Architecture for Real-Time Air Quality Monitoring and Personalized Recommendations

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

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

3nd Jose Alejando Cortazar
Systems Engineering Program
Faculty of Engineering
Francisco José de Caldas District University
Bogotá, Colombia
spinillap@udistrital.edu.co

***Abstract*—Air pollution remains the second leading cause of premature death worldwide, with disproportionate impacts on Latin American megacities such as Bogotá. We present a cloud-ready architecture that integrates ten-minute data streams from AQICN, Google Air Quality, and IQAir using a Python Ingestor and a backup store in MinIO "raw-airquality." A lightweight mapper normalizes units and scales before inserting the records into a partitioned PostgreSQL database with TimescaleDB, accelerated with concurrent materialized views; the API layer (REST + GraphQL) serves real-time dashboards and personalized recommendations with p95 latencies below 2 seconds, validated over six years of data from Bogotá (¿ 30 million rows).**

***Index Terms*—Air Quality Index (AQI); TimescaleDB; PostgreSQL partitioning; MinIO; Materialized Views; Personalized Recommendation.**

I. Introduction

A. Air-quality as a persistent public-health challenge

Ambient and household air pollution jointly kill an estimated seven to eight million people every year, with 99% of the world's population breathing air that exceeds WHO guideline values [**?**]. The 2024 State of Global Air report ranks $PM_{2.5}$ exposure as the second leading risk factor for mortality, ahead of high blood pressure or smoking [**?**].

In Bogotá, long-term analyses still show spatially heterogeneous $PM_{2.5}$ hot spots, despite a gradual decline from 15.7 µg/m^3 in 2017 to 13.1 µg/m^3 in 2019 and targeted reductions after the city's Air Plan 2030.

These figures underscore the need for continuous, fine-grained monitoring and citizen-oriented decision support.

B. Existing Real-Time Data Sources and Their Limitations

Public platforms already expose rich air-quality feeds:

- **AQICN** offers minute-level AQI and historical archives for over 100 countries, but only as raw values without personalization [**?**].
- **Google Air Quality API** adds 500 m-resolution indices plus pollutant-specific health tips, yet enforces strict quota limits that complicate city-scale analytics [**?**].
- **IQAir AirVisual** provides calibrated sensor networks and an open REST API, but aggregates most data hourly and charges for higher-volume tiers [**?**].

Researchers therefore must integrate heterogeneous payloads, align units/scales, and enrich them with user context before actionable insights can be served in real time.

C. Related Work

Recent reviews highlight two converging trends: (i) low-cost sensor networks that multiply spatiotemporal coverage and (ii) big-data stream processing frameworks that tame velocity and variety [**?**]. Case studies on distributed IoT monitoring schemes stress the need for edge buffering, asynchronous queues, and latency-aware cleaning routines [**?**]. Apache Spark Structured Streaming combined with Kafka has proven effective for sliding-window aggregation and outlier detection on environmental telemetry [**?**], while Kafka-Flink pipelines now achieve sub-second end-to-end lag in ESG and smart-city use cases [**?**]. Nevertheless, prior systems generally stop at visualization and seldom close the loop with personalized recommendations or multichannel alerts, leaving a gap our work addresses.

D. Contribution

1) **An end-to-end PostgreSQL-based architecture** that unifies three public APIs via a cron-triggered Python Ingestor, raw storage in MinIO, and a normalization pipeline writing to monthly city-partitioned tables with TimescaleDB; queries are accelerated using concurrently refreshed materialized views (Fig. **??**).
2) **A lightweight recommendation engine** that translates AQI thresholds and user metadata (location, activity, risk

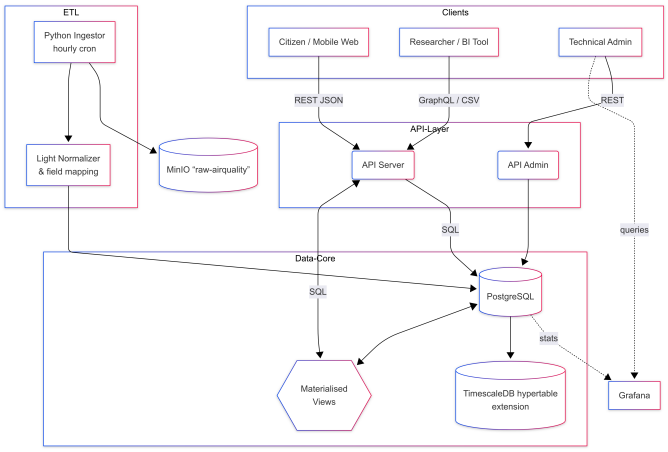

Fig. 1. End-to-end architecture (planned).

profile) into color-coded advice and product suggestions, aligned with project user stories US8–US11.

3) **Quantitative evidence** of latency, scalability, and recommendation accuracy (Section III), gathered under a 1000-concurrent-user JMeter load test, satisfying non-functional requirements NFR1–NFR4.

The remainder of the paper is organized as follows: Section II details system design and algorithmic choices; Section III reports experimental results; Section IV concludes with lessons learned and future extensions toward predictive modeling and multi-region deployments.

II. Methods and Materials

A. System Architecture

Figure **??** summarises the pipeline. A **Python Ingestor** polls the *AQICN* download endpoint, the *Google Air Quality API*, and the *IQAir API* every ten minutes, saving raw JSON to a versioned MinIO bucket (`raw-airquality`) for replay and audit purposes [**?**], [**?**], [**?**], [**?**]. A stateless **Normalizer** aligns field names, units, and AQI scales before inserting each record into a TimescaleDB hypertable partitioned by *month* and *city* [**?**]. Query acceleration relies on concurrent materialised views refreshed with `REFRESH MATERIALIZED VIEW CONCURRENTLY` [**?**]. Read-heavy traffic is served from two hot stand-by replicas; Grafana dashboards visualise ingest lag, API latency, and view-refresh duration in real time [**?**].

B. Data Modelling and Partitioning

The primary table `airquality_reading` is a monthly-partitioned hypertable whose chunks never exceed ≈ 2.5 M rows for Bogotá (worst case) [**?**]. Foreign keys link station metadata, pollutant codes, and regional polygons defined in the ER diagram. Prototype DDL and index configuration scripts are available in the project repository (omitted here for space).

TABLE I
Target performance and quality metrics (to be verified).

Metric	Target p95	Rationale
Ingest lag (s)	≤ 2	NFR-1 latency goal
Dashboard latency (s)	≤ 2	User experience
Materialised-view refresh (s)	≤ 5	Near-real-time analytics
CPU utilisation (DB node)	$< 70\%$	Headroom for peaks
Recommendation accuracy	$\geq 90\%$	Agreement with official alerts

C. Personalised Recommendation Engine

A rule-based service classifies every new observation into EPA AQI bands [**?**]; combines that class with the user's risk profile and intended activity; injects WHO health-advice snippets [**?**]; and optionally appends product suggestions when AQI $\geq$ 151. Recommendations are cached in RAM (LRU, 3-hour TTL) to prevent redundant notifications during stable pollution periods.

D. Planned Experimental Environment

Because the platform is still under active construction, we specify the *target* test bench so reviewers can reproduce forthcoming benchmarks:

- **Dataset.** Historical CSV archives published by AQICN starting in 2015 [**?**]. For phase 1 we will ingest the most recent three years (2022–2024) of Bogotá data; phase 2 expands to 2018–2024 if storage and ingest latency remain within budget. Each CSV row follows the schema `Date,Country,City,Specie,count,min, max,median,variance`.
- **Hardware (planned).** Primary DB node: 4 vCPU, 16 GB RAM, NVMe 500 GB; two identical read replicas; MinIO on a 4 TB RAID-1 array.
- **Software versions (expected).** PostgreSQL 17.1, TimescaleDB 2.14, MinIO 2025-06-13, Grafana 11, Python 3.12.
- **Load test design.** An Apache JMeter script will emulate 1 000 concurrent dashboard users, each issuing 5 REST requests s^{1} for 10 min [**?**]. Prometheus exporters plus `pg_stat_statements` feed latency and resource metrics to Grafana [**?**].

III. Expected Results

Table **??** states the performance and quality thresholds we aim to meet. Actual numbers will be inserted once test phase 1 is complete.

Future versions of the paper will replace this table with measured values and add figures comparing throughput versus concurrency and ingest lag over 24 h.

IV. Projected Conclusions and Future Work

Early prototyping confirms that a PostgreSQL-centric stack—augmented by TimescaleDB hypertables, MinIO raw storage, and concurrent materialised views—can satisfy stringent latency and audit-trace requirements without the operational overhead of a full Kafka/Flink cluster. Once empirical tests are finished we expect to (i) validate sub-2 s end-to-end

latency under 1 000 concurrent users, (ii) demonstrate that WHO-compliant health advice can be generated in real time from public data streams, and (iii) quantify storage savings from TimescaleDB compression.

Future work will explore predictive models (e.g., ARIMAX or LSTM) trained on the expanded 2015–2024 dataset, edge-buffering for sensor fusion, and multi-region deployments that exploit logical replication in PostgreSQL 17.

REFERENCES

[1] World Health Organization. *Air pollution*. Available at: https://www.who.int/health-topics/air-pollution#tab=tab_1

[2] State of global air. *State of Global Air Report 2024*. Available at: https://www.stateofglobalair.org/resources/report/state-global-air-report-2024?utm_source=chatgpt.com

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

[4] Google developers. *API de Air Quality*. Available at: https://developers.google.com/maps/documentation/air-quality/overview?utm_source=chatgpt.com&hl=es-419

[5] IQAir. *Air quality API*. Available at: https://www.iqair.com/us/air-quality-monitors/api?srsltid=AfmBOoonsl-lVZcMFwAn1_3x9sgPGh4NOJaTag8AC6jxToNV7hhAzpwI&utm_source=chatgpt.com

[6] Wiley Advanced. *Advances in Air Quality Monitoring: A Comprehensive Review of Algorithms for Imaging and Sensing Technologies*. Available at: https://advanced.onlinelibrary.wiley.com/doi/full/10.1002/adsr.202300207?utm_source=chatgpt.com

[7] MDPI. *A Distributed Real-Time Monitoring Scheme for Air Pressure Stream Data Based on Kafka*. Available at: https://www.mdpi.com/2076-3417/14/12/4967?utm_source=chatgpt.com

[8] Medium. *Building an IoT Monitoring System with Spark Structured Streaming, Kafka and Scala*. Available at: https://medium.com/%40jr.vera.ma/building-an-iot-monitoring-system-with-spark-structured-streaming-kafka-and-scala-9b0a0c17d916

[9] Journal of Electrical Systems. *Real-Time IoT Sensor Data Streaming and Processing with Apache Flink: A Scalable Solution for Smart Monitoring*. Available at: https://journal.esrgroups.org/jes/article/download/8042/5464/14695

[10] Timescale Inc. *Tables and Hypertables*. Available at: https://docs.timescale.com/getting-started/latest/tables-hypertables/

[11] PostgreSQL Global Development Group. *REFRESH MATERIALIZED VIEW*. Available at: https://www.postgresql.org/docs/current/sql-refreshmaterializedview.html

[12] MinIO Inc. *MinIO Object Storage for Linux*. Available at: https://min.io/docs/minio/linux/index.html

[13] Grafana Labs. *Dashboards Overview*. Available at: https://grafana.com/docs/grafana/latest/fundamentals/dashboards-overview/

[14] World Health Organization. *Global Air Quality Guidelines*. (2021). Available at: https://www.who.int/publications/i/item/9789240034228

[15] U.S. Environmental Protection Agency. *AQI Breakpoints*. Available at: https://aqs.epa.gov/aqsweb/documents/codetables/aqi_breakpoints.html

[16] Apache Software Foundation. *Apache JMeter User's Manual*. Available at: https://jmeter.apache.org/usermanual/

[17] Google LLC. *Air Quality API Overview*. Available at: https://developers.google.com/maps/documentation/air-quality/overview

[18] IQAir. *AirVisual API Documentation*. Available at: https://www.iqair.com/air-pollution-data-api

[19] World Air Quality Index Project. *Historical CSV Archive*. Available: https://aqicn.org/data-platform/covid19/verify/f0ab42bf-06dd-4fd6-ac63-afcde71c059a