**Big Data**

**Integantes:**

* CHAMBI MAMANI VLADIMIR
* ESCALERA MUÑOZ CRISTHIAN ANDRES
* MALDONADO CABALLERO ERICK
* CAMACHO BLANCO FABRICIO
* MARTINEZ DEL CASTILLO MILTON RAEL (Team Leader)
* PAREJA ALMENDRAS SAMUEL REYNALDO (proximo Team Leader)

**Links:**

[**GitHub**](https://github.com/MiltonMDelCastillo/Big-Data_Big_Grupo)

[**Trello**](https://trello.com/b/xrDsIi3a/bigdata)

Se usara datos proporcionados y privados por el Ing.

* Calidad de aire
* Sonido
* Soterrados

**Semana 1: 20–27 Octubre**

ETL inicial con Spark sobre datos CSV

Diseño del diagrama de arquitectura

Investigación de librerías para ingestión en tiempo real

Propuesta de almacenamiento SQL/NoSQL

Investigación de métricas clave para analítica de calidad de aire

**Semana 2: 27 Octubre – 03 Noviembre**

**Integrante 1 – ETL (Samuel)**

**Objetivo:** Leer y limpiar CSV (aire, sonido, soterrados). Automatizar la carga.

* **Entregables**
  + Script PySpark etl\_integrante1.py (lectura, limpieza, cast de tipos, dedupe, columnas derivadas).
  + Carpeta /salida\_limpia/\* con CSV limpios.
  + Ingesta automática: job programado (spark-submit o cron) que toma nuevos CSV de /dropbox/.
  + Mini‐reporte (1–2 páginas) con lo aplicado y evidencias.
* **Herramientas**: PySpark, Python, Git.
* **Criterios de aceptación**
  + 0 nulos críticos, tipos correctos, logs de ejecución y archivos generados reproducibles.
* **Semanas (cronograma):** **15–21 (X)**, **22–24 (X)**.

**Integrante 2 – Sistema en tiempo real (bidireccional)(Milton)**

**Objetivo:** Canal de comunicación tiempo real para panel/servicios.

* **Entregables**
  + Microservicio **WebSocket** (FastAPI+websockets o Node.js+socket.io).
  + Demo “eco bidireccional” y broadcast a clientes.
  + Guía de configuración y ejemplo de cliente web.
* **Herramientas**: FastAPI/Node, socket.io, Docker.
* **Criterios de aceptación**
  + Conexiones concurrentes (≥50), mensajes ida/vuelta <200 ms en LAN, reconexión automática.
* **Semanas:** **15–21 (X)**, **22–24 (X)**.

**Integrante 3 – Data Ingestion (cola de mensajes)**

**Objetivo:** Ingesta rápida/asíncrona y comparación **Kafka vs MQTT**.

* **Entregables**
  + **Pipeline A (Kafka)**: producer/consumer (erp.sales/gmc.sensors).
  + **Pipeline B (MQTT)**: publisher/subscriber (mosquitto).
  + **Cuadro comparativo** (latencia, throughput, QoS, orden, persistencia, complejidad, costo) y **elección justificada**.
* **Herramientas**: Kafka (Bitnami Docker), Mosquitto, Python (kafka-python/paho-mqtt).
* **Criterios de aceptación**
  + Métricas medidas con el generador (ver Int. 6): latencia p50/p95, msgs/s sostenidos ≥1 000 en local.
* **Semanas:** **15–21 (X)**, **22–24 (X)**.

**Integrante 4 – Storage (SQL + NoSQL)**

**Objetivo:** Repositorios destino configurables.

* **Entregables**
  + **PostgreSQL**: esquema relacional (mediciones, estaciones, variables, índices por (station\_id, ts)).
  + **MongoDB**: colección events con documento flexible (JSON de sensores).
  + Conectores desde ingestion y desde analytics.
  + Diagrama de datos + README de conexión (.env de ejemplo).
* **Herramientas**: Postgres, MongoDB, SQLAlchemy/PyMongo, Docker Compose.
* **Criterios de aceptación**
  + Escrituras ≥2 000 inserts/s (batch/ COPY en Postgres), índices correctos, consultas de ventana por rango de tiempo <1 s en dataset de prueba.
* **Semanas:** **15–21 (X)**, **22–24 (X)**.

**Integrante 5 – Analítica de datos (Spark)**

**Objetivo:** Métricas clave para aire/sonido/soterrados y funciones analíticas.

* **Entregables**
  + Jobs Spark: agregaciones por ventana (1 min, 5 min), detección de umbrales (alertas PM2.5, dB).
  + Feature store simple (tabla métricas por estación/intervalo).
  + Reporte técnico de variables y resultados (KPI: PM2.5 med/máx, dB med/máx, anomalías).
* **Herramientas**: Spark (batch/structured streaming), Python.
* **Criterios de aceptación**
  + Cálculos reproducibles sobre datos reales/simulados, código probado con muestras.
* **Semanas:** **15–21 (X)**, **22–24 (X)**, **27–31 (X)**.

**Integrante 6 – Visualización, Testing (generador) y Deployment**

**Objetivo:** Dashboards, generador de datos y empaquetado Docker.

* **Entregables**
  + **Dashboards**: Grafana/Superset o app web React (charts: series por estación, mapa, alertas).
  + **Generador de datos** (testing): script que emite JSON aleatorio a Kafka/MQTT (temas gmc.sensors.air|noise|underground), con control de rps y outliers.
  + **Dockerización**: docker-compose.yml (Kafka/Mosquitto, WS, ETL job, APIs, Postgres, Mongo, Grafana).
  + Script make up/down/logs.
* **Herramientas**: Grafana/React, Python generator, Docker/Compose.
* **Criterios de aceptación**
  + Dashboard refleja datos en ≤2 s desde publicación; docker compose up levanta todo el entorno.
* **Semanas:** **27–31 (X)**, **3–7 nov (X)**, **10–14 nov (X)**, **17 nov (Presentación X)**.

**Artefactos mínimos por frente (para calificar)**

* **Código:** repos etl/, realtime/, ingestion/, storage/, analytics/, viz/, con README y .env.example.
* **Docs:** 1 página por frente (objetivo, diseño, cómo correr, evidencias).
* **Arquitectura:** 1 diagrama (draw.io) con: *Producers → Broker (Kafka/MQTT) → Ingestion svc → Storage (Postgres/Mongo) → Analytics (Spark batch/stream) → WebSocket/API → Dashboards*.
* **Testing:** generador reproduce 3 variables (PM2.5, dB, nivel soterrado) con timestamps y station\_id; permite --rps, --burst, --qos.