Real-Time Air Quality Monitoring and Recommendation System: A Data-Driven Software Approach

1st Johan Esteban Castaño Martínez
Systems Engineering Program
Faculty of Engineering
Francisco José de Caldas District University
Bogotá, Colombia
jecastanom@correo.udistrital.edu.co

2nd Stivel Pinilla Puerta
Systems Engineering Program
Faculty of Engineering
Francisco José de Caldas District University
Bogotá, Colombia
spinillap@correo.udistrital.edu.co

***Abstract*—This research project presents the development of a software system designed to integrate real-time data ingestion, processing, and visualization using a hybrid database architecture. The system combines relational databases (PostgreSQL) for structured, transactional data and NoSQL databases (MongoDB) for flexible, high-volume document storage. It enables efficient management of large-scale air quality datasets, supports business intelligence features such as dashboards and custom reports, and offers personalized user recommendations. The architecture ensures scalability, high availability, and multiregional access while optimizing performance for both analytical and operational workloads. Initial tests confirm the system's capacity to process streaming data, support multiple user roles, and deliver actionable insights through a web-based interface.**

***Index Terms*—Hybrid Database Architecture, Real-Time Data Processing, Business Intelligence, Data Visualization, Environmental Data Systems, Scalable Web Application, Big Data Analytics.**

I. Introduction

The development of modern information systems requires integrating scalable architectures, real-time processing and an interactive user experience. In this context, designing software to manage large volumes of heterogeneous data has become a key challenge for projects with high availability, real-time analysis and customization requirements. Traditionally, solutions oriented to environmental monitoring have shown technical limitations, such as low flexibility, difficulty of scaling and lack of advanced analysis.

This project proposes the design and implementation of a modular platform focused on the management, analysis and visualization of data related to air quality. The system integrates a hybrid database architecture, using PostgreSQL for structured data and MongoDB for semi-structured documents, allowing efficient and flexible ingestion from multiple sources. In addition, it incorporates a recommendation engine, business intelligence dashboards and real-time notification mechanisms. The software is built as a scalable web application, with multi-device access, designed to support both mass queries and specific user needs.

II. Methods and Materials

A. High-Level System Architecture

The platform follows a modular and scalable architecture to support real-time air quality data ingestion, processing, and visualization. The system is composed of multiple integrated components, each responsible for a key stage of the data lifecycle.

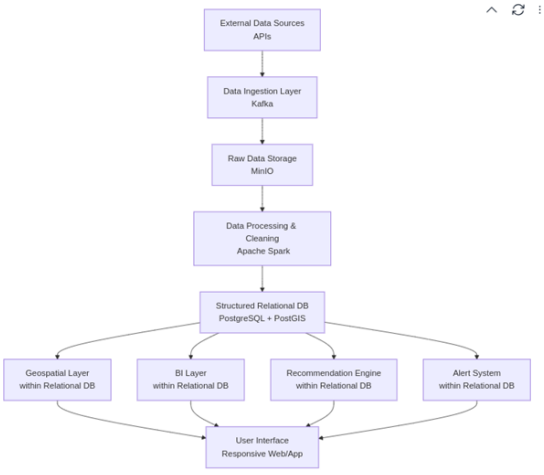


Fig. 1. High-Level Architecture of the Platform

Component Descriptions and Tools Used:

- **External Data Sources:** Public APIs and monitoring stations are integrated using RESTful connections.
- **Data Ingestion Layer:** Utilizes Apache Kafka for continuous, real-time streaming of data from multiple sources.
- **Raw Data Storage (Data Lake):** Uses MinIO for storing unstructured data, supporting long-term retention and reprocessing.
- **Data Processing & Cleaning:** Employs Apache Spark for batch and stream-based cleaning, validation, and transformation.

- **Structured Relational Database:** PostgreSQL with PostGIS handles structured storage, time-series, and spatial queries.
- **Geospatial Layer:** Enables querying and visualization of georeferenced data using PostGIS.
- **BI Query Layer:** Composed of optimized indexes, materialized views, and partitions to support complex analytics.
- **Recommendation Engine:** Based on user preferences, stores suggestions dynamically.
- **Alert Management:** Thresholds and triggered events are logged and sent using a customizable notification system.
- **User Interface:** Developed with React and Leaflet.js, supports dynamic maps, charts, and interactive dashboards.

B. Database Design

The database schema follows a normalized structure to ensure efficient query performance and maintainability. It supports role-based access control, personalized recommendations, geospatial queries, and historical data tracking.

Fig. 2. Entity-Relationship Diagram of the Platform's Database

Key Entities and Relationships:

- **Station:** Contains metadata and location for monitoring stations.
- **AirQualityReading:** Stores timestamped pollutant readings.
- **Pollutant:** Catalog of pollutant types with measurement units.
- **User:** Registered user data including preferences and location.
- **Alert:** User-specific alert configurations and events.
- **Recommendation:** Personalized suggestions based on air quality levels.
- **Report:** BI reports generated by users, linked to selected parameters.
- **MapRegion:** Geospatial regions to support navigation and filtering.
- **DashboardConfig:** Stores user-specific visualization preferences.
- **ProductRecommendation:** Certified products recommended during pollution events.
- **Role, Permission, RolePermission:** Define and associate user roles and permissions.

C. Technologies Used

- **Backend:** Apache Kafka, Apache Spark, PostgreSQL, PostGIS
- **Frontend:** React, Leaflet.js
- **Storage:** MinIO (Data Lake)
- **Visualization:** Dynamic dashboards, interactive maps and charts
- **Deployment:** Cloud-compatible and horizontally scalable

III. Results

.

IV. Conclusions

.

References

[1] Dbdiagram.io. Holistics Software. (2024). *Design and visualize your database*. Available at: https://dbdiagram.io/

[2] Apache Kafka. Apache Software Foundation. (2024). *A Distributed Streaming Platform*. Available at: https://kafka.apache.org/

[3] PostGIS. Refractions Research. (2024). *Spatial and Geographic Objects for PostgreSQL*. Available at: https://postgis.net/

[4] AQICN API Documentation. World Air Quality Index Project. (2024). *Real-time Air Quality Index API*. Available at: https://aqicn.org/api/

Real-Time Air Quality Monitoring and Recommendation System: A Data-Driven Software Approach

Johan Esteban Castaño Martínez, Stivel Pinilla Puerta,
Systems Engineering Program,
Faculty of Engineering,
Francisco José de Caldas District University.

Introduction

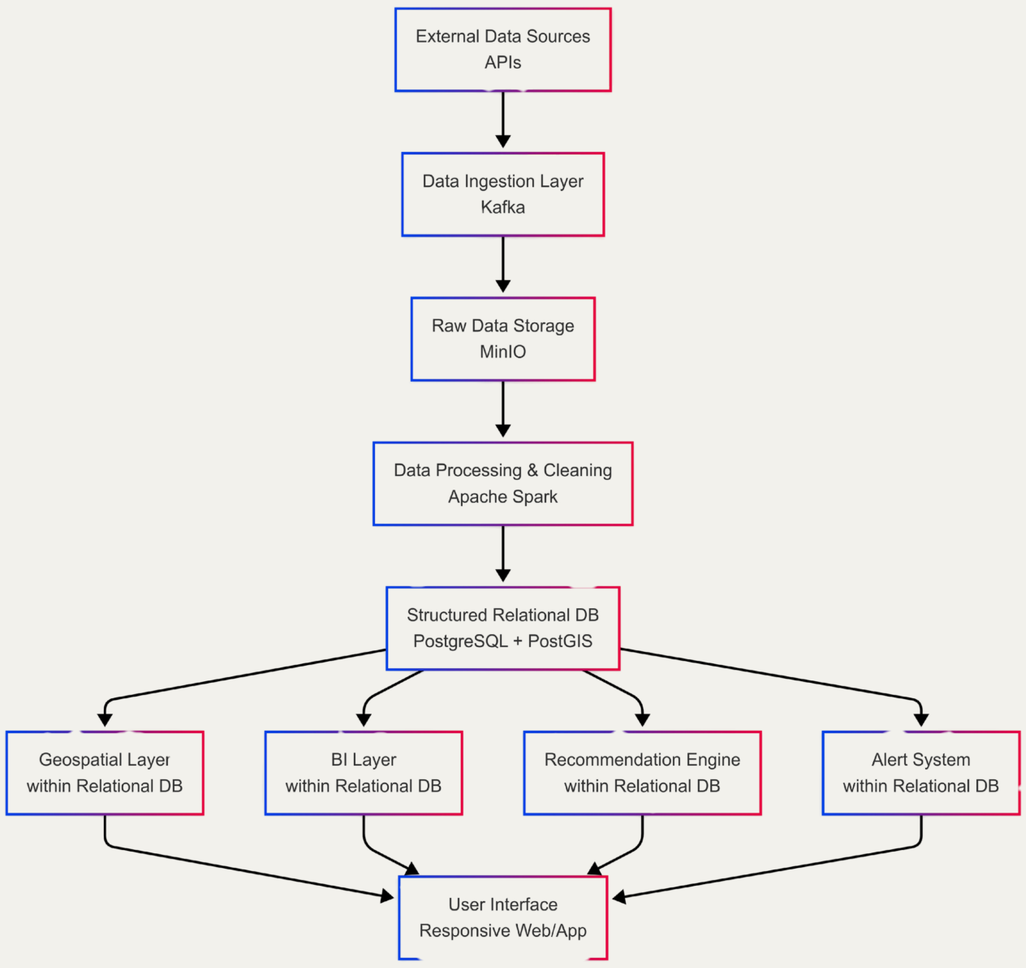
The development of database systems oriented to real applications is a key area in software engineering. This project explores the design and implementation of a distributed architecture for managing real-time data, combining relational and NoSQL databases. Traditional solutions face limitations in scalability, geospatial queries and result customization. The main challenge is to achieve an efficient integration that supports massive data ingestion, real-time queries and historical data storage, while maintaining performance and availability.

Goal

- Investigate how a hybrid, distributed and scalable database system can be designed for real-time and geospatial data applications.
- The final product is a working prototype with PostgreSQL/PostGIS and MongoDB, which allows ingest, analysis and visualization of environmental data.

Proposed Solution

The implemented system combines a microservices-based architecture with a double persistence layer: PostgreSQL + PostGIS for relational operations and geospatial queries, and MongoDB to handle semi-structured documents such as alerts and user preferences. Apache Kafka is used for real-time data ingestion from the AQICN API. The backend was developed in Go (Golang) and exposes a REST API for communication with the frontend. Geospatial indexes, normalized relationships and history mechanisms were designed for efficient queries.



Experiments

- Ingestion Performance:

Multiple queries to external APIs were simulated with readings every 10 minutes.

- Geospatial Queries:

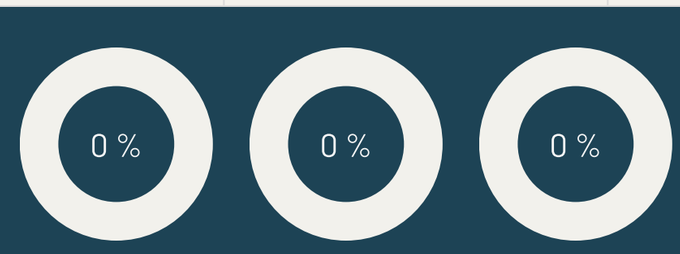
The average time to obtain readings near a specific location was evaluated.

- Historical Persistence:

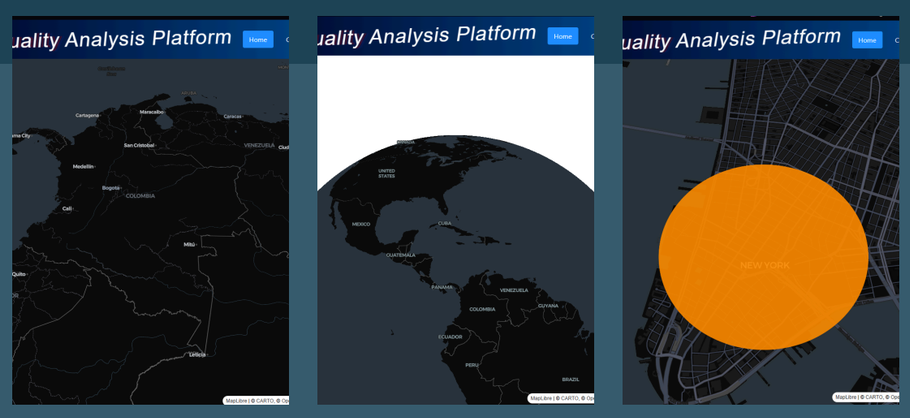
The efficiency of time series storage in PostgreSQL was measured.

Results

Experiment	Tools	Main Result
Real-time data ingestion	"Kafka, Go"	
Geospatial queries	PostgreSQL + PostGIS	
Historical storage	PostgreSQL Partitioning	



Conclusions



adsf

Bibliography

adsf

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS
Acreditación Institucional de Alta Calidad

AIR QUALITY ANALYSIS PLATFORM

Johan Esteban Castaño Martínez, Stivel Pinilla Puerta,
Systems Engineering Program,
Faculty of Engineering,
Francisco José de Caldas District University.

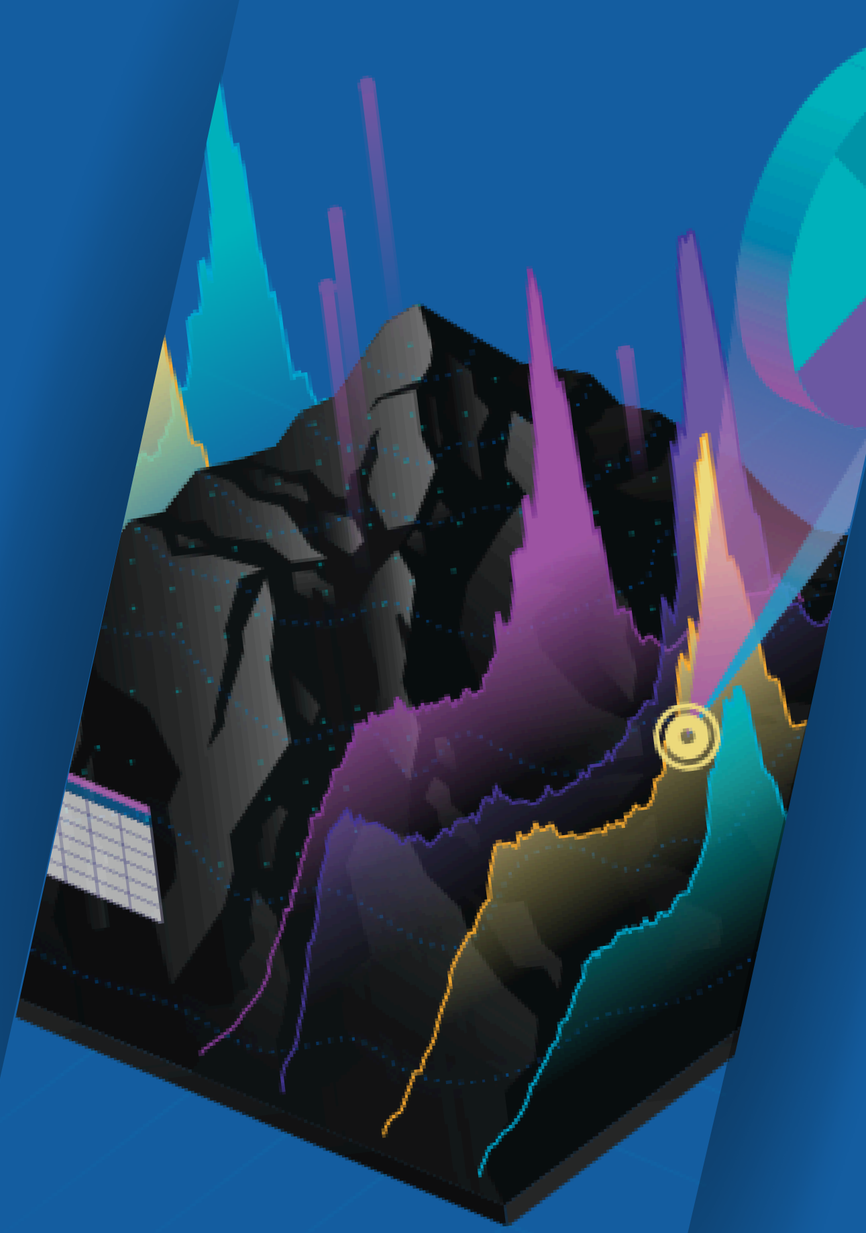
PROJECT SUMMARY

This project focuses on building a real-world database system using a distributed architecture.
It combines relational and NoSQL databases to handle real-time data efficiently.

Traditional solutions struggle with:

- Scalability
- Geospatial queries
- Custom results

Our main challenge:
Integrate data ingestion, live queries, and historical storage — without losing performance or availability.



PROBLEM & MOTIVATION

- Real-time air quality monitoring needs systems that are scalable, resilient, and support custom queries.
- Traditional monolithic databases:
  - Can't handle massive ingestion.
  - Struggle with geospatial filtering.
  - Are inflexible for user personalization.
- Goal: Build a modern data platform to address these gaps.

SYSTEM ARCHITECTURE

- Data Flow:
  - AQICN API → Kafka → Storage → Query Layer → Users
- Tech Stack:
  - Kafka: high-throughput streaming
  - PostgreSQL + PostGIS: relational & spatial data
  - MongoDB: user preferences & flexible documents
- Supports: real-time data, location-based search, custom alerts.

DATABASE DESIGN

Hybrid Relational Structure

The platform uses a normalized schema in PostgreSQL to manage structured and geospatial data efficiently.
Core entities include:

- Station: geographic monitoring points (with spatial support via PostGIS)
- AirQualityReading: timestamped measurements with pollutant metadata
- User-centric Tables: personalized alerts, dashboard config, reports & recommendations
- RBAC Support: Roles, Permissions and their mappings

Key Design Features

- Spatial Queries: MapRegion.geom enables proximity searches using GiST indexes
- Historical Tracking: time-series readings with indexed timestamps
- Personalization Layer: custom alerts, saved reports, and dynamic dashboards per user
- Security Layer: fine-grained access control using Role-Permission links

EXPERIMENTS & PERFORMANCE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis vulputate nulla at ante rhoncus, vel efficitur felis condimentum. Proin odio odio.

01 Cumplir con los plazos y el presupuesto

02 Lograr los resultados propuestos en el proyecto

03 Satisfacer las necesidades del cliente

04 Mejorar la eficiencia y efectividad

CONCLUSIONS & FUTURE WORK

GOAL ACHIEVED: REAL-TIME, GEOSPATIAL, AND PERSONALIZED AIR QUALITY PLATFORM.

NEXT STEPS:

- Integrate ML for predictions
- Add more cities
- Deploy full user dashboard