Universidad Panamericana
Maestría en Ciencia de Datos
Datos Masivos

Proyecto Final: *Monitoreo Inteligente de Tráfico Marítimo con Big Data AIS*

Enrique Ulises Báez Gómez Tagle, Luis Alejandro Guillén Alvarez

24 de septiembre de 2025

Índice

1. Resumen ejecutivo

1.1. Introducción

El tráfico marítimo global genera millones de registros de posicionamiento a través del Sistema de Identificación Automática (AIS), que transmite automáticamente la identidad, posición, velocidad y otros datos de los buques. Originalmente implementado para mejorar la seguridad de la navegación, hoy es crucial para la gestión del tráfico marítimo, la conciencia del entorno y operaciones de búsqueda y rescate.

Este proyecto propone aprovechar Big Data y aprendizaje automático (ML) para analizar una base de datos masiva de mensajes AIS y ofrecer una solución innovadora a la pregunta: **¿Cómo detectar y entender comportamientos anómalos en el tráfico marítimo para mejorar la seguridad y la eficiencia?**

Responder esta pregunta implica analizar patrones normales de navegación y descubrir desviaciones significativas. En esencia:

- **¿Qué estamos haciendo?** Diseñamos un sistema de análisis que procesa grandes volúmenes de datos AIS para identificar anomalías en el comportamiento de los buques (posiciones fuera de lugar, velocidades inusuales, maniobras erráticas, etc.) y extraer patrones útiles sobre la operación de diferentes tipos de embarcaciones.
- **¿Para qué lo hacemos?** Para mejorar la seguridad marítima, la gestión del tráfico y la toma de decisiones, ofreciendo alertas tempranas de potenciales riesgos (colisiones, actividades ilícitas o fallos) y conocimiento profundo a autoridades y empresas navieras.
- **¿Cómo lo hacemos?** Empleando herramientas de Big Data de la Suite de Google Cloud (Storage + DataProc + BigQuery) para procesar datos geoespaciales masivos en en poco tiempo, complementado con modelos de ML que aprenden patrones habituales y detectan comportamientos que se apartan de lo normal.
- **¿A quién beneficia?** A organismos de seguridad marítima (marinas, guardacostas), a empresas navieras optimizando rutas y monitoreo de flotas, a aseguradoras evaluando riesgos operativos e incluso a investigadores del medio marino en estudios ecológicos.

Finalmente, los resultados se integraron en un *dashboard* interactivo con visualizaciones (mapas geográficos, gráficos de tendencias y rankings) que facilitan la interpretación y la toma de decisiones.

2. Visión general del desarrollo

2.1. Solución Actual / visión general

La solución actual se implementa de extremo a extremo sobre la nube de Google Cloud. El flujo completo es el siguiente:

- Una máquina virtual en Google Compute Engine se encarga de **scrapear los datos AIS** y cargarlos en Google Cloud Storage.
- En Cloud Storage se organizan dos capas diferenciadas: *raw* y *curated*, con prefijos de carpeta y particiones por mes (`YYYY-MM=`).
- Un clúster de Google Cloud DataProc ejecuta dos jobs:
  - Job **raw**: descomprime los archivos obtenidos y organiza los datos en particiones por año y mes, y finalmente los guarda en formato Parquet en Google Cloud Storage.
  - Job **curated**: lee la capa cruda, aplica transformaciones iniciales y genera la capa refinada lista para análisis.
- Una Cloud Function crea el **dataset y la tabla en BigQuery**, cargando la información desde la capa curated.

- En BigQuery se centraliza la explotación de datos mediante consultas SQL optimizadas, generando los datasets finales que alimentan las visualizaciones.
- El **dashboard de Streamlit**, desplegado en la misma VM, consume los resultados de BigQuery y presenta mapas, gráficos de tendencias y rankings de manera interactiva.

Toda la solución se ejecuta en la región `us-central1`. Para evitar duplicados en el flujo, se utiliza la clave compuesta **MMSI + timestamp**.

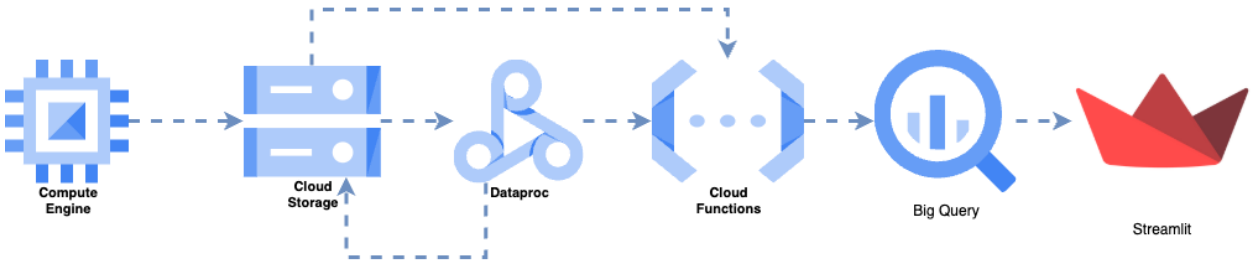

Figura 1: Arquitectura actual de la solución: flujo end-to-end en Google Cloud.

2.2. Limitaciones actuales de la solución

Actualmente la solución no está implementada de manera completamente *event-driven*. El disparo de los diferentes procesos (por ejemplo, la ejecución de la Cloud Function tras la escritura en la capa **curated**) no se encuentra automatizado mediante eventos de GCS o Pub/Sub, sino que requiere pasos manuales o ejecución programada. Esto representa una limitación en términos de eficiencia y escalabilidad. Como mejora futura se plantea migrar a un esquema **event-driven**, en el cual cada etapa del pipeline se active automáticamente al completarse la anterior, reduciendo la intervención manual y acelerando el flujo de datos de extremo a extremo.

2.3. Propósito, uso y alcance de la herramienta

3. Revisión y uso de datos

3.1. Orígenes y control de datos

Los datos utilizados provienen del portal oficial de la NOAA (`https://coast.noaa.gov/htdata/CMSP/AISDataHandler/2024/index.html`), el cual publica archivos comprimidos (.zip) con los mensajes AIS recolectados a lo largo del año 2024. Para garantizar la trazabilidad y el control de los datos, se desarrolló un script en Python (`scrapper.py`) que automatiza la descarga de todos los archivos disponibles, verificando duplicados mediante el tamaño de los ficheros y evitando descargas innecesarias. De esta manera se asegura que la totalidad de los datos originales publicados por la NOAA sean capturados sin alteración.

3.2. Preparación de datos

La preparación de los datos comienza en una máquina virtual de Google Compute Engine, donde se ejecuta el script de scraping. Este proceso descarga en paralelo (con *threads*) todos los archivos .zip hacia un directorio local (`ais_2024`) y posteriormente los transfiere a Google Cloud Storage en la capa **raw**.

Resumen de etapas (alto nivel). El pipeline de preparación contempla dos jobs en DataProc que operan sobre Cloud Storage:

1. **RAW_INGEST** (ingesta a *raw*): descomprime lotes `.zip` provenientes del scraping y los convierte a formato Parquet *particionado por mes* (columna derivada `ym`), añadiendo trazabilidad mínima (archivo fuente y timestamp de ingesta).
2. **CURATED_TRANSFORMATIONS** (hacia *curated*): toma la salida *raw* y aplica transformaciones básicas para dejar un esquema uniforme y listo para consumo analítico (normalizaciones de tipos, columnas temporales y desduplicación por `MMSI+BaseDateTime`).

Para mayor detalle, los detalles finos de limpieza, reglas y enriquecimientos revisar en la sección *Limpieza y tratamiento de datos*.

Automatización hacia BigQuery (Cloud Function). Tras la generación de la capa **curated** en Cloud Storage, una Cloud Function automatiza la creación/verificación de **datasets** y **tablas** en BigQuery. Su lógica principal es idempotente:

- Si la tabla ya existe, responde `action: none`.
- Si no existe y no se proporcionó esquema, intenta **autodetectarlo** cargando una muestra desde `gcs_uri`. Soporta formatos *Parquet/CSV/JSON/Avro/ORC* y versiones comprimidas (`.gz/.bz2/.zst`).
- Si se proporciona `schema`, crea la tabla con ese esquema explícito.

En la solución actual, esta función se integra al final del ETL, tras DataProc, para que la capa **curated** quede disponible en BigQuery lista para explotación y consumo por el dashboard.

3.3. Limpieza y tratamiento de datos

En esta sección se describen de manera detallada las transformaciones y criterios aplicados para convertir los datos AIS en una capa **curated** funcional para el análisis. Se listan los pasos con su *porqué* operativo o estadístico.

Normalizaciones y tipos.

- **Casteo y estandarización**: se convierten columnas numéricas a tipos adecuados (`double/int`) y se normalizan textos (`trim`, mayúsculas controladas) para evitar variabilidad en joins/agrupaciones.
- **Timestamps en UTC**: `BaseDateTime` se parsea a tipo timestamp con zona UTC para análisis temporales coherentes; sobre esta base se derivan `date`, `hour`, `week`, `month`, `quarter` y una columna `ym` (AAAA-MM) para particionado.

Coordenadas y límites físicos.

- **LAT/LON válidos**: se recortan latitudes al rango [-90, 90] y longitudes se *envuelven* a [-180, 180] (corrigiendo valores fuera de dominio por errores/reporte). Se descartan registros con coordenadas nulas tras la corrección.
- **Redondeo razonable**: se redondean LAT/LON a 5 decimales para homogenizar precisión sin perder resolución náutica relevante.

Velocidad, rumbos y estados.

- **SOG (Speed Over Ground)**: se acota a [0, 70] nudos para eliminar outliers físicos/telemetría corrupta; adicionalmente se calcula `SOG_ms` en m/s para modelos.
- **COG/Heading**: se normaliza módulo 360°. El valor `Heading=511` (no disponible) se marca como nulo para evitar sesgos en métricas de rumbo.
- **NavStatus**: se genera `NavStatusInt` a partir de `Status` y se une contra un catálogo para `NavStatusName`; códigos fuera de 0–15 se etiquetan como *Unknown* y faltantes como *Not reported*.

Dimensiones y calado.

- **Length/Width**: se acotan a rangos plausibles (1–450 m y 1–70 m, respectivamente) para descartar errores de orden de magnitud.
- **Draft**: se limita a [0, 25] m; valores negativos o excesivos suelen provenir de carga mal reportada o errores de parsing.

Identificadores y texto.

- **MMSI**: se extrae la porción numérica válida (1–9 dígitos) y se rellena a 9 con ceros a la izquierda para formato consistente.
- **IMO/CallSign/VesselName**: se normalizan espacios y caracteres no alfanuméricos; se limpian IMOs genéricos (p.ej., `IMO0000000`, `0`, vacío) para evitar falsas uniones.

Clasificación de tipo de buque.

- **VesselType**: se derivan `VesselTypeInt`, `VesselTypeName` (mediante mapa IMO) y `VesselTypeClass` (WIG, Cargo, Tanker, Passenger, etc.). Esto permite comparar métricas por clase homogénea y soportar reglas dependientes del tipo.

Enriquecimiento geoespacial.

- **Geohash9**: se calcula un `geohash9` distribuido (UDF con `pygeohash`) para agregación espacial eficiente y enlaces a capas geográficas (vallas, puertos, TSS) en análisis posteriores.

Calidad e idempotencia.

- **Desduplicación**: se eliminan duplicados por clave compuesta `(MMSI, BaseDateTime)` para asegurar series temporales limpias por embarcación.
- **Validación de conteos**: en *raw*, se comparan conteos diarios (CSV vs Parquet) y se aborta si no coinciden; en *curated*, se escriben marcadores de éxito por partición (`_markers/ym=.../_SUCCESS`) para reanudación y ejecución incremental sin sobrescribir trabajo correcto.

3.4. Integridad de los datos

3.5. Limitaciones de los datos

4. Proceso de desarrollo

4.1. Metodología

La metodología seguida para el proyecto se basa en un pipeline de datos con varias etapas bien definidas. En la primera etapa, una VM en Google Compute Engine ejecuta el módulo de scraping, encargado de descargar automáticamente los archivos AIS de la NOAA y trasladarlos a la nube (Cloud Storage). A partir de ahí, se desencadenan las siguientes fases: procesamiento con DataProc, almacenamiento en diferentes capas (raw y curated), carga a BigQuery y visualización con Streamlit. De esta manera, la metodología integra la automatización de la recolección de datos como un paso fundamental previo al procesamiento distribuido.

Proceso de desarrollo.

- **Orquestación y repetibilidad**: los jobs se ejecutan en DataProc con configuraciones de tolerancia a fallos (speculation, retries) y *idempotencia* (partition overwrite dinámico + marcadores). Esto permite relanzar sin duplicar datos ni romper particiones.
- **Particionado y costos**: todas las salidas en GCS se particionan por `ym` (AAAA-MM), lo que luego se refleja en BigQuery para escaneos selectivos y costos predecibles.
- **Versionamiento de dependencias**: se distribuyen "wheels"(p.ej., `pygeohash`) y un entorno reproducible (tarball de venv) a los ejecutores vía `spark.yarn.dist.archives`, garantizando que las UDFs geoespaciales funcionen de forma consistente.
- **Observabilidad**: se registran conteos globales y por día, así como tiempos de etapa, para detectar desviaciones (picos inusuales, lag IO). Los logs documentan causas de fallas y se retienen para auditoría.
- **Publicación a BigQuery**: una vez escrita la capa *curated*, la Cloud Function verifica/crea dataset y tabla en `us-central1` y, de ser necesario, autodetecta esquema a partir de una muestra en GCS, dejando la data lista para explotación por el dashboard Streamlit.

Diferencia conceptual: **Preparación de datos** resume *qué* pasos iniciales se ejecutan (ingesta y preprocesos), mientras que **Proceso de desarrollo / Metodología** documenta *cómo* se construye, ejecuta y asegura la calidad del pipeline (idempotencia, tolerancia a fallos, particionado, dependencias y publicación).

4.2. Pruebas

5. Resultados y conclusiones

6. Código utilizado

6.1. Link al repositorio con código fuente y salidas correspondientes

`https://github.com/enriquegomeztagle/MCD-BigData-SmartMaritimeTrafficMonitoring-FinalProject`