description
Customer Layer	User	Platform user and profile settings.	id (PK), name, email, password_hash, preferences, location
Recommendation Engine	Alert	Alert configurations and triggered events.	id (PK), user_id (FK), pollutant_type, threshold, method, triggered_at

Component	Entity Name	Description	Attributes
Recommendation Engine	Recommendation	User-specific recommendations based on AQI.	id (PK), user_id (FK), location, suggestion, pollution_level, created_at
Recommendation Engine	Report	Generated analytical reports.	id (PK), user_id (FK), created_at, parameters, file_path
Geospatial Layer	MapRegion	Regions used for geospatial filtering and navigation.	id (PK), name, geom (geometry)
Customer Layer	Dashboard Config	User preferences for dashboard visualizations.	id (PK), user_id (FK), settings_json, last_updated
Recommendation Engine	Product Recommendation	Suggested certified products during high pollution levels.	id (PK), recommendation_id (FK), product_name, product_type, product_url
Customer Layer	Role	Defines user roles in the platform (e.g., Admin, Citizen, Analyst).	id (PK), name (unique)
Customer Layer	Permission	Specifies distinct permissions that can be granted to roles.	id (PK), name (unique)
Customer Layer	Role Permission	Associates roles with permissions in a many-to-many relationship.	role_id (FK), permission_id (FK), PRIMARY KEY (role_id, permission_id)

Table 3: Entities