Hackathon LogView360

Grupo #1

Juan David Quiroga

Daniel Stiven Puerto

2025

Documento Técnico: Sistema de Ingesta, Integración y Detección de Anomalías en Logs Bancarios

1. Introducción

Este documento describe el funcionamiento técnico de un sistema de análisis de logs bancarios diseñado para un entorno de arquitectura distribuida en el sector financiero. El sistema tiene como objetivo unificar, normalizar y detectar inconsistencias en registros provenientes de tres fuentes principales: CoreBank, MidFlow ESB y SecuCheck.

2. Arquitectura del Sistema

El sistema está dividido en tres módulos principales:

- Módulo 1: Ingesta y Unificación (modulo1_ingesta.py)
- Módulo 2: Análisis Visual / Operacional (modulo2_grafos.py)
- Módulo 3: Lógica de Detección (modulo3_deteccion.py)
  Adicionalmente, se incluye un script de generación de datos (generador.py).

3. Módulo 1: Ingesta y Unificación

Este módulo se encarga de procesar los logs desde tres fuentes:

- CoreBank: un archivo .log que requiere un parser basado en expresiones regulares para extraer campos como timestamp, canal, usuario, IP, tipo de transacción, estado, etc.
- MidFlow ESB: un archivo .csv estructurado, requiere normalización de nombres de columnas y limpieza de datos.
- SecuCheck: archivo .json con datos jerárquicos, que se convierte a formato tabular usando json_normalize.

Los tres DataFrames resultantes se concatenan y ordenan cronológicamente por ID de transacción y timestamp. El archivo final se guarda como logs_unificados.csv.

4. Módulo 2: Análisis Visual y Operacional

Este módulo importa las funciones analíticas del módulo 3 y permite integración con bibliotecas como matplotlib para visualización de resultados. Aunque el código actual no implementa visualizaciones, está preparado para hacerlo.

5. Módulo 3: Detección de Anomalías

Este módulo carga los logs unificados y originales, y contiene funciones para detectar:

- Transacciones aprobadas no ejecutadas: cruza resultado_validación de SecuCheck con estado de CoreBank.

- Transacciones duplicadas o con latencia anormal: detecta duplicados por timestamp y valores anómalos de latency_ms comparando con media + std.

- Registros indeseables: errores o fallos en ESB relacionados con eventos en CoreBank.

6. Script de Generación (generador.py)

Se encarga de convertir el log plano de CoreBank a un archivo CSV usando la misma función parse_corebank_line que en el módulo 1, para reutilización de lógica de extracción.

7. Consideraciones de Normalización

Se estandarizan nombres de columnas y valores en minúscula sin espacios, para facilitar la comparación y análisis cruzado. Esta limpieza es crucial para evitar errores semánticos en el análisis.

8. Hallazgos y decisiones tomadas

- Decidimos csv como mejor opción para convertir el archivo CoreBank ya que se encontraba en un formato muy poco práctico.
- Se separó el código con el fin de tener mejor practicidad y comodidad, intentándolo hacer modular (POO).
- Detectamos que el modelo ML de Insolation Forest así también como la aplicación de un autoencoder serviría demasiado para la automatización de estos procesos y facilitar su desarrollo. Si se observa en un nivel escalar podría llevar a más problemas sin embargo este puede ser cambiado fácilmente y llevar al máximo su optimización
- La detección por latencia al principio se regía por una fórmula donde se sumaba dos veces la desviación estándar a la media sin embargo detectamos que dicho resultado no se ajustaba bien a los valores que se tenían planteados por tanto se decidió alterar la fórmula con el fin de llevar un resultado mejor y más acorde a la detección de anomalías por Latencia.

- En el CoreBank se puede detectar la limpieza de datos que tiene a cabo en cambio la “suciedad” entre los datos que tenemos se es notable aun más al principio de los procesos.
- Los problemas de latencia que se registran no son tan altos haciendo más difíciles de detectar los datos no anómalos.

Conclusiones

La experiencia de esta competencia fue muy enriquecedora para aprender más acerca del funcionamiento de los sistemas bancarios y sus tipos de detecciones en anomalías. Este sistema permite transformar datos heterogéneos en un repositorio coherente y estructurado, preparado para auditorías, detección de fraude, o integración con modelos de machine learning. La modularidad del código facilita su escalabilidad y adaptación a nuevas fuentes de datos o reglas de negocio.