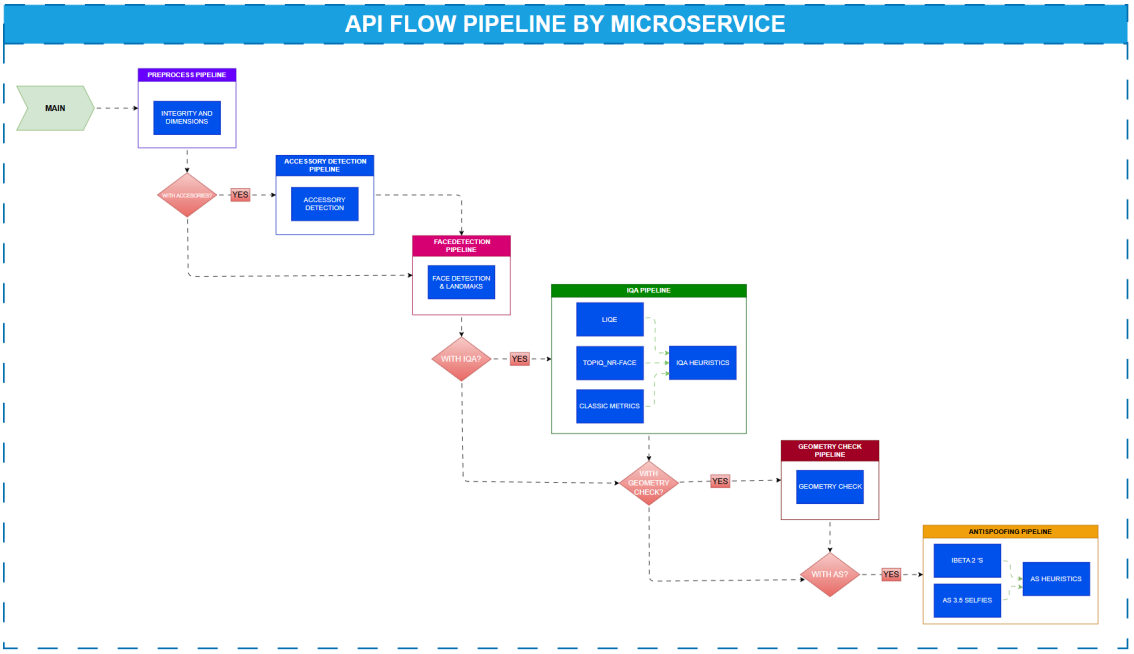
ETL Final Workshop - Manuel Henao - Github Repo

Problem Description: The project aims to analyze the performance of a facial validation API using an ETL process. The API, built on a microservices architecture, includes face detection, accessory detection, image quality assessment, and Presentation Attack Detection (PAD) to prevent identity spoofing. Performance data is stored in a non-relational database. The primary objectives are to identify execution time bottlenecks of each microservice, analyze acceptance rates per client, and improve user experience by examining retry patterns.

Context: The facial validation API is designed to ensure secure and efficient identity verification by capturing two moments: one where the user is near the camera and another where the user is farther away. This process, known as liveness detection, aims to enhance the performance of the PAD microservice. Each microservice will have execution metrics for both images. The API comprises several microservices:

- Face Detection: Identifies and locates faces in images.
- Accessory Detection: Detects accessories like glasses or masks that might obscure the face.
- Image Quality Assessment: Evaluates the quality of the image to ensure it meets required standards using various methods.
- Presentation Attack Detection (PAD): Detects attempts to spoof the system using photos, videos, or masks.



Workflow Diagram of Facial Validation API by microservice

The API's performance data is stored in a non-relational database using MongoDB, capturing metrics such as execution times, acceptance rates, and retry counts for two attempts. The pipeline workflow executes the microservices in a serial manner. For example, if the accessory detection microservice detects an accessory, it will terminate the transaction without processing the subsequent microservices.

Dataset Description: The dataset includes various performance metrics for each microservice. Below is a model card for the variables of interest (Staging Table):

Process

The ETL Workshop will be broken down into 3 main stages

1. **Extract Stage:** Involves the collection performance data from the API's non-relational database.
    a. Due to sensible data the raw data which is stored in a non-relational db in a json format some features were omitted and some others were anonymized because clients transactions behavior were involved. In order to achieve that as part of the ETL process the first stage called **000_anonymized.py** anonymizes and omits raw semi-structured data into a tabular format represented in a CSV file with the features mentioned in the model card.
    b. Following the ETL process the second stage is loading the data as part of the staging part that executed by **001_staging.ipynb** stages the raw data (after being anonymized) in a relational database.
2. **Transform Stage:** It covers the cleaning, transformation, and preprocessing of the data to ensure consistency, accuracy, and relevant insights. The transformation stage is executed by **002_eda_transform.ipynb**. This notebook covers the EDA of the staging table obtained and performs the transformation workflow to discover insights in the data that can describe API performance. It aims to help identify possible time bottlenecks of each microservice, analyze acceptance rates per client, and improve user experience by examining retry patterns. These transformations are described below:
    - **Transform Aspect Ratio**: Determines the aspect ratio of an image based on its width and height. It returns the following categories (‘SQUARE’, ‘PORTRAIT’, ‘LANDSCAPE’)
    - **Transform Compare Aspect Ratio**: This process compares the outputs of aspect ratio transformations and returns a binary category (True, False). Under normal API execution, this value is expected to be true. If false, it may indicate an anomaly with the input images used in the transaction, potentially signaling a presentation attack attempt.
    - **Transform Quality Image**: Determines the quality of an image based on its resolution. It returns the following categories ('8K', '4K', 'ULTRAHD', 'FULLHD', 'HD', 'XGA', 'SD', 'QVGA', 'LOWERSCALES')
    - **Transform Compare Quality Image**: This process compares the outputs of image quality transformations and returns a binary category (True, False). Under normal API execution, this value is expected to be true. If false, it may indicate an anomaly with the input images used in the transaction, potentially signaling a presentation attack attempt.
    - **Transform Date by Range**: This transformation converts a date column into multiple date-related columns. It returns the data divided by year, month, day, day of the week, hour, and minute. Additionally, it maps the month and day of the week into human-readable formats.
    - **Transform Missing Values related with Time**: Due to the sequential execution of the pipeline, some microservices may not be executed if an event is triggered in a previous microservice. For example, if the image contains accessories, the subsequent microservice will not be executed. To analyze the total execution time per microservice without affecting overall performance, rows with missing values will be filled with 0 seconds.
    - **Transform Missing Values related with metrics and probabilities**: Due to the sequential execution of the pipeline, some microservices may not be executed if an event is triggered in a previous microservice. For example, if the image contains accessories, the subsequent microservice will not be executed. The metrics and probabilities cover numerical scores with different ranges. In this case, filling a value with 0 could affect subsequent analysis to establish a possible image quality

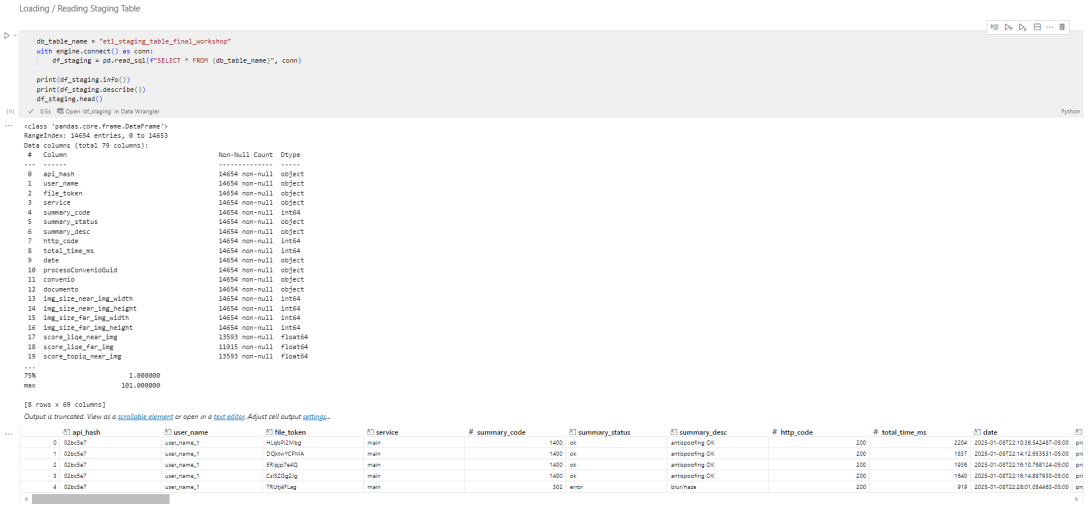
assessment threshold for accepting or denying a transaction. Therefore, missing values will be filled with -1 for columns related to probabilities and -2 for columns related to scores.

- **Transform Getting Total Time by Microservice:** Total time in ms and queue time per each microservice for near and far images are sum in order to get a total in ms by each microservice. As output the following columns are the result the original or individual time executions are dropped by this totals
    - 'total_time_ms_accessory_detector'
    - 'total_time_ms_face_detector'
    - 'total_time_ms_liqe'
    - 'total_time_ms_topiq'
    - 'total_time_ms_classic_metrics'
    - 'total_time_ms_as_35_selfies'
    - 'total_time_ms_ibeta2_crops'
    - 'total_time_ms_ibeta2_full'
    - 'total_time_ms_ibeta2_clip'
    - 'total_time_ms_geometry_check'
- **Transform Getting Overall Other Time Microservice:** Using the 'Transform Getting Total Time by Microservice' as input and the 'total_time_ms' column, the remaining time, which encompasses all execution times not returned by the API payload, is estimated. These times include, for example, transmission time between microservices, preprocessing times, image upload time to cloud storage, and time for insertion into the database.
- **Transform Getting Transactions without retries**: As part of the acceptance rates per client analysis, the column 'procesoConvenioGuid' represents the unique identifier of the transaction. Since each client has a different number of maximum retries for transactions per day, the final acceptance rate per client depends on the last response of each 'procesoConvenioGuid'. The final API response for each case depends on the columns 'summary_status' and 'summary_descr', which specify which microservice was triggered.
- **Dataset Description:** After the dataset transformations the model card for the variables is updated below (Transform Table):

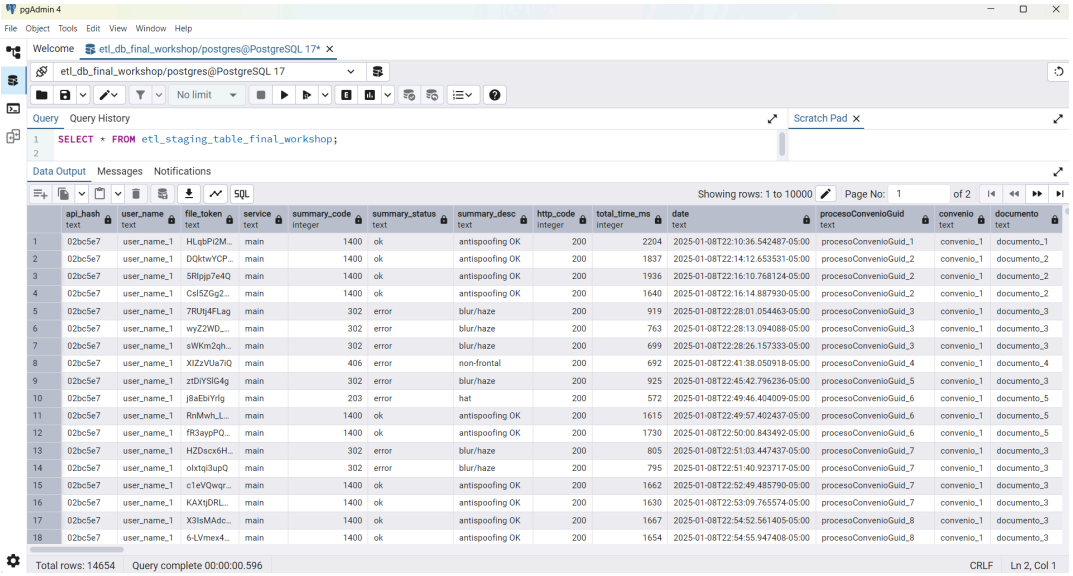
3. **Load Stage:** Load the transformed data into an analytical database for further analysis those DBs are etl_transform_table_final_workshop and etl_transform_table_final_workshop, both of them ready to use for visualization purpose , the first one more related to overall API performance considering transaction retries and the second one focuses on the acceptance rate per clients, due to the fact it considers only the last transaction.

Evidences

Extract Stage Evidence after executing **001_staging.ipynb**



pgAdmin4 query access



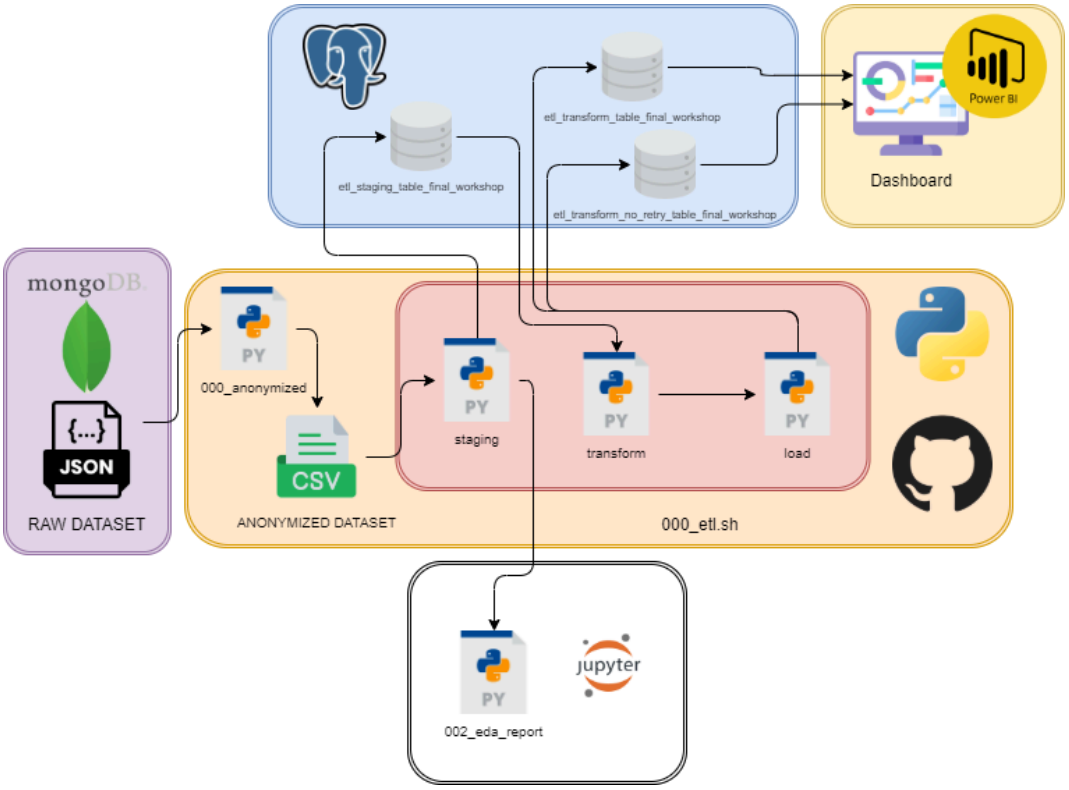
Transform Stage Evidence after executing **002_eda_transform.ipynb** also the EDA is available in this EDA report also executing **002_eda_report.ipynb** the EDA reports for:

- **EDA_etl_staging_table_final_workshop**
- **EDA_etl_transform_table_final_workshop**
- **EDA_etl_transform_no_retries_table_final_workshop**

ETL WORKFLOW

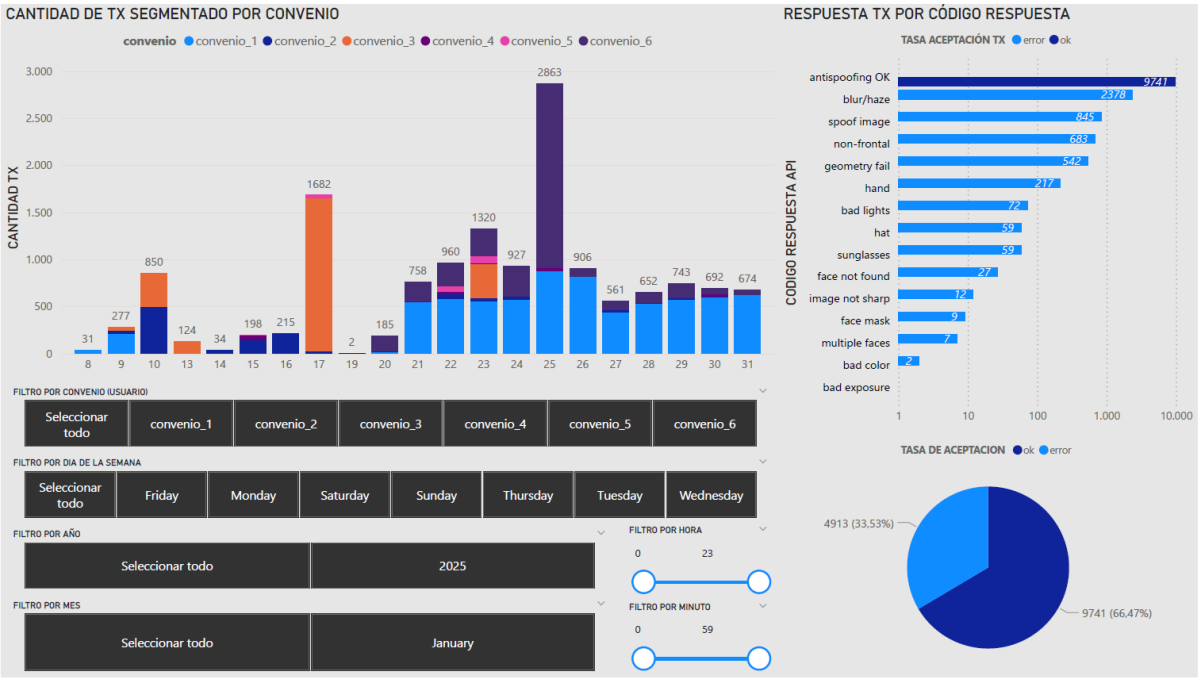
The ETL workflow is executed running the bash script 000_etl.sh it launches a python script 000_etl.py where all stages described before were automatically launched, it executes the staging, transform and load steps saving the output db ready to visualize are etl_transform_table_final_workshop and etl_transform_table_final_workshop

The



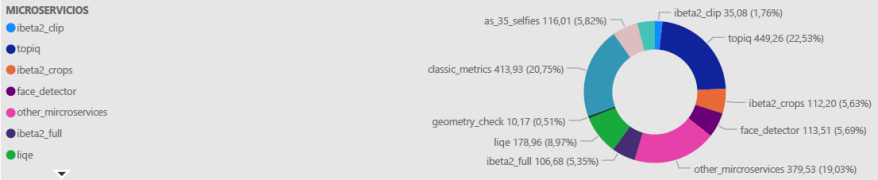
DASHBOARD TABS

USER CLIENT INSIGHTS TAB

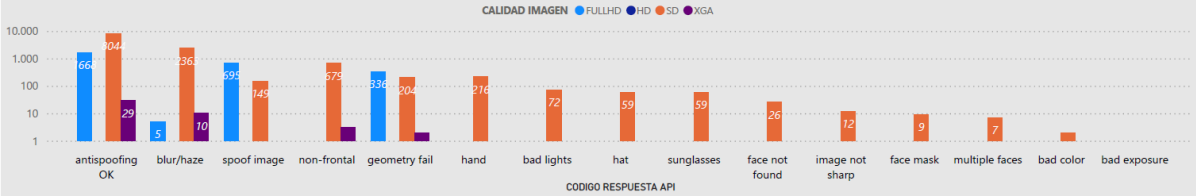


MICROSERVICE API TIME EXECUTION INSIGHTS TAB

PROMEDIO TIEMPO EJECUCION POR MICROSERVICIO



RESPUESTA TX POR CÓDIGO RESPUESTA SEGMENTADO POR CALIDAD DE IMAGEN



RESPUESTA TX POR CÓDIGO RESPUESTA SEGMENTADO POR RELACIÓN DE ASPECTO DE IMAGEN