Analytical Solutions for Performance Tracking in Automated Document Systems

Marta Correia

Internship Report

Master in Modelling, Data Analysis and Decision Support Systems

Supervised by:

PhD Bruno Veloso

2025

Acknowledgments

The completion of this work would not have been possible without the support and collaboration of several people and institutions, to whom I express my deepest gratitude.

First, I would like to thank my supervisor, Professor Bruno Veloso, for his dedicated guidance, constructive feedback and the trust he placed in me throughout the entire process.

To the DataColab association, I extend a special thanks for the opportunity to carry out my curricular internship and for providing access to the tools and data that were essential to the development of this study.

I am especially grateful to my mentors, Margarida Prozil and Mariana Silva, as well as the entire team, for their constant availability, knowledge sharing and constant support throughout this journey.

To my colleagues and friends, thank you for your encouraging words, insightful ideas and for the moments that made this journey lighter and more meaningful.

To the professors and the entire Faculty of the Master's, my sincere appreciation for the solid academic foundation and the valuable contribution to my personal and professional growth.

Finally, I wholeheartedly thank my family, especially my parents, for their unconditional love, constant support and for always believing in me, even in the most difficult moments.

To everyone who, in any way, contributed to the completion of this project: Thank you!

Abstract

The increasing use of digital documents underscores the growing need for tools that can extract and process information quickly and accurately.

In this context, this internship report, developed as part of a curricular internship at the DataColab association, aims to create a robust analytical framework for monitoring, analyzing, and evaluating the performance of the Docgenius software. This software is responsible for processing documents and images, specifically within the context of commercial invoices, by automatically extracting fields and recording them accordingly. The data generated throughout the process, from document submission to invoice processing and user modifications, is stored in a PostgreSQL database. As part of this project, this information was imported into Power BI, where data validation techniques, analytical dimension definitions and table relationships were applied, along with the creation of essential metrics for evaluating the software's performance. As a result, an interactive dashboard was developed, allowing the visualization of the system's current state and identifying potential improvement opportunities.

Additionally, a second dashboard was developed, dedicated to analyzing the software's backend logs stored in Elasticsearch. First, the logs are exported, processed, cleaned and stored in a PostgreSQL database and then directly connected to the Power BI dashboard. This integration is optimized and automated on a daily basis using Apache Airflow.

The main objective of this second dashboard is to identify operational patterns and detect potential anomalies, contributing to continuous monitoring and the optimization of the software's performance.

Beyond performance analysis, the project introduces an additional module, an automated pipeline for invoice validation and correction, combining rule-based and predictive mechanisms to reinforce data quality and reduce user intervention.

In this context, this project contributes to the continuous improvement of DocGenius by proposing an integrated approach to performance monitoring and analysis, combining interactive visualizations and log analysis. The results obtained highlight the potential of business intelligence and data engineering tools in developing effective solutions for tracking and optimizing complex document processing systems.

Keywords: document processing, performance metrics, analytics, logs, dashboards

Resumo

O crescente uso de documentos digitais destaca cada vez mais a necessidade de ferramentas que possam extrair e processar informações de forma rápida e precisa.

Neste contexto, este relatório de estágio, desenvolvido no âmbito de um estágio curricular na associação DataColab, tem como principal objetivo a criação de uma estrutura analítica robusta para monitorizar, analisar e avaliar o desempenho do *software* Docgenius, responsável pelo processamento de documentos e imagens, especificamente no contexto de faturas comerciais, através da extração automática de campos e do seu respetivo registo.

Os dados gerados ao longo do processo, desde a submissão de documentos até ao processamento das faturas e às alterações feitas pelos utilizadores, são armazenados numa base de dados PostgreSQL. No âmbito deste projeto, essas informações foram importadas para o Power BI, onde foram aplicadas técnicas de validação de dados, definição de dimensões analíticas e estabelecimento de relações entre tabelas, além da criação de métricas essenciais para a avaliação do desempenho do *software*. Como resultado, desenvolveu-se um *dashboard* interativo que permite visualizar o estado atual do sistema e identificar potenciais oportunidades de melhoria.

Adicionalmente, foi desenvolvido um segundo *dashboard* dedicado à análise dos logs de *backend* do *software*, armazenados no Elasticsearch. Primeiramente, os logs são exportados, processados, limpos e armazenados numa base de dados PostgreSQL, sendo posteriormente ligados diretamente ao *dashboard* do Power BI. Essa integração é realizada de forma otimizada e automatizada diariamente por meio do Apache Airflow.

O principal objetivo deste segundo *dashboard* é identificar padrões de funcionamento e detetar possíveis anomalias, contribuindo para a monitorização contínua e a otimização do desempenho do *software*.

Além da análise de desempenho, este projeto introduz um capitulo adicional, um pipeline automatizado para validação e correção de faturas, que junta mecanismos baseados em regras e técnicas preditivas para reforçar a qualidade dos dados e reduzir a intervenção do utilizador.

Neste sentido, este projeto contribui para a melhoria contínua do DocGenius ao propor uma abordagem integrada de monitorização e análise de performance, combinando visualizações interativas, análise de logs e mecanismos de correção automatizada. Os resultados obtidos evidenciam o potencial das ferramentas de *business intelligence* e engenharia de dados no desenvolvimento de soluções eficazes para o acompanhamento e optimização de sistemas complexos de processamento de documentos.

Palavras-chave: processamento de documentos, métricas de desempenho, análise, logs, dashboard

Contents

List of Figures

Chapter 1

Introduction

This chapter introduces the background and motivation that led to the development of this research, in the broader context of document automation and performance monitoring.

The chapter outlines the specific problem addressed by this project and presents the objectives pursued.

Finally, it describes the overall structure of the work, providing an overview of the different chapters and how each contributes to the proposed analytical framework. This serves as the foundation for the technical and methodological approaches explored in the following sections.

1.1 Context and Motivation

In the information age, with a paperless environment, the volume of data generated and stored grows exponentially, making manual processing unfeasible. Among these are digital documents such as PDFs and images containing structured or semi-structured information (Peña et al., 2024; Saxena & Parveen, 2018).

These documents are widely used in sectors such as healthcare, finance, legal and logistics, serving as the foundation for critical operations. However, manually extracting information from these documents is costly, error-prone and impractical on a large scale (Malashin et al., 2024). The increasing complexity and volume of transactions in modern business operations make the automation of invoice processing a critical necessity (Hassle & Bardvall, 2024).

In this context, solutions based on technologies such as Optical Character Recognition (OCR), Natural Language Processing and Large Language Models (LLMs) have emerged. These text recognition and detection techniques offer the promise of automating the reading and processing of such content (Hassle & Bardvall, 2024; Peña et al., 2024; Saxena & Parveen, 2018). Automated invoice processing must address diverse document features, including variations in format and layout, differences in language and terminology, as well as data inaccuracies. Techniques such as labeling and classification are crucial for isolating key invoice elements, especially given the lack of a global standard and the different ways legal information is presented across countries. Moreover, many companies lack a substantial corpus of well-labeled invoices for training or testing, making it essential to quickly adapt systems to new contexts and customize them for specific situations (Saout, Lardeux, & Saubion, 2024).

Despite the significant advancements of these technologies, many systems still face challenges that compromise their efficiency and usability. Automatic data extraction models often

struggle with variations in document formats, low image quality and the complexity of text structuring (Q. Zhang et al., 2024). A significant challenge is the efficient detection and extraction of tables from invoices, as they play a crucial role in presenting accounting information but can vary greatly in format (Saout et al., 2024).

Moreover, even high-performing systems cannot entirely eliminate the need for human intervention to validate and correct data, especially in domains where accuracy is critical (Inojosa et al., 2024).

This scenario is reflected in the case of DocGenius, despite its essential contribution to the digitization and recording of invoice information, there are still errors that require user correction. The absence of a monitoring mechanism within the software limits the identification and resolution of issues, hindering its evolution.

The motivation for this work arises from these limitations and the need for an analytical approach that goes beyond the system's basic operation. The creation of a dashboard with metrics such as success rates, error patterns and processing time will allow for the identification of improvement areas, making the automatic document reading software more reliable and efficient.

In addition to monitoring limitations, the absence of automated mechanisms to validate and correct extracted data further hinders the software's reliability. This challenge motivated the inclusion of an automated invoice validation and correction workflow, designed to proactively identify common data inconsistencies and reduce the manual workload. This module, supported by rule-based logic and predictive models, complements the analytical dashboards by closing the loop between monitoring, diagnosis and automated resolution.

Thus, the objective of this study is not only to understand the system's current performance but also to propose iterative improvements based on existing data.

1.2 Problem Definition

This project specifically focuses on creating an analytical solution to monitor the DocGenius software, an automated document and image reading system that processes commercial invoice content.

This software automatically processes documents and stores the extracted data in a "raw" table. Subsequently, users can review and correct the extracted information, if necessary, through a dedicated interface, with modifications recorded in a secondary table.

Currently, there is no integrated mechanism to assess performance metrics such as extraction success rates, document processing volumes, trend analysis or the frequency and nature of user corrections. Without these insights, it becomes difficult to identify weaknesses in the system, detect recurring issues or evaluate the impact of improvements over time.

To address these challenges, this project proposes a dual and complementary contribution.

Firstly, the development of two analytical dashboards in Power BI, one that enables a critical and detailed analysis of the software's status. This panel will include metrics such as success and failure rates, the number and types of corrections made and the most common error patterns. And another dashboard will integrate performance monitoring through log analysis retrieved from Elasticsearch, allowing the identification of average processing times, critical failures and other relevant metrics.

Secondly, and equally important, the project introduces an automated pipeline for invoice validation and correction, implemented with Python and Apache Airflow. This workflow applies rule-based mechanisms and predictive algorithms to detect common inconsistencies in the extracted data and automatically generate corrections or suggestions. By proactively addressing errors, this module enhances data quality and significantly reduces the need for manual user intervention, improving both reliability and scalability.

Together, these components reinforce the software's robustness and reliability through continuous analysis and proactive correction mechanisms.

This contribution supports the strategic goals of DataColab by enabling a deeper evaluation of the current system, facilitating error reduction and promoting more reliable and intelligent document automation workflows.

1.3 Project Structure

The structure of this project is organized into seven main chapters, each addressing a key component of the proposed analytical solution.

Following the introduction, the second chapter lays the theoretical framework by presenting the key concepts necessary to contextualize the study, covering essential techniques such as text mining, Natural Language Processing (NLP), Optical Character Recognition (OCR) and Large-Scale Language Models (LLMs), which form the technological foundation of the software.

The third chapter examines the existing literature in the field of Business Intelligence, with particular emphasis on Data Warehousing, ETL processes, OLTP/OLAP architectures and data visualization techniques. This review also highlights relevant research gaps, especially concerning performance monitoring within automated document processing systems.

Subsequently, the fourth chapter outlines the methodological approach adopted, structured according to the CRISP-DM framework. It encompasses the characterization, preparation and modeling of data derived from both invoice extraction and backend system logs.

The fifth chapter presents the design and implementation of interactive dashboards developed in Power BI. These dashboards facilitate performance monitoring, anomaly detection and the exploration of user interaction patterns, thereby enabling data-driven insights into system behavior.

In addition to the initially defined scope, a sixth chapter was proposed and developed independently. This chapter introduces an automated pipeline for invoice validation and correction, relying on rule-based mechanisms to identify and rectify common data inconsistencies. By integrating this corrective layer, the analytical framework is further strengthened, enhancing the overall robustness and reliability of the system.

The final chapter synthesizes the main findings of the study, reflects on its contributions and limitations and proposes directions for future research and potential system enhancements.

Chapter 2

Theoretical foundations

In this chapter, the principles of Text Mining, Natural Language Processing, Optical Character Recognition (OCR), and Large Language Models (LLMs) technologies will be discussed according to the existing literature. These concepts offer the theoretical bases necessary to understand the functioning of the software used, as well as the solutions proposed in this project.

2.1 Text Mining and Natural Language Processing

Text mining has emerged as a vital technique for the automatic extraction of information from textual data, whether structured or unstructured. Its applications span across various domains, encompassing tasks such as topic identification, keyword extraction and semantic and syntactic analysis (Feldman & Sanger, 2007; Yang, Kleissl, Gueymard, Pedro, & Coimbra, 2018). This approach leverages the integration of methods from data mining, machine learning and natural language processing (NLP), which have seen significant advancements over the past decades (Shamshiri, Ryu, & Park, 2024).

One of the defining aspects of text mining is its processing pipeline, which begins with text pre-processing. This step is critical for enhancing the effectiveness of subsequent analyses. (Mahdikhani & Meena, 2024) Today, this process must balance three sometimes-competing goals: noise suppression (e.g., handling OCR errors), semantic preservation, retaining morphology and domain terminology, and computational efficiency by limiting dimensionality (Göksel, Arslan, & Dinçer, 2023; Saraswati, Yanti, Muku, & Dewi, 2025). Processes such as tokenization, stemming, lemmatization and the removal of stop words are essential for the efficient preparation of textual data (Mahdikhani & Meena, 2024).

Tokenization involves classical methods, such as space-based division, which break the text into individual words. These methods are still popular (Feldman & Sanger, 2007), but there is a growing trend toward using subwords-based techniques (e.g., Byte Pair Encoding, WordPiece) that mitigate issues related to unknown words and improve performance in multilingual or specialized domains (Chai, 2023). Normalization, which standardizes case, accent marks and punctuation based on context, is also an important step. However, there is evidence suggesting that different tasks may require distinct approaches, and indiscriminate removal of punctuation could hinder analysis in certain specialized domains (Chai, 2023; Nesca, Katz, Leung, & Lix, 2022).

Regarding noise removal and irrelevant element elimination, robust strategies are avail-

able for handling typos, slang and special symbols. New methods, including those based on large language models (LLMs), are capable of detecting and removing noise contextually (Jalili, Tabrizchi, Mosavi, & Varkonyi-Koczy, 2024). While the use of fixed stopword lists remains common, recent studies advocate for dynamic approaches, such as exclusion based on frequency or structural text analysis, to eliminate irrelevant words without losing expressive value in the data (Oner, Hakli, & Zengul, 2024; L. Zhu & Luo, 2023). Stemming and lemmatization treat derived words as single entities, optimizing both semantic and syntactic analysis (Mahdikhani & Meena, 2024).

Recent scientific literature consistently emphasizes the critical role of preprocessing for the success of text analysis in NLP. Another fundamental component is feature extraction, which involves selecting and weighting relevant attributes and reducing dimensionality to simplify the feature space and improve model performance (Chen, 2024). Texts are then represented in computational formats such as structured vectors or distributed representations, enabling tasks like classification, topic modeling and information extraction (Y. Liu, Li, Deng, Hao, & Wang, 2024).

NLP, as a subfield of text mining, plays a crucial role by offering tools to understand and simulate human language. Its applications include analyzing relationships between words, text classification, sentiment extraction and generating semantic embeddings (Y. Liu et al., 2024; Vallelunga, Scarpino, Martinis, Luzza, & Zucco, 2024). With advancements in pre-trained models, such as transformers and contextualized embeddings, NLP has expanded its reach to specialized domains, where technical concepts and specific terms pose unique challenges (Y. Liu et al., 2024; Vallelunga et al., 2024).

Finally, text mining follows a structured methodology that involves defining objectives, collecting and organizing textual data, feature extraction, analysis and generating insights (Kwartler, 2017). These systematic steps are indispensable for ensuring reliable and meaningful results. The potential of text mining is evident in diverse fields such as weather forecasting (Yang et al., 2018), literature analysis (Chen, 2024) and innovations in supply chain management within the metaverse (Mahdikhani & Meena, 2024).

Overall, text mining and NLP not only facilitate the analysis of large volumes of text but also enable the extraction of valuable insights to support decision-making, research and development across various sectors.

2.2 Image recognition OCR

The development of computational technologies has significantly advanced the analysis and research of diverse types of information, particularly in the collection and processing of data. One area that has seen remarkable progress is computer vision, a field within artificial intelligence that aims to simulate human vision by acquiring, processing, analyzing and understanding images to perceive and recognize the external world (Chen, 2024). A core goal of computer vision is pixel-to-semantic conversion, which involves transforming raw image data into meaningful semantic representations.

Within computer vision, Optical Character Recognition (OCR) has emerged as a fundamental technology. OCR enables the digitization of text by converting printed or handwritten text into editable and searchable digital formats. This capability has become essential in an era characterized by the rapid expansion of digital information (Hamza, Ashraf, & Gomaa, 2024; Younis & Khateeb, 2017). The transformative power of OCR lies in its ability to bridge

the gap between analog and digital data, facilitating applications ranging from document digitization to advanced text analysis.

The OCR process involves several stages designed to convert images into text accurately. In the pre-processing stage, the quality of the image is enhanced, the digital area is positioned using processing algorithms and the region of interest is selected. Following this, the segmentation stage identifies and extracts relevant structures within the region of interest, tailored to the task's specific requirements. Feature extraction consists of obtaining feature vectors from the digital image and discretized characters, aiming to maximize differentiation between them. Finally, the pattern recognition stage analyzes the extracted vectors using pattern-matching algorithms, enabling the correct identification of characters and completing the image processing task (X. Zhang, Bai, & Cui, 2023).

Modern OCR systems leverage a range of advanced algorithms to enhance accuracy and performance. These include artificial neural networks, support vector machines (SVMs) and deep learning techniques. Each of these approaches presents unique challenges and opportunities, requiring careful analysis to optimize their application. Among these, deep learning, particularly Convolutional Neural Networks (CNNs), has shown exceptional performance in recognizing both handwritten and printed text (Saxena & Parveen, 2018; Younis & Khateeb, 2017).

One example of OCR's capabilities is its application in handwritten text recognition, which presents unique challenges due to variability in handwriting styles, orientations and character shapes. Deep learning-based approaches have shown promising results in addressing these challenges, especially in languages like Arabic, where characters can be more complex due to their cursive nature (Hamza et al., 2024; Younis & Khateeb, 2017).

OCR has been widely employed to digitize physical documents such as articles, books and official records, significantly improving work efficiency and enhancing capabilities for storing and managing text (X. Zhang et al., 2023). This technology serves as a crucial intersection between text and image analysis within computer vision, enabling powerful tools for transforming visual information into actionable digital data. Beyond its applications in text digitization, OCR plays a vital role in broader computer vision tasks, such as image classification, graphic segmentation and content recognition. These tasks enable systems to analyze external stimuli and replicate, or even surpass, human visual capabilities (Chen, 2024). For instance, computer vision technologies aim to compensate for limitations in human vision by applying advanced algorithms to understand and interpret the environment in innovative ways.

Despite its limitations, continuous advancements in artificial intelligence continue to enhance OCR technology, making it an indispensable tool across various fields. Researchers face ongoing challenges in designing systems capable of intuitively and accurately acquiring information from images and converting it into text, a task humans perform naturally (X. Zhang et al., 2023). These efforts keep OCR and image recognition leading the way, making it easier to connect the physical and digital worlds and supporting many different industries.

2.3 Large Language Models (LLMs)

Large Language Models (LLMs) are advanced neural networks with billions of parameters, trained on vast amounts of unlabeled textual data through self-supervised learning. This approach enables these models to comprehend and generate complex texts with contextual

accuracy (Alemifar, 2025; Mahadevkar, Patil, Kotecha, Soong, & Choudhury, 2024). Significant advancements in LLMs occurred with the introduction of the Transformer architecture by Google in 2017 (Alemifar, 2025).

The Transformer architecture serves as the foundation for LLMs, efficiently processing sequential data through the self-attention mechanism. This mechanism identifies global relationships between text elements, assigning different weights to words based on their context (Alemifar, 2025). The Transformer structure includes an encoder to process input and a decoder to generate output. In GPT models, the decoder functions independently, utilizing masked self-attention to predict word sequences (Alemifar, 2025).

LLMs differ from traditional programs because they learn patterns and nuances of human language from large datasets, enabling them to produce text that mimics human writing and conversation (Hassle & Bardvall, 2024). Notable examples of LLMs include OpenAI's GPT series, Meta's LLaMA and Google's Gemini. A fundamental characteristic of these models is the conversion of words into numerical vectors, allowing them to identify patterns and semantic similarities (Hassle & Bardvall, 2024).

The evolution of LLMs has been exponential, with the number of parameters increasing significantly across different GPT versions (Hassle & Bardvall, 2024). Additionally, LLMs have been applied across multiple modalities, including text-to-image, audio and video generation, establishing themselves as essential tools in generative AI (Mahadevkar et al., 2024).

LLMs have demonstrated great potential in processing unstructured documents, especially when combined with Optical Character Recognition (OCR) technologies. This process enables the conversion of physical or scanned documents into machine-readable text, facilitating the extraction of relevant information (Mahadevkar et al., 2024).

GPT Vision integrates LLMs with image and document recognition, making it useful for invoice processing. It analyzes images and generates textual responses based on extracted data. However, it has limitations, such as difficulties in reading low-contrast text. Adjustments to its interpretative capabilities can be made through detailed configurations (Hassle & Bardvall, 2024).

LLMs have been used to enhance document classification, data extraction and automated review. They can detect anomalies, suggest corrections and enrich documents through data integration. However, challenges such as algorithmic bias and high computational costs still pose obstacles to their implementation (Hassle & Bardvall, 2024).

Invoice processing involves challenges such as variable layouts, linguistic differences and data errors. To address these difficulties, labeling and typification techniques are employed to identify key invoice elements (Saout et al., 2024). The use of OCR for digitization is an essential approach, allowing the conversion of physical invoices into structured text, while NLP techniques are applied to extract relevant information (Saout et al., 2024).

The integration of LLMs with OCR and Computer Vision enhances the structural and semantic recognition of invoices, ensuring more accurate information extraction (Daqqah, 2024). Hybrid methods, such as combining Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and LLMs, have proven highly effective in recognizing handwritten text, leveraging context and visual patterns to improve transcription accuracy (Mahadevkar et al., 2024).

In summary, LLMs represent a revolution in natural language processing and document automation. Their ability to interpret context, generate coherent text and integrate with technologies such as OCR and Computer Vision makes them essential for the future of business

document analysis (Daqqah, 2024; Mahadevkar et al., 2024).

2.4 Summary of Key Concepts

This chapter provided an overview of the theoretical foundations that support the development of this project.

By exploring key concepts such as Text Mining, Natural Language Processing (NLP), Optical Character Recognition (OCR) and Large Language Models (LLMs), it established the technological and methodological background necessary to understand the challenges and opportunities in automated document processing.

Text mining and NLP techniques play a central role in extracting structured information from unstructured text, enabling more effective data analysis and decision-making. OCR technologies allow for the conversion of image-based documents into machine-readable formats, serving as a critical bridge between physical and digital data. Meanwhile, LLMs represent a significant advancement in the ability to interpret, generate and refine textual content within complex and variable document contexts.

Together, these technologies provide the foundational framework for the software architecture analyzed in this project, with a particular focus on automated invoice extraction and processing.

Chapter 3

Literature Review

The objective of this thesis is to analyze the software data extraction process within the framework of Business Intelligence (BI). This study employs a comprehensive BI approach to understand how different components contribute to the effectiveness of data analysis and decision-making. The following subsections provide an in-depth review of the literature on key elements of the BI process, including Data Warehouse (DW), ETL (Extract, Transform, Load) and Data Quality processes, OLAP (Online Analytical Processing) and OLTP (Online Transaction Processing) systems and Data Visualization. These topics are explored individually to highlight their roles, challenges and interdependencies in supporting robust analytical workflows and business intelligence strategies.

3.1 Data Warehouse

Data Warehouse (DW) was defined by Bill Inmon in 1990 as a subject-oriented and time-variant collection of data, created with the goal of supporting business decisions. In contrast to traditional relational databases that are optimized for transaction processing, data warehouses are purpose built to support complex querying and analytical tasks (Hashem, Rahouma, Abd, & Hameed, 2025). In this context, the DW acts as a central repository that allows companies to manage data collected from various sources, organizing it in a structured way and facilitating deep access and analysis to support data-driven decisions (Cormier et al., 2025).

According to Hashem et al. (2025), a complete DW environment includes three main components, a method for integrating various data sources through the Extract, Transform and Load (ETL) process, OLAP or OLTP engine, which facilitates multidimensional data analysis and analytical tools, which allow validation and visualization of data using Business Intelligence tools to create interactive dashboards and provide valuable insights to business users.

The design of a DW should be optimized for the efficient use of resources without compromising other parts of the system when adding new dimension tables or fields to the fact table (Cormier et al., 2025).

Moreover, the data schema used in a DW has a significant impact on query efficiency and data management. Hashem et al. (2025) defines three main types of schemas: Star Schema, Snowflake Schema and Galaxy Schema. Star Schema, characterized by a simple and intuitive design, where a central fact table is connected to several dimension tables. The dimension

tables store descriptive information about the involved entities, while the fact table contains quantitative data related to these entities (Cormier et al., 2025; Hashem et al., 2025). Snowflake Schema expands the Star Schema by further normalizing the dimension tables, splitting them into several related sub-tables. This reduces data redundancy and improves data integrity, but may result in more complex queries due to the increased number of joins required (Hashem et al., 2025). And finally, Galaxy Schema involves multiple fact tables sharing common dimension tables. This schema is ideal for complex multidimensional analysis as it links different business processes through shared dimensions (Hashem et al., 2025).

These different schemas offer advantages and disadvantages depending on the specific needs of each organization, being chosen according to the type of analysis and the volume of data to be processed.

The emergence of cloud data warehouses, such as Google BigQuery, Amazon Redshift and Snowflake, has introduced elastic compute resource scaling and serverless architectures (Armbrust, Ghodsi, Xin, & Zaharia, 2021). These systems decouple storage from processing, offering on-demand performance and reduced maintenance overhead. Such advances have positioned cloud warehouses as a standard in modern analytics pipelines, particularly for organizations that demand near real-time data refresh and simplified infrastructure management (Gulnar, 2023)

It is worth noting the growing interest in hybrid architectures like the data lakehouse. As described by (Armbrust et al., 2021), lakehouses enable unified access to structured and semi-structured data while maintaining schema enforcement and high performance. While lakehouses are particularly useful for handling diverse data sources such as images, audio and raw documents, which go beyond the scope of this study, their mention situates the chosen architecture within a broader technological landscape.

In summary, the data warehouse continues to serve as the backbone of Business Intelligence solutions. Its role in aggregating historical, structured data for efficient querying remains essential, even as architectures evolve to support real-time processing and more heterogeneous data formats.

3.2 ETL and Data Quality

Data Quality (DQ) plays a crucial role in Business Intelligence and Analytics processes and the effectiveness of Data Warehouses. The accuracy and relevance of business analyses depend directly on the high quality of data processed through ETL (Extraction, Transformation and Loading), a complex process subject to challenges related to data heterogeneity and quality issues (Georgiev & Valkanov, 2024; Souibgui, Atigui, Zammali, Cherfi, & Yahia, 2019; Wang et al., 2024).

In this context, recent literature has emphasized the role of quality verification embedded in ETL workflows, using validation rules and tracking mechanisms to detect anomalies in early stages (Souibgui et al., 2019)

Data quality can be categorized into four main dimensions: intrinsic (accuracy, reputation, credibility and provenance), contextual (quantity, relevance, completeness and timeliness), representational (comprehensibility, consistency and conciseness) and accessibility (ease of access and security) (Souibgui et al., 2019).

Each stage of the ETL process, extraction, transformation and loading, is prone to different quality issues, making it essential to track and correct defects throughout the data flow

(Georgiev & Valkanov, 2024).

To ensure high data quality, it is essential to adopt a structured approach, including specialized teams, data profiling and remediation strategies such as improving data quality at the source and automation to reduce input errors (Karnani, 2025). Common sources of poor data quality include human error, multichannel inconsistencies, multiple data sources and definition and validation issues. Additionally, customized quality mechanisms can optimize the integrity and accuracy of data within DWHs, ensuring reliability and analytical performance (Georgiev & Valkanov, 2024).

Data quality directly impacts decision-making, as poor-quality data can compromise business analyses and strategies (Wang et al., 2024). In the Big Data context, it is necessary to adapt quality dimensions and explore new approaches to ensure precise and relevant data. Factors such as management, operational costs, time and user experience also influence data quality.

The impact of data quality is significant, affecting end users and the reliability of data warehouses (Wang et al., 2024).

3.3 OLTP and OLAP

The distinction between OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing) is fundamental to understanding the different types of data processing in database systems. OLTP databases are designed for real-time transactional processing, whereas OLAP databases are optimized for complex analyses and multidimensional queries (Rose et al., 2024).

OLTP is widely used in systems that require constant updates and the handling of large volumes of short read and write transactions (Shioi et al., 2024). Real-time accuracy is essential in these systems, as any delay or error can compromise data integrity. Databases such as Microsoft SQL Server, MySQL, PostgreSQL and others use a row-oriented approach to optimize data insertion and updating (Asplin & Daneshtalab, 2025).

The need for integration between OLTP and OLAP is evident, as raw data stored in OLTP systems often undergo cleaning and transformation processes before being transferred to OLAP dimensions (Aslan & Çelebi, 2024). This ensures the quality of information used in managerial and strategic analyses.

OLAP databases are designed to facilitate complex queries and in-depth analyses. In contrast to OLTP, OLAP databases store information in a column-oriented manner, allowing more efficient access for aggregate calculations and historical analyses Asplin and Daneshtalab (2025).

OLAP technology enables the generation of multidimensional reports and interactive dashboards that help decision-makers identify trends and patterns. The integration of OLAP with Business Intelligence (BI) has been a critical factor in the evolution of decision support systems, enabling fast and accurate analyses (Aslan & Çelebi, 2024). The use of OLAP cubes is a common approach for consolidating large volumes of data into analytical dimensions, providing detailed insights into key performance indicators (Rose et al., 2024).

3.4 Data Visualization

Data visualization can be defined as the creation of graphical representations to display information (Unwin, 2020). These representations can range from a detailed display of each data point, such as in a scatter plot, to statistical summaries like histograms. The primary goal of visualization is to facilitate data interpretation and communication, enabling the identification of patterns, trends, outliers and other relationships (Healy, 2024). Additionally, charts are essential tools in exploratory data analysis (EDA), allowing analysts to understand data structures before applying more complex models (Unwin, 2020).

In the context of Business Intelligence (BI), a concept that refers to a set of processes, methodologies and technologies aimed at data collection, transformation, storage and analysis to support organizational decision-making, data visualization plays a crucial role. Key technologies in BI include Data Warehousing (DW), Online Analytical Processing (OLAP), data mining, ETL (Extract, Transform, Load) processes and other analytical approaches. Data visualization is an essential component of dashboards, which present relevant metrics and information on a single screen. The effectiveness of these dashboards depends on their ability to communicate visually, ensuring clarity, interactivity and alignment with organizational objectives (Khan, 2024). Well-designed dashboards also enable flexible analysis and support strategic decision-making.

The effectiveness of data visualization depends on the appropriate use of design principles and visual perception. Midway (2020) presents ten fundamental principles for creating effective visualizations. The first, prior planning, is essential for defining the core message before creating charts, ensuring that the visualization emphasizes the most relevant aspects of the data. Next, choosing the appropriate chart type is crucial, as different visual forms (bars, lines, points) represent data in distinct ways, making it essential to select the best geometry for each context. Another important principle is the strategic use of color to highlight information and avoid confusion. Additionally, accessibility should be considered, ensuring readability for color-blind readers. The inclusion of uncertainty is also key, representing data uncertainty through error bars, shaded intervals or box plots helps prevent misinterpretations. Another principle is the use of small multiples, where repeating a chart with variations in a specific variable can facilitate visual comparisons and reveal hidden patterns. Finally, differentiating between data and models is critical: exploratory charts can display raw data, statistical summaries, or model results and distinguishing these approaches is essential to avoid misconceptions.

Moreover, Healy (2024) emphasizes the influence of visual perception on graph interpretation. He highlights that different visual channels, such as position, length, angle and color, vary in effectiveness for representing ordered and unordered data. The application of Gestalt principles, such as proximity, similarity and connection, can improve the readability and organization of information in a visualization.

Dashboards represent one of the primary uses of data visualization in the organizational context. According to Khan (2024), an effective dashboard should present information on a single screen, use appropriate visualizations, offer interactivity (such as drill-down), organize elements clearly and highlight relevant metrics. The arrangement of elements should be guided by the user's workflow rather than purely aesthetic considerations.

Visual perception plays a crucial role in dashboard effectiveness. Pre-attentive attributes, such as color, shape, position and motion, can subconsciously direct users' attention, cre-

ating visual hierarchies and enhancing the analytical experience (Khan, 2024). Additionally, avoiding visual clutter ensures that information is quickly comprehensible (Unwin, 2020).

Unwin (2020) differentiates between exploratory and presentation graphics. Exploratory graphics are used during data analysis to uncover patterns and insights, often being interactive and flexible. Presentation graphics, on the other hand, are designed to communicate final results to a specific audience, requiring greater visual refinement and detailed explanations.

Exploratory charts should be simple and intuitive, while presentation charts need to be well-designed and accompanied by informative captions. Healy (2024) suggests that charts with some degree of embellishment may be more memorable, as long as they do not compromise information accuracy.

The correct choice of chart type, efficient organization of elements and use of well-designed dashboards significantly contribute to insight extraction and informed decision-making. As noted by Unwin (2020) and Midway (2020), data visualization practice requires not only technical knowledge but also a critical perspective to interpret and communicate information clearly and effectively.

3.5 Summary and Research Gaps

This literature review has explored the fundamental components that support Business Intelligence solutions, including Data Warehousing, ETL processes, OLTP/OLAP architectures and data visualization strategies. These elements form the fundamental pillars of analytical systems and have been extensively studied in various domains. However, despite the maturity of these individual areas, a few gaps remain when it comes to their practical integration and application in real-world systems focused on document automation.

Firstly, while data warehouses and ETL pipelines are well-documented, few studies address the combined use of structured data and system logs to monitor and optimize software performance. Most literature focuses either on user-facing data (such as processed document fields) or on technical backend logs, but rarely on their joint analysis to identify operational inefficiencies or user behavior patterns. This gap limits the capacity of organizations to obtain a complete view of software effectiveness and user interaction.

Secondly, the use of interactive dashboards for performance monitoring is generally concentrated on business metrics (e.g., sales, revenue, KPIs), with limited focus on software performance analytics in document processing systems. Specifically, dashboards that integrate error detection, correction rates and log-based event tracking are underexplored in academic literature, despite their potential to significantly enhance transparency and facilitate continuous improvement.

Thirdly, tools such as Power BI are widely applied in business reporting and Apache Airflow is a well-established tool for data pipeline orchestration. However, there is limited academic literature detailing their combined use in end-to-end automated reporting workflows, particularly within the scope of document processing and performance monitoring systems. Most available resources on this integration come from technical blogs or enterprise documentation, leaving a gap in formal case studies that explore their practical synergy in structured research contexts.

Finally, although data visualization is recognized as a powerful tool for insight generation, there is a lack of concrete case studies that illustrate end-to-end solutions, from data extraction to dashboard deployment, specifically in the context of invoice processing systems. The

present report addresses this limitation by proposing a dual dashboard architecture that not only evaluates extraction quality but also analyzes system logs to detect anomalies, delays and backend inefficiencies.

Despite the fact that invoice processing is a common use case in document automation, few academic studies explicitly address the automated evaluation and correction of extracted data. There is a noticeable gap regarding structured frameworks that combine extraction performance monitoring with corrective workflows capable of autonomously addressing common inconsistencies. This project contributes to filling this gap by proposing a pipeline that not only identifies recurring issues but also initiates rule-based and predictive corrections, thereby reducing the need for user intervention and improving overall reliability.

By identifying and responding to these gaps, this research contributes to both the academic field and practical implementations, offering a comprehensive and replicable framework for performance tracking in automated document systems.

Chapter 4

Methodology

This chapter begins with an overview of the CRISP-DM methodology, providing an in-depth explanation of its various phases, followed by a comprehensive description of the data used in the study. The tools and technologies that play a crucial role in the data processing, analysis and visualization are explored in detail in Appendix A.1, offering a deeper understanding of their application and significance within the context of the research.

CRISP-DM (Cross Industry Standard Process for Data Mining) model is widely recognized as one of the most used frameworks in machine learning and data mining projects. This open process model was developed to provide a neutral approach regarding industry, tools and applications, allowing specialists in the field to follow a structured guide for goal-oriented projects, even a midst technological transformations over recent decades (Schröer, Kruse, & Gómez, 2021; Zolbanin & Aubert, 2025). The systematic and iterative structure of CRISP-DM consists of six main phases, each playing a crucial role in the success of analytical projects.

4.1 Business Understanding

In the initial phase, Business Understanding, the focus is on clearly defining the problem or opportunity to be addressed. This step involves evaluating the business situation, understanding the available resources and translating them into specific and measurable objectives. This preparatory analysis is essential to align the project with organizational requirements and business goals (Gonçalves, Salgado, de Sousa, & Teixeira, 2025; Schröer et al., 2021). For instance, in the context of business intelligence solutions, this phase may involve identifying data dispersion as a problem and setting goals related to operational efficiency and decision-making (Gonçalves et al., 2025).

4.1.1 Software Monitoring via Logs

Log analysis is a widely recognized approach for monitoring the internal behavior of computing systems, particularly in distributed and high-complexity environments. According to (Cappelletti & Maglione, 2020), logs are files automatically generated by applications, containing records of executed actions and messages about the system's state. This data is essential for developers and analysts, as it allows for the reconstruction of events, detection of failures and a deeper understanding of system behavior over time.

The literature reinforces that log-based monitoring is a critical component of modern observability, complementing metrics and traces to provide a detailed and contextualized view of application performance (Bessa et al., 2024; Cappelletti & Maglione, 2020). As highlighted by Rigueira, Bernardino, and Pedrosa (2020), the logs generated daily by application servers contain valuable information, but their large volume and unstructured format make it difficult to extract useful insights directly.

Moreover, the variety of formats, heterogeneity of content and lack of standardization are cited as major challenges for effective log analysis (Bessa et al., 2024). Even specialized tools often face limitations when processing large-scale data, requiring customized parsing and transformation solutions tailored to the specific context of each application (Cappelletti & Maglione, 2020).

In the context of this project, log analysis is particularly relevant due to the absence of native monitoring mechanisms in the DocGenius software. As such, system logs represent the main source of data for reconstructing document workflows, identifying critical errors, measuring processing times and understanding backend behavior.

4.1.2 Software Monitoring via Invoice Extraction Data

From a business perspective, the need for software monitoring extends beyond analyzing system logs, as invoice extraction data itself represents a valuable source of insight into operational effectiveness and user experience. Structured data resulting directly from the extraction and validation process offers the company clear visibility into the accuracy and efficiency of automated document handling key metrics closely aligned with customer satisfaction and operational productivity.

Monitoring structured extraction data addresses important business questions, such as identifying recurring inaccuracies, quantifying the proportion of documents requiring manual corrections and pinpointing which document fields frequently generate errors. By systematically analyzing these elements, the business gains deeper knowledge about process inefficiencies, enabling proactive measures to reduce manual intervention and streamline validation efforts.

Furthermore, structured extraction data provides essential insights into user behavior patterns, highlighting common issues encountered during document processing. Understanding these patterns helps businesses optimize the automation process, prioritize improvements and develop targeted training for both users and extraction models. Ultimately, this contributes to reducing operational costs, enhancing user satisfaction and improving overall service quality.

In summary, integrating structured extraction data into the performance monitoring strategy directly supports critical business goals by providing measurable, actionable insights. This approach ensures continuous alignment between analytical outcomes and operational priorities, empowering decision-makers to optimize processes effectively and sustainably.

4.2 Data Understanding

Next, in the Data Understanding phase, information sources are collected and initially analyzed. This step includes assessing data quality by considering dimensions such as integrity, accuracy, consistency and accessibility. Tools such as visualizations, statistical analyses and interviews with experts play an important role in obtaining preliminary insights and guiding

the next steps of the process (Gonçalves et al., 2025; Schröer et al., 2021). Moreover, data quality is often regarded as a determining factor for the success of subsequent steps (Zolbanin & Aubert, 2025).

4.2.1 Invoice Extraction Data Characterization

The first stage of this project involves a detailed analysis of the software's performance based on data extracted from invoices. The structure of the initial database used for this analysis is presented in Appendix A.2.

To provide context, the software supports multiple users, each of whom can create several projects depending on how they wish to organize their invoices. Each project is associated with a specific client. Within a project, users can upload multiple documents, each of which may contain several invoices as well as other types of files. The purpose of the software is to identify which pages are invoices and extract relevant fields from them.

The database consists of four main tables. The *Projects Table* contains metadata related to each project, such as the project name, the user responsible for its creation, creation timestamps and a unique project identifier (ID). The *Documents Table* stores metadata about the uploaded files, including file names, number of pages, number of invoices, timestamps and document types. The *Invoices Table* includes structured and processed data extracted from the invoices. This data covers fields such as invoice number, issue date, total amount, VAT information and supplier or customer details. These fields can be edited by users through the system's interface. Lastly, the *Invoice Raw Table* stores unprocessed, raw data extracted directly from the invoices, serving as an immutable record to ensure data integrity and traceability.

4.2.2 Log Data Characterization

The second part of the project focuses on system log analysis. The software implements a comprehensive logging architecture that captures workflows and monitors overall system operations. This logging structure is illustrated in Appendix A.3.

The extracted log data includes a wide variety of event types, with the most relevant attributes being the timestamp, which indicates the date and time of the log entry; the logger name, which identifies the module or component responsible for generating the log; the document or request ID, which associates the log with a specific document or request. When the log refers to a document-related event, such as submission or processing, it is labeled as [doc_id] otherwise, it is labeled as [req_id] to allow proper tracking. In addition, the log level indicates the severity of the event, such as INFO, WARNING or ERROR and the description field provides detailed information about the recorded action.

Logs can be categorized into five main types. The first are extraction events, which document the fields extracted from invoices, such as invoice numbers, costs and other details. The second are validation steps, which log the verification of critical fields, including VAT numbers, postal codes and country codes. Third, database operations track SQL queries executed by the system, such as SELECT, INSERT and UPDATE. Fourth, file operations record events related to file handling, including uploads, image preprocessing and OCR execution. Lastly, logs related to processing times capture timestamps of key workflow stages, allowing for the analysis of processing efficiency and system performance.

This structured and detailed logging approach supports comprehensive analysis of document processing while ensuring full traceability and effective monitoring of system behavior.

4.3 Data Preparation

The Data Preparation phase is characterized by the transformations, cleaning and integrations necessary to adapt the data to the machine learning model to be applied. Typical tasks include error correction, replacement of missing values, removal of duplicates and the creation of derived variables, such as those employing regression and classification techniques (Gonçalves et al., 2025; Schröer et al., 2021). This is a highly intensive step but is fundamental to ensuring the quality and relevance of the data used in modeling.

In addition to classical data preparation tasks, such as handling missing values, removing duplicates and creating derived variables, the importance of structured data cleaning stands out as essential to ensure the quality and reliability of models and analyses. According to Guo et al. (2023), data cleaning is a critical process that involves identifying and correcting inconsistencies, errors and improper formats and is indispensable for the success of any real-world data-driven analysis.

In the context of Python-based analysis, data preparation becomes even more robust through libraries such as Pandas and NumPy, which allow for complex manipulations, type transformations, grouping, aggregations and the application of custom functions (Guo et al., 2023; Zervou, 2024). Strategies such as column renaming, logical filtering, text standardization, date parsing and data type transformations are widely adopted to ensure consistency and facilitate subsequent modeling steps (Gupta, 2021).

In the Power BI environment, data preparation is predominantly performed through Power Query, a visual data transformation tool that operates under a non-destructive, step-by-step logic. As described by Das, Banerjee, Nath, and Chatterjee (2023), Power Query enables tasks such as data type conversion, duplicate removal, column splitting, handling of missing values and data reshaping, making datasets more suitable for visual analysis and decision-making.

Moreover, Power Query facilitates connection to various data sources (e.g., SQL, Excel, Web APIs), allowing for pre-cleaning before the data is loaded into the model. The transformations performed are documented in sequential steps, which supports reproducibility and maintenance. Additionally, with the M language, it is possible to apply custom functions and perform advanced transformations that optimize the performance and scalability of the report (Becker & Gould, 2019).

In this regard, Collier (2024) and (Das et al., 2023) demonstrate that even public datasets require substantial transformations before they become suitable for analysis, including header promotion, removal of irrelevant columns, unpivoting tables, handling missing values and standardizing naming conventions. Power Query proved to be the central tool for performing these tasks, standing out due to its intuitive interface and integration with Office 365, which significantly reduces the learning curve for novice users, offering an embedded ETL (Extract, Transform, Load) interface (Collier, 2024).

This accessibility makes Power BI especially useful in business environments where data preparation must be carried out by analysts with limited programming experience, in contrast to the more technical and code-oriented approach required by Python.

Therefore, Python and Power BI offer complementary approaches to data preparation, while Python provides greater flexibility, automation capabilities and control for complex manipulations, Power BI through Power Query delivers a visual and user-friendly solution that democratizes the process of data transformation and real-time integration.

4.3.1 Invoice Data Processing Pipeline

The data used for monitoring the quality and performance of the invoice extraction process originates from four core tables Projects, Documents, Invoices and Invoices Raw, already described in detail in Section 4.2.1. These tables reflect the sequence of operations carried out during the document submission, processing and validation workflow. Each invoice is linked to a document via the document_id and both Invoices and Invoices Raw share the invoice_id as a common reference. These relationships enable comparisons between raw and validated data at the field level, supporting the measurement of extraction accuracy and the detection of systematic issues. It is important to note that the data is already processed, structured and organized by DataColab responsible for maintaining the system. This dataset is stored in a PostgreSQL database, from which it was accessed directly using Power BI. The availability of pre-processed data significantly reduced the need for extensive data cleaning, allowing the focus to shift towards metric development and analytical modeling.

Once connected to the PowerBI, the data was pre-processed using Power Query to ensure consistency and optimal performance. Key pre-processing steps included the verification and correction of column data types, particularly for dates, numeric fields and categorical values, as well as the splitting of timestamp fields into separate Date and Time columns to align with the temporal dimensions, among other transformations.

Together, these four tables provide a comprehensive view of the end-to-end extraction process, from document submission to final validation and form the foundation for the analytical work developed in this study.

4.3.2 Log Data Processing Pipeline

The performance monitoring conducted is based on log data generated by the backend services of the DocGenius system, which operates in a containerized environment orchestrated by Kubernetes. These logs are collected and indexed through the Elasticsearch engine and made accessible via the Kibana API, which provides filtered access to large volumes of log entries in near real-time.

Each log record corresponds to a discrete system event and is represented in semi-structured JSON format, containing metadata and textual descriptions of the actions performed. The log entries are automatically generated by different components of the application and follow a line-by-line format where each line reflects a single event. As described by Cappelletti and Maglione (2020), this type of architecture is common in distributed systems and enables detailed inspection of application behavior and failure reconstruction.

Among the various fields available in the raw log structure, the following are particularly relevant for this study as previously explained in Section 4.2.2.

timestamp The exact time the event occurred, used for ordering and temporal analysis.

module The internal application component responsible for generating the log.

level The severity classification of the log, including standard categories such as INFO, WARNING and ERROR.

message The main message describing the event, often including embedded key-value pairs.

identifiers Fields such as [req_id] and [doc_id], which reference specific requests and documents processed by the system.

This type of information allows analysts to reconstruct the flow of operations, trace document processing paths and detect abnormal behavior. The logs processed in this project are exclusively filtered by the tag field to include only those emitted by the DocGenius container, ensuring that only relevant system events are ingested.

Log Extraction and Storage Process

As shown in Figure 1, the process begins with the extraction of log data from the system via HTTP requests to Kibana's internal API, which provides access to the Elasticsearch index where the logs are stored.

A Python script was specifically developed for this purpose, using the requests library to perform POST requests with defined query parameters, including log tags, date ranges and pagination through the search_after mechanism. The use of search_after ensures that the complete dataset is retrieved in chronological order, avoiding duplication or data loss during pagination, an essential consideration when handling large volumes of log data.

Logs are extracted starting from the most recent date available in the database and are iteratively saved to a temporary file for further processing. Once extracted, the raw log data is stored in plain text files and passed to a transformation script. This script parses and converts the data into structured formats, as described in Section 4.3.2.

The transformed data is then written to a PostgreSQL database, chosen for its compatibility with analytical tools such as Power BI and its robustness in handling relational data models. Each resulting dataset was stored in separate tables to enable modular access and querying.

To ensure the reproducibility and automation of the entire workflow, the pipeline was integrated into an Apache Airflow DAG, running in Docker. Airflow manages the execution order of the tasks, beginning with log extraction, followed by parsing and database storage. A fixed schedule was defined to enable regular updates of performance data.

This process reflects what Rigueira et al. (2020) describe as a practical and scalable method for handling large log datasets, combining custom Python scripts with structured storage to facilitate visualization and KPI generation.

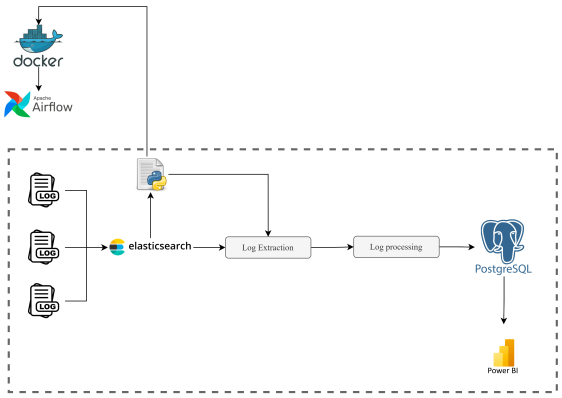


Figure 1: Overview of the log processing pipeline

Log Parsing and Structuring with Python

After extracting the raw logs from the Elasticsearch interface, the data undergo a crucial stage of cleaning and transformation in order to be structured and made suitable for subsequent analysis.

Given that the logs are in a semi-structured format, containing information embedded in free-text strings and following multiple message patterns, a key step in enabling analytical processing is their transformation into structured, machine-readable data. This phase, known as log parsing, is widely recognized in the literature as a foundational step in log mining, facilitating the extraction of operational insights and anomaly detection (Bessa et al., 2024; Cappelletti & Maglione, 2020).

While several authors recommend the use of specialized tools and libraries such as Drain, Logstash or Splunk for parsing and structuring logs (Cappelletti & Maglione, 2020), recent studies emphasize that no single parser performs well across all types of logs due to the high variability in log structures (J. Zhu et al., 2019).

In scenarios involving highly specific or non-standardized formats, a manual parsing approach using Python offers greater flexibility and interpretability. Similar approaches have been adopted by J. Zhu et al. (2019) and He, Zhu, He, and Lyu (2016), who demonstrate the viability of regex-based parsing strategies for extracting structured data from application logs in controlled environments.

Regular expressions (regex) remain a widely used technique for pattern recognition and are at the core of many traditional log parsing solutions.

According to J. Zhu et al. (2019), although handcrafted regex becomes difficult to maintain at scale, it remains a direct and interpretable method for extracting key parameters in smaller or domain-specific contexts. Furthermore, for targeted log analysis tasks in environments with moderate volumes or highly specific requirements, regular expressions provide an efficient, customizable and transparent solution, qualities consistently reinforced in the log mining literature (S. Liu, Yun, Nie, Zhang, & Li, 2024; J. Zhu et al., 2019).

In this study, after discussing the available solutions with the business management team, a manual approach was adopted.

The complete implementation of this process is presented in the figure in Appendix A.6.

The GitHub repository (logs-monitoring-pipeline) contains the full source code, structured into several key components that reflect the modular organization of the pipeline. The first module, docgenius_dag, defines the Airflow Directed Acyclic Graph (DAG) responsible for orchestrating two main tasks that rely on the core functions from both the Kibana and log processing modules. The general_utils module contains utility functions, primarily focused on managing the connection to the database.

The kibana_main module encapsulates the main functionalities required to interact with the Kibana API, including the definition of date ranges, the construction of search queries with all necessary parameters and the extraction of logs based on those queries. Once the logs are retrieved, they are processed by the logs_main module, which is responsible for the core logic used to clean, transform and structure the raw log data into a format suitable for analysis.

Supporting this process, the logs_utils module provides auxiliary functions that assist the main log processing tasks. In addition, there are two specialized utility modules designed to handle the twi specific categories of logs, log_doc_utils, which contains functions for processing logs related to document operations and log_req_utils, which deals with logs corre-

sponding to HTTP requests.

Finally, the email_utils module is responsible for formatting and sending email notifications. The mail module, ensures that the message content is clear and informative, allowing stakeholders to be promptly notified in case any errors occur during the execution of the pipeline.

In this approach, the logs are parsed using a regular expression to extract the main components, "timestamp", "module", "log level", "identifier" (either 'req' or 'doc') and the message body. This parsing step enables the segmentation of the logs into two separate dataframes, req_logs containing HTTP request logs and doc_logs containing document processing logs.

From the msg field, additional parsing is performed, splitting the text using the delimiter "|" and converting each fragment into named columns. Specific fields such as "duration_s", "status_code", "error_detail", "login" and "ip" are also extracted when present. URL strings are generalized using placeholders (e.g., {project_id}, {invoice_id}) to facilitate aggregated analysis. An example of this pre-processing step is shown in Appendix A.4.

Additionally, columns containing null or irrelevant values are discarded to reduce noise and inconsistencies. The entire pipeline is fully automated in Python. An example of this pre-processing step on de document logs is shown in Appendix A.5.

These transformations were essential to standardize the data and enable the subsequent construction of metrics and visualizations.

Creation of Analytical Fact Tables

From the structured logs, several metrics were derived through grouping and filtering operations, resulting in fact tables optimized for analysis in Power BI.

This approach was designed because a single request can generate multiple log entries and the relevant information may be distributed across all of them. Therefore, the log data was consolidated according to specific analytical objectives. The construction of these tables follows the logic described below:

- **Request Errors (req_errors):** Filtering log entries with a log level different from INFO (i.e., entries with ERROR or WARNING). Then, for each specific req_id, the code retrieves the corresponding timestamp, status_code and error message.

- **Document Processing Errors (doc_errors):** Similar to the above, but applied to logs identified by doc_id. When available, the fields error_message or error_detail are used to describe the error.

- **Logins (req_logins):** Constructed from requests containing login information, capturing request duration, status_code and user ip. Any associated error messages are also linked to the corresponding login event.

- **Document Duration (doc_logs_duration):** Derived from unique document log messages where the duration field is present. Each occurrence is recorded with the corresponding timestamp and doc_id.

- **Invoice Processing Time (doc_time):** Logs containing the message *"Process invoice finished"* are used to calculate the total processing time per document, also associating the number of invoices created, when available.

- **Request Path Duration (req_path_duration):** For each request, metrics such as method, generalized path, duration, status_code and timestamp are computed, allowing for aggregated analysis by endpoint.

These metrics are automatically exported to a PostgreSQL database and consumed by PowerBI, where they are visualized in an interactive and aggregated manner.

4.4 Modeling

In the Modeling phase, CRISP-DM emphasizes the selection and application of modeling techniques suited to the identified problem. The selection of algorithms and parameters is guided by an understanding of the data characteristics and the previously defined business objectives. Modeling also involves rigorous model validation through criteria such as accuracy, recall and derived metrics (Schröer et al., 2021). Studies also highlight the use of multiple models to compare performance, ensuring the robustness of results (Zolbanin & Aubert, 2025).

4.4.1 Dimensional Model for Analysis of Invoices Data Extraction

Temporal Dimensions and Data Modeling

To enable flexible temporal analysis and support the development of time-based metrics, two separate temporal dimensions, Date and Time, were created and integrated into the data model. As in the log analysis, these dimensions allow document and invoice events to be analyzed across various levels of granularity, including daily, monthly and hourly perspectives.

The decision to split timestamps into two distinct dimensions was driven by analytical and performance considerations. The Date table includes attributes such as Day, Month, Year, Week Number and Month Name, which facilitate temporal grouping and trend analysis. Meanwhile, the Time table captures the Hour, Minute and Second, allowing for the identification of usage peaks and patterns throughout the day.

These temporal dimensions proved particularly valuable for evaluating document processing volume over time, monitoring system activity patterns and assessing the evolution of extraction quality. For example, they made it possible to compare error rates across different months, identify processing peaks by hour or track improvements in extraction performance across projects and time periods.

Power BI Implementation and Analytical Logic

The analytical process was implemented in Power BI, leveraging its integration capabilities with PostgreSQL to directly access the structured invoice extraction data provided. The PostgreSQL connector was used to establish a connection, allowing the dataset to be refreshed automatically as updates occurred in the database.

Following the data preparation, a series of DAX measures were developed to support analytical monitoring. These included metrics such as the daily variation in submitted documents and invoices, the average number of changes per invoice and per field, the error rate per document, the total number of documents, invoices, changes and several others.

The combination of pre-processing, modeling and analytical logic provided a robust foundation for data visualization.

Schema Summary and Design Justification

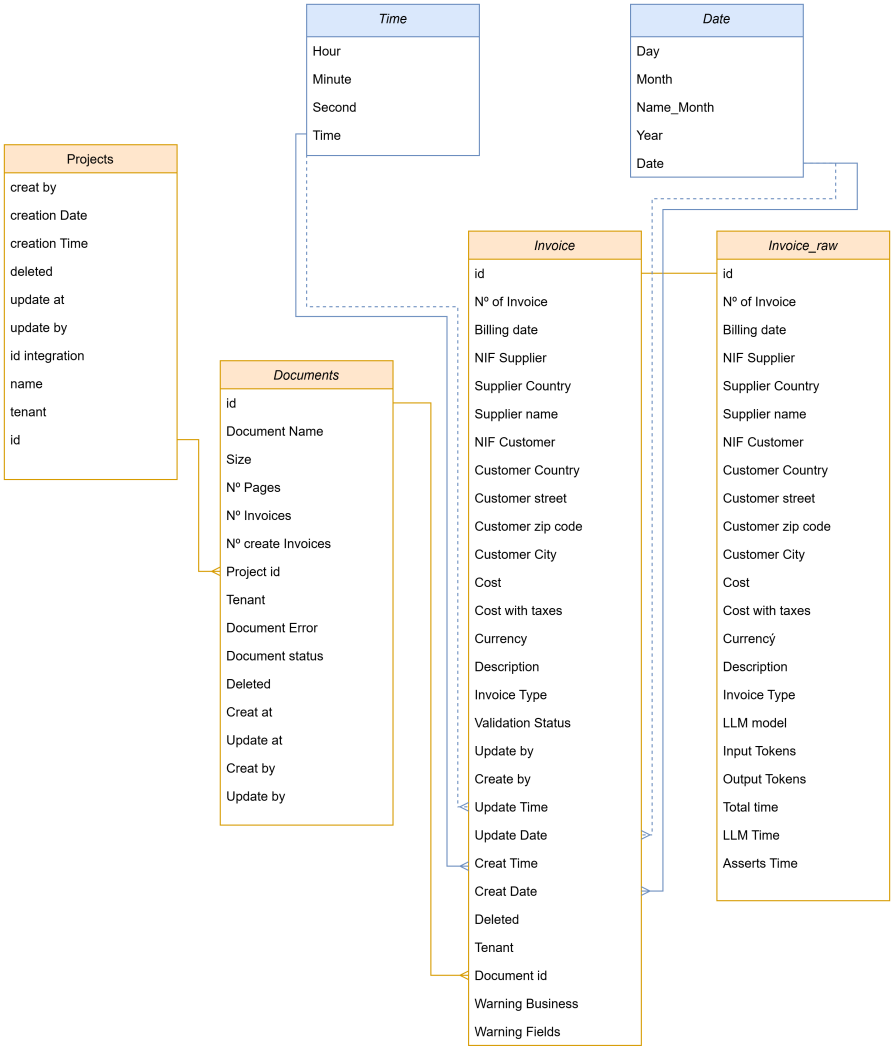


Figure 2: Relationship diagram for invoice data monitoring model using a Snowflake Schema

The database schema adopted in this part of the study is presented in Figure 2 and follows a Snowflake Schema model, which is particularly well-suited for analytical processing and multidimensional data analysis. As explained by (Gruenberg, 2022), the main difference between the traditional star schema and the snowflake schema lies in the structure of the dimension tables. In a snowflake schema, dimensions are normalized, meaning they can be split into multiple related tables connected through foreign keys, resulting in a structure that resembles a snowflake.

In this schema, the central fact tables, Invoice and Invoice_raw, store detailed transactional information extracted from the invoice documents, such as billing data, customer and supplier identifiers, costs and metadata related to document processing.

Surrounding the fact tables, several dimension tables provide contextual and categorical information. The Documents table functions as a core dimension, capturing attributes of each processed document, such as name, size, page count, error status and other metadata. This table is linked to the Projects dimension, which stores higher-level organizational data

and enables analyses grouped by project or tenant identifiers.

Temporal dimensions, namely Date and Time, are also normalized, in alignment with the snowflake design principles. This approach facilitates consistent and flexible time-based aggregations across various records and use cases.

The adoption of this schema is justified by several factors, the need to preserve the richness and variety of the original source data as much as possible, the normalization of dimensions, which reduces redundancy and the size of each join operation and the ability to enable efficient querying across high volumes of data and multiple analytical granularities (Benjelloun, El, & Amin, 2018; Gruenberg, 2022).

4.4.2 Dimensional Model for Log Analysis

Fact and Dimension Tables

The construction of the analytical model for log visualization was based on the logical separation between fact tables and dimension tables, as recommended in dimensional modeling approaches, which allows different types of events, such as processed documents, HTTP requests, errors and login sessions, to be represented independently while maintaining coherence and integrability through shared dimensions.

The fact tables, as described in the previous section, contain the main events monitored by the system with varying levels of granularity depending on the type of data analyzed, including doc_time, doc_logs_duration, doc_errors, doc_logs, req_path_duration, req_login, req_errors and req_logs.

To ensure multidimensional analysis these tables were linked to a set of common dimensions that contextualize the facts and enable flexible analytical segmentation such as the Date table which includes daily information like day, number of month, year and name of month allowing temporal analysis and grouping by period, the Time table which organizes events by hour, minute and second supporting the identification of time-based patterns response times and usage peaks and the doc_id and req_id dimensions which serve as primary keys for the document and request entities allowing events to be associated with their corresponding unique identifiers.

The relationships between fact and dimension tables were implemented in Power BI using the relationship management functionality, with each fact table connected to the relevant dimension,s ensuring referential integrity and enabling the use of cross-filtering across visualizatio.ns

Power BI Implementation and Analytical Logic

A view is essentially a stored query within the database, particularly useful for operations that are executed frequently. It is considered a key aspect of effective SQL database design (Inersjö, 2021; Obe & Hsu, 2017). Some purists argue that queries should always target views rather than base tables, as this approach simplifies permission management and promotes data abstraction (Obe & Hsu, 2017).

In line with this best practice, views were created in PostgreSQL before connecting the data to Power BI. These views were designed to streamline the dataset by excluding unnecessary columns that were not relevant to the analysis, resulting in lighter and more focused data.

The connection to the data source was established using the native PostgreSQL connector in Power BI, enabling direct access to the structured tables stored in the analytical database. Once connected, some data preparation was carried out in Power Query, where initial transformations were applied to ensure consistency and performance. These included the correction of column data types, filtering out irrelevant rows and the normalization of datetime fields by splitting them into separate Date and Time columns, allowing for finer-grained temporal analysis.

To optimize performance, categorical fields were correctly defined to support efficient filtering and grouping. These adjustments reduced the dataset size and improved the responsiveness of the visualizations. The analytical layer was built using DAX (Data Analysis Expressions), with the creation of key measures that support the evaluation of system performance and log activity. Examples include average number of errors per request, average number of errors per invoice created, average number of warnings per invoice created, average processing time per invoice, total number of errors, total number of documents, total number of invoices and many others.

These metrics were prioritized for their relevance in monitoring the system's behavior and identifying potential issues. Combined with interactive visual filters and slicers, they allow for exploring trends, isolating anomalies and assessing software usage patterns across different dimensions.

Schema Summary and Design Justification

The final data model, as illustrated in the relationship diagram presented in Figure 3, was structured according to the principles of dimensional modeling, with a clear separation between fact and dimension tables. This organization facilitates data exploration, supports performance optimization and ensures analytical scalability within the Power BI environment.

Given the nature of the system logs, which comprise a wide variety of event types, a single fact table would be insufficient to capture the heterogeneity and granularity required for meaningful analysis.

As proposed by (Thor, Golovin, & Rahm, 2005), who applied a Galaxy Schema in the context of a web data warehouse to structure and store information derived from web usage logs and recommendation systems, this schema was also adopted in this work, which is also described in Section 3.1.

This multidimensional modeling approach allows multiple fact tables to coexist, each representing a specific type of event or interaction, while sharing a common set of conformed dimensions such as Date, Time, doc_id and req_id.

With clearly defined relationships and modular fact tables, the model simplifies maintenance, promotes reusability of DAX measures and improves query performance in Power BI by avoiding unnecessarily complex joins or nested calculations.

It enables both high-level overviews and in-depth investigations into operational behavior, ensuring that the dashboards deliver actionable insights for both technical and managerial stakeholders.

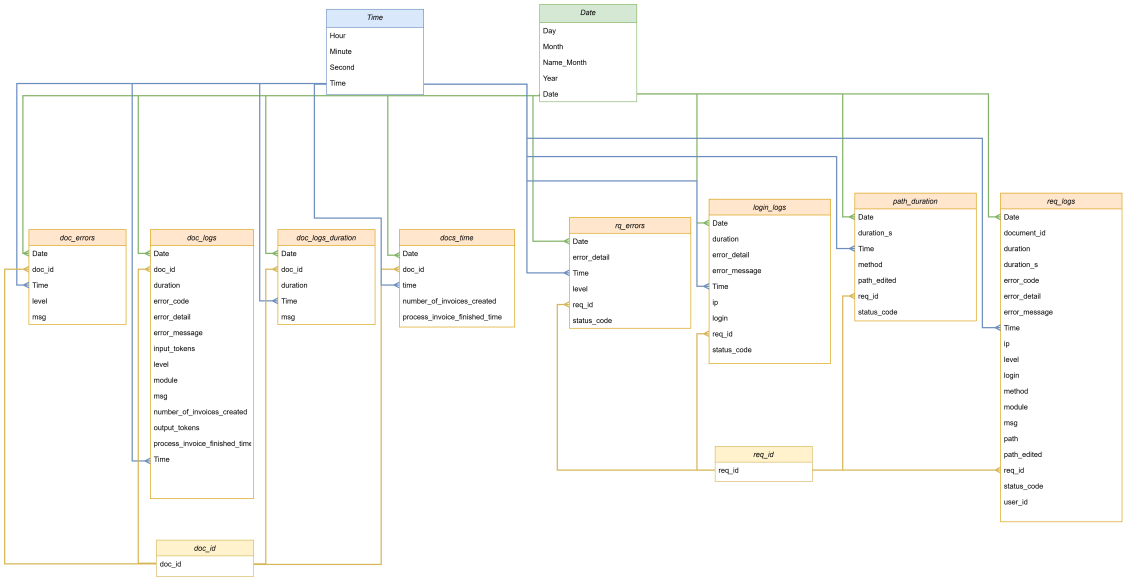


Figure 3: Relationship diagram for log monitoring model using a Galaxy Schema

4.5 Evaluation and Deployment

The Evaluation phase is dedicated to verifying the adherence of the results to the defined objectives. Here, it is crucial to interpret the model's results, identify possible adjustments and determine the next steps. In some cases, tools such as confusion matrices and SWOT analyses are employed to detail the solution's effectiveness (Gonçalves et al., 2025; Schröer et al., 2021).

Finally, the Deployment phase encompasses the planning, monitoring and maintenance of the implemented model or tool. Although many studies do not complete this step, it is considered critical to translate analytical results into real impacts on organizational processes (Schröer et al., 2021). Deployment may include anything from final reports to the integration of new systems in production environments (Gonçalves et al., 2025).

The practical orientation of CRISP-DM is one of its main strengths, as it provides a clear pathway to solving relevant problems in the real world, validating solutions in concrete scenarios. Its structured and flexible nature continues to be a reference for machine learning and data mining projects, even in the context of technological advances and increasing complexity of information systems (Gonçalves et al., 2025; Schröer et al., 2021; Zolbanin & Aubert, 2025).

Chapter 5

Dashboard Development and Analysis

This chapter presents the design, development and analytical interpretation of the dashboards created to monitor the performance of the DocGenius software.

Building upon the data modeling and preparation steps outlined in the previous chapters, two interactive dashboards were developed using Power BI, one focused on the quality of invoice data extraction and the other dedicated to the backend log monitoring.

These dashboards serve as practical tools for visualizing key performance indicators, detecting anomalies and supporting decision-making through data-driven insights. The sections that follow describe the structure, objectives and main analytical features of each dashboard, as well as their contributions to system transparency and continuous improvement.

5.1 Invoice Extraction Quality Dashboard

5.1.1 Overview of Dashboard Objectives

The Invoice Extraction Quality Dashboard was created with the main objective of analyzing the quality and consistency of the information extracted from invoices processed by the DocGenius system. Since the platform relies on accurate data extraction to structure key fields such as invoice numbers, amounts, dates and tax identifiers, it becomes essential to monitor how reliably these values are extracted across different documents and formats.

This dashboard is designed to answer the principal question: *"How accurate and consistent is the extraction of invoice fields in the DocGenius system and are there specific patterns or anomalies that indicate issues with the extraction process?"*.

To address this, the dashboard includes various visual indicators and validation metrics that highlight the frequency of extraction errors or inconsistencies across specific fields. It also enables the identification of documents with missing or invalid values, helping to detect recurring issues that may compromise data quality.

5.1.2 Dashboard Main Structure

The Invoice Extraction Quality Dashboard is structured across multiple analytical layers, with each page addressing a specific dimension of the extraction process. This segmentation allows for a comprehensive and targeted analysis of the quality and consistency of extracted invoice data. The dashboard comprises five main pages: *Overview*, *Field Level*, *Country Level*, *Data Level* and *Invoices vs Docs*.

All pages share a common set of four filters that allow users to refine the data being analysed. The first filter selects only the documents that were processed successfully. For all pages except the first one, this filter must be activated to ensure that the analysis is based only on completed and validated extractions. The second filter distinguishes the status of the invoice, whether it was edited manually, is still under review, or has been validated.

The remaining two filters provide additional control by enabling to refinement of the data based on a specific date range and individual user activity. These filters are particularly useful for narrowing down the analysis to a defined period or focusing on actions performed by a particular user.

The first page, **Overview**, can be found in Appendix A.7 and provides a high-level summary of system activity and extraction performance. This page includes an additional filter related to field priority. After a discussion with the development and operations teams, it was decided to classify fields by priority to support more focused analysis.

Priority 1 fields include mostly numeric and critical information such as client and supplier tax numbers, client postal code, total cost, invoice date and the countries of both the client and the supplier. In contrast, Priority 2 fields consist of more descriptive data such as client city and street, invoice description, supplier name and invoice type.

In this page, the processed_success filter is typically disabled when the goal is to analyze the number of documents that experienced extraction issues or required manual submission.

This page presents key performance indicators (KPIs) such as the number of users, documents and invoices processed, as well as the overall success rate of field extraction and the proportion of manually corrected invoices. Visual components include a time series chart showing daily invoice processing volumes, the overall success rate of field validation and a bar chart showing intervention rates per country.

A pie chart is also included to display the distribution of successful and failed field validations. However, it is important to note that this chart should only be interpreted when the Validated filter is enabled. Otherwise, unvalidated invoices are included in the calculation, potentially leading to an inflated success rate, since fields that have not been reviewed will be counted as valid by default.

The second page, **Field Level** and the third page, **Country Level**, shown in Figure 4 and Figure 5 respectively, were selected for further detailed analysis in the following subsections.

The **Data Level** page, presented in Appendix A.8, allows for the observation of whether the number of field-level interventions per invoice is stable over time or increasing. This is supported by a bar chart with an overlaid trend line. In the lower part of the page, a matrix allows users to explore field-level changes by invoice for a selected day, this selection is made directly through interaction with the chart. Only invoices submitted on the selected day are shown, allowing for targeted temporal analysis of data quality.

Finally, the **Documents vs Invoices** page, shown in Appendix A.9, presents a comparison between the number of documents uploaded and the number of invoices extracted. KPI values are shown at the top of the page, along with a time series line chart that tracks these metrics over time. The primary goal of this page is to detect whether any invoices were deleted by users and understand why.

As mentioned throughout this document, platform users may upload documents that include both invoices and non-invoice pages. Occasionally, the system may incorrectly classify a page as an invoice. When this occurs, users can manually delete the entry. To support this analysis, the bottom-left section of the page includes a matrix listing all document_ids in

the system, the number of invoices associated with each and an alert in case any invoice was deleted. By selecting a specific document_id, the table on the right updates to display detailed information about the fields of the deleted invoice. This feature allows for the identification of edge cases, such as shipping guides that resemble invoices in structure, which may confuse the classification logic of the system.

Together, these pages form a cohesive monitoring structure that facilitates both high-level oversight and in-depth analysis of the document extraction workflow. The modular layout of the dashboard supports continuous improvement efforts by enabling stakeholders to quickly identify quality issues, monitor field-level performance and assess the impact of changes to the extraction logic.

5.1.3 Field Level Page

The *Field Level* page offers a granular perspective on the performance of the software's automatic extraction process, focusing specifically on individual fields obtained from invoices, as illustrated in Figure 4.

The primary objective of this page is to assess, at the attribute level, which fields are being correctly extracted by the system and which tend to require user intervention. In this context, the page seeks to answer the following question: *"Which specific fields extracted from invoices are most prone to manual intervention and what is the success rate of the automatic extraction process at the attribute level?"*.

To support this analysis, two key indicators are presented at the top of the page. The first, Fields by Invoice reflects the total number of expected fields to be extracted per invoice, serving as a baseline for evaluation. The second indicator, Success Rate, corresponds to the percentage of values successfully extracted by the system without requiring manual corrections, thus providing a global measure of extraction quality.

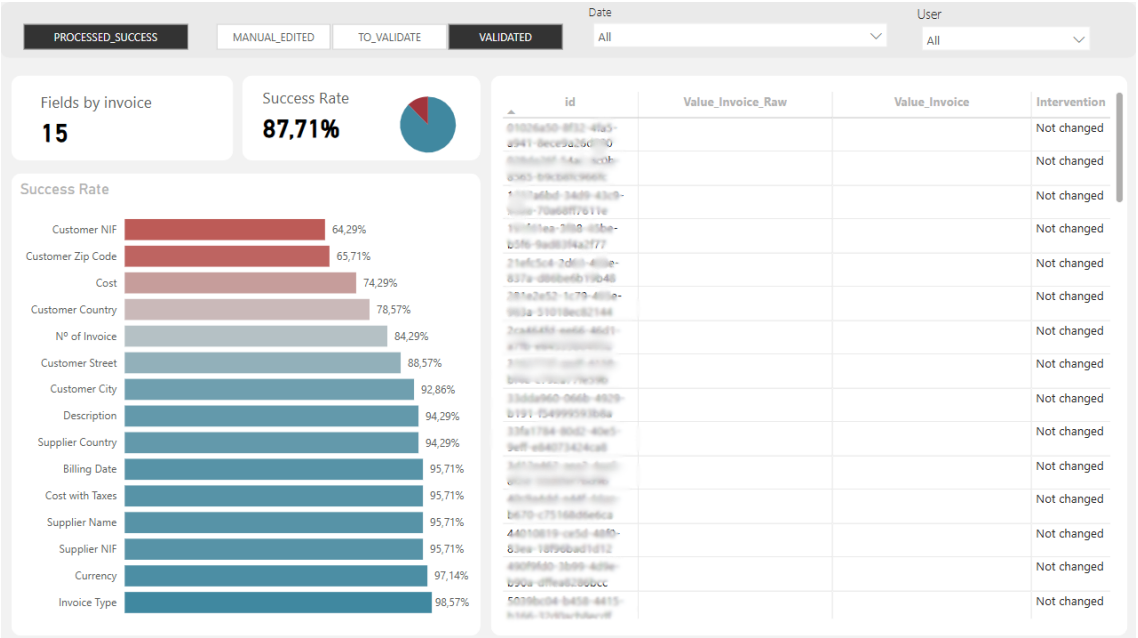


Figure 4: Field Level page of the Log Monitoring Dashboard

The central component of the page is a horizontal bar chart that displays the success rate per field. This visualization allows for a rapid comparison of the automatic extraction performance across different attributes, highlighting which fields achieve high levels of accuracy and which exhibit lower success rates, thereby signaling potential weaknesses in the extraction model or limitations in the source documents.

In addition to this summary visualization, the page includes a detailed table composed of three columns: Raw Value, Final Value and Intervention Status. The Raw Value represents the value initially extracted by the system, while the Final Value corresponds to the value confirmed or modified by the user during the validation process. The Intervention Status indicates whether a manual change occurred, with "Changed" referring to corrected values and "Not Changed" indicating those that remained unchanged. To facilitate interpretation, corrected values are visually highlighted in red. This tabular format enables users to directly compare the extracted versus validated values, fostering the identification of patterns of recurrent manual corrections.

The page also incorporates a degree of interactivity. Users may select a specific field to view all corresponding values across the available invoices, which is particularly useful for tracking inconsistencies. Additionally, a drill-through functionality is available, allowing users to right-click on an invoice_id and navigate to a detailed page that consolidates all fields associated with that invoice. This Detail Page provides a comprehensive view of the extraction and validation process for a specific invoice and is particularly valuable in audit contexts or for conducting in-depth case analyses. An example of this page, based on a fictitious invoice, is presented in Appendix A.10.

In summary, the Field Level page constitutes a fundamental analytical layer within the Invoice Extraction Quality Dashboard, offering a detailed and structured approach to identifying field-level deficiencies in the automatic extraction process.

Analysis

From the data displayed on the Field Level page, it is evident that the overall extraction performance remains high, with a total success rate of 87.71%, as shown in the top-level indicator. This means that nearly 13 out of 15 fields are correctly extracted without requiring manual correction.

A more detailed analysis, however, reveals significant variability across individual fields. Fields such as "Invoice Type" (98.57%), "Currency" (97.14%) and "Supplier Name" (95.71%) exhibit near-perfect success rates, indicating a high level of consistency in their automatic recognition. These fields likely follow standardized textual formats or appear in consistent positions on invoice templates, which facilitates accurate extraction by the system.

In contrast, fields like "Customer NIF" (64.29%) and "Customer Zip Code" (65.71%) show significantly lower success rates, suggesting that they pose greater challenges for the extraction model. This may result from variations in how these values are presented, such as differences in format, position or image quality, as well as the strict formatting rules these fields often require, including the use of separators or special characters.

A closer inspection reveals that, in the case of "Customer NIF", the low success rate is not due to incorrect values but to missing extractions altogether. The software often fails to detect any value for this field, requiring users to enter it manually. For "Customer Zip Code", multiple failure modes were observed, ranging from the complete absence of a detected value,

to incorrect formatting (e.g., the software extracting "4925104" instead of the required "4925-104") and even cases where the value extracted belonged to the supplier instead of the client.

By highlighting these discrepancies, this page provides actionable insights to improve extraction logic, particularly for structurally complex or inconsistently formatted fields.

In summary, while the system demonstrates strong performance across most fields, a small subset consistently underperforms. Ongoing monitoring at this level supports targeted enhancements and fosters a data-driven approach to improving extraction accuracy.

5.1.4 Country Level Page

The *Country Level* page of the Invoice Extraction Quality Dashboard, presented in the Figure 5, offers a geographical perspective on data extraction performance, enabling users to monitor and compare the relative frequency of manual interventions across different supplier countries.

This type of analysis is essential for identifying country-specific challenges related to invoice format heterogeneity, language differences or localized layout patterns that may hinder automatic data extraction. In this sense, the page is designed to answer the question: *"Are there specific countries whose invoices consistently require more user intervention and which fields are most affected in each case?"*

The page includes several key components designed to support multi-level exploration. The key indicators include the total number of supplier countries, shown through a numeric card that summarizes the total number of distinct supplier countries processed. A summary table is also presented, showing, per country, the number of relative interventions (total relative number of fields requiring user correction), the percentage of relative interventions (the proportion of altered fields relative to the total extracted for that country) and the number of invoices received from that country. Additionally, a line chart displays the relative percentage of interventions over time. This visualization helps assess whether the system's performance is improving, declining or remaining stable. Finally, a horizontal stacked bar chart provides a detailed breakdown of which fields were most frequently corrected per country. Each colored segment represents a specific country, enabling quick comparison of field-specific issues across regions.

One of the most valuable features of this page is its interactivity. Users can click on any country in the summary table to dynamically filter all visual elements on the page, allowing the analysis to focus exclusively on that specific location. Furthermore, a drill-through functionality allows the user to navigate to a secondary detail page, similar to the one described in the Field Level section, where the raw and final values for each modified field can be individually inspected. This enables full traceability of every intervention performed by the user.

This page confirms that performance variability in invoice extraction is not only field-specific but also geographically distributed. Countries with higher relative intervention rates may be using invoice layouts that diverge more significantly from the structural patterns on which the extraction model was trained. Ultimately, the Country Level view strengthens the analytical framework by introducing a spatial dimension into the monitoring process, ensuring that performance optimization efforts are responsive to both technical limitations and contextual variability in document structure.

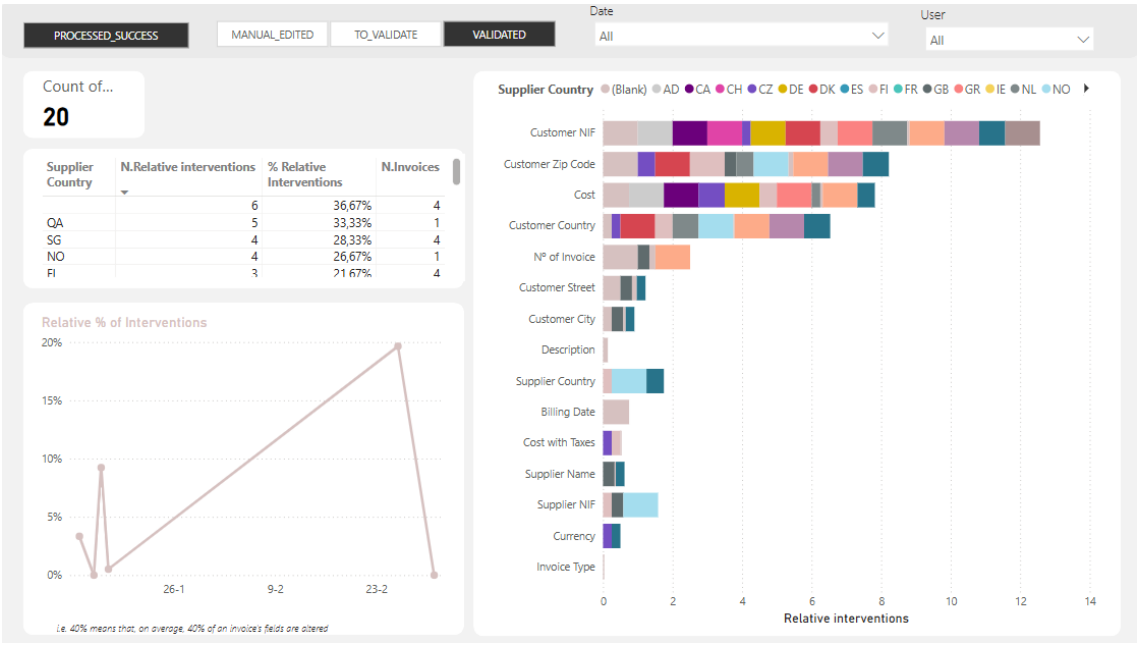


Figure 5: Country Level page of the Log Monitoring Dashboard

Analysis

From the data displayed on the Country Level page, it becomes clear that extraction performance varies significantly depending on the origin of the invoice.

Countries such as QA, SG, NO and FI exhibit higher levels of manual interventions relative to the number of fields extracted. It is important to note that countries appearing as blank entries are not considered in this analysis, since the supplier country is not a mandatory field and in some cases, may be missing entirely.

In the case of QA (Qatar), for example, five out of the fifteen fields extracted from the only available invoice required correction, resulting in a relative intervention rate of 33.33%. While this percentage appears high, it must be interpreted with caution due to the very limited sample size. A drill-through analysis reveals that these corrections were concentrated in the fields "Customer Zip Code", "Cost", "Customer NIF", "Nº of Invoice" and "Customer Country".

In comparison, invoices from SG (Singapore) show an average of 4 corrected fields out of 15, corresponding to a relative intervention rate of 28.33%. A deeper analysis (shown in the Appendix A.11) reveals that these four corrections are distributed across nine distinct fields. On average, each invoice contains approximately 0.25 corrections in fields such as 'Customer Street ", "Currency", "Supplier Name" and 'Customer City ", 0.5 corrections occur in "Cost" and "Supplier Country" and 0.75 corrections are observed in "Customer Zip Code", "Customer NIF" and "Customer Country". This indicates that while the corrections are dispersed, certain fields consistently present more challenges.

These findings suggest that the structure and formatting of invoices from these countries may pose specific challenges to the extraction model. Non-standard layouts, unusual value formats or language differences may negatively impact the system's ability to accurately recognize and extract information.

Conversely, many countries in the dataset show minimal or even zero corrections, reinforcing the idea that layout consistency and alignment with the model's training data play

a critical role in successful extraction. Countries such as IE (Ireland), FR (France) and ES (Spain), with 1, 3 and 4 invoices respectively, are among those with the best overall performance, where the extraction engine produced highly accurate results across most fields.

Additionally, the relative percentage of interventions over time, as displayed in the line chart, suggests that performance fluctuations may also be linked to the introduction of new invoice formats or templates in specific regions. Monitoring this trend over time provides valuable context for identifying when and where degradation or improvements occur.

By combining field-level granularity with country-level segmentation, this page adds a crucial spatial dimension to the performance analysis. It not only reveals where the extraction model performs inconsistently, but also helps prioritize countries and fields for targeted improvement.

5.2 Log Monitoring Dashboard

5.2.1 Overview of Dashboard Objectives

As previously explained throughout the document, the absence of native monitoring mechanisms in the DocGenius software motivated the development of a dedicated dashboard for analyzing the logs generated by backend operations. The main goal of this dashboard is to provide a comprehensive analytical view of the system's internal behavior through log analysis.

This dashboard enables the detection of operational anomalies, monitoring of critical failures (such as the occurrence of 401 or 400 error codes), tracking of endpoint performance, identification of usage patterns by user and endpoint and assessment of the system's overall efficiency and stability over time.

By centralizing technical metrics into interactive visualizations, the dashboard supports both the technical team in error diagnosis and the management team in identifying optimization opportunities.

5.2.2 Dashboard Main Structure

The Log Monitoring Dashboard was developed with a modular and hierarchical design, in which each page is dedicated to a specific analytical perspective, collectively contributing to a comprehensive understanding of the backend performance of the system. The dashboard is structured into several distinct pages, each addressing a different operational dimension.

The **first page**, presented in Appendix A.12, provides an overview of the system's activity through a consolidated presentation of key performance indicators (KPIs). To begin with, a date slicer allows the data to be filtered according to the desired analysis period. Regarding the KPIs, the dashboard displays the total number of simple requests, document-related requests, users and distinct paths accessed within the system.

Donut charts illustrate the distribution of log levels (INFO, WARNING and ERROR), as well as the proportion of HTTP methods used (GET, POST, PUT and DELETE). HTTP status codes are represented through a bar chart, while the volume of input and output tokens is visualized using a line chart.

The average processing time per request is also prominently displayed. On the lower right side of the page, a line chart shows the daily variation in the number of logins, simple requests and document-related requests, enabling a temporal analysis of system usage.

The **second page** of the dashboard, included in Appendix A.13, is dedicated to the analysis of system logins. This page allows for the observation of both the temporal distribution of accesses and specific metrics related to login duration and failed login attempts.

In the upper right corner, a time slicer is available, enabling data filtering by a selected date range. On the upper left, the total number of users is displayed. The section Login Duration Metrics summarizes the minimum, average and maximum login durations, allowing the identification of any outliers or discrepancies. Below this section, a table lists individual users, their respective login times and corresponding IP addresses.

At the center of the page, an additional table displays authentication error cases, with emphasis on status code 401 (Unauthorized). The affected usernames, the associated error message, the number of occurrences and the related IP addresses are shown, making it possible to identify patterns of failed access attempts.

Finally, a line chart located at the bottom of the page illustrates the variation in the number of logins over time, supporting the analysis of usage trends and access peaks.

The **third** and **fourth pages** were selected for a more in-depth analysis. They are described in detail in Section 5.2.3 and Section 5.2.4, respectively.

Finally, the **fifth page** of the dashboard, presented in Appendix A.14, focuses on analyzing invoice processing times. It provides insights into both overall and component-level durations within the invoice handling process.

At the top, a date filter and a document_id selector allow users to narrow down the data for analysis. The line chart on the left shows the daily average processing time of invoices, helping identify fluctuations or trends over the selected period.

On the right side, a table displays detailed information for each document_id, including the number of processed invoices and the average time taken to complete the process.

At the bottom, another table lists the average duration of each subprocess involved in the invoice workflow, such as API calls, validation steps, OCR and QR code processing, offering a breakdown of where time is being spent within the process.

Each page integrates a combination of visual elements designed to maximize clarity and analytical value. Summary cards (KPIs) are used to display key metrics such as the number of requests, document interactions, users and average processing durations. Donut and bar charts illustrate categorical distributions, including log levels, HTTP methods and status codes. Line charts are employed to show temporal trends and usage fluctuations, while interactive tables allow for detailed exploration of specific data points such as user behavior, endpoint performance, error messages and document-level logs.

This structured and layered approach enhances the interpretability of backend log data, enabling both technical teams and decision-makers to monitor system stability, track performance evolution and respond proactively to anomalies.

5.2.3 Endpoints Page

One of the key pages of the Log Monitoring Dashboard is dedicated to the analysis of *API endpoints*. This page, illustrated in Figure 6, is designed to address the following analytical question: *"Which API paths are experiencing the highest number of errors or performance degradation and how consistent is their behavior over time? Are there any identifiable patterns associated with these errors?"*

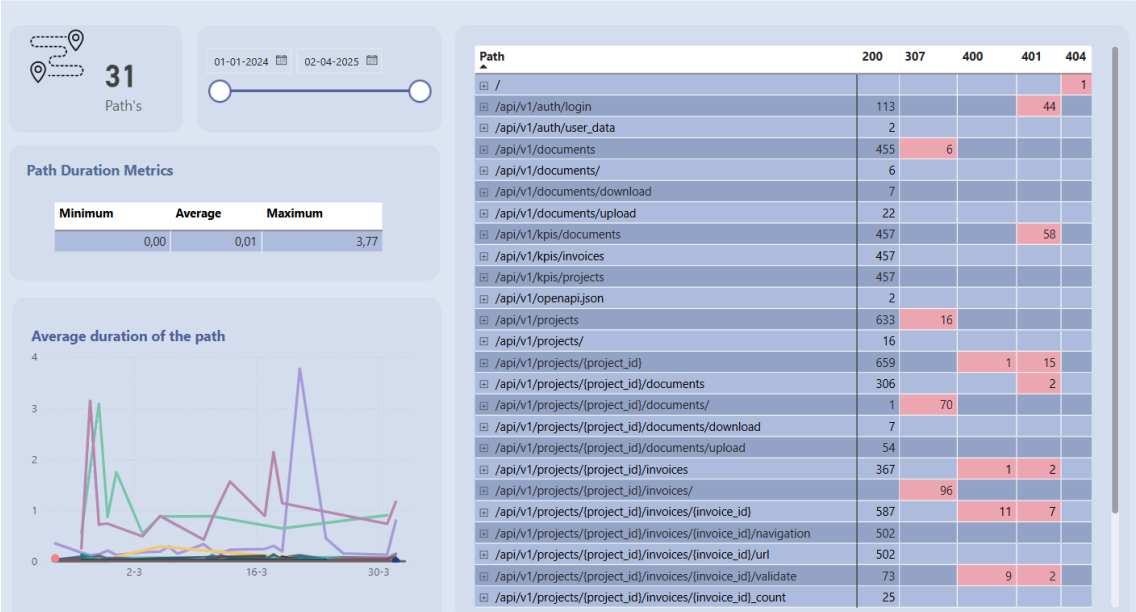


Figure 6: Endpoints analysis page of the Log Monitoring Dashboard

To answer this question, the page offers a detailed operational view of the backend flows by analyzing each accessed path in terms of response time and error frequency. The layout follows a structured and intuitive design, combining summary indicators, visual trends and detailed tabular data to support both high-level system monitoring and granular analysis.

At the top of the page, users can apply a dynamic date filter to define the analysis period, ensuring that all visual elements and metrics are updated accordingly. The left panel presents key indicators summarizing the number of unique API paths accessed. Just below, a horizontal bar visualizes the Path Duration Metrics, including the minimum, average and maximum response times, providing immediate insights into performance variability. A line chart titled Average Duration of the Path illustrates the evolution of response times over time, allowing for the detection of latency peaks or anomalies across endpoints.

On the right-hand side, the core analytical component is an interactive matrix table displaying all API endpoints along with the corresponding count of HTTP status codes (e.g., 200, 307, 400, 401, 404). The table uses conditional formatting to highlight error-related codes in red, enabling quick identification of endpoints with recurring issues. For better readability and aggregation, dynamic route parameters (such as *project_id* or *invoice_id*) are generalized, as explained in Section 4.3.2.

This page not only facilitates the identification of technical debt and performance bottlenecks but also supports prioritization of corrective actions based on usage patterns and error frequency.

Analysis

The analysis of the *Endpoints page* enables the identification of usage patterns across the various request paths (endpoints) of the DocGenius backend, based on structured request log data. Through the interactive visualizations, it is possible to observe the distribution of calls per endpoint, the average response time associated with each one and the most frequently used HTTP methods, revealing not only which endpoints are most accessed but also where

performance bottlenecks or inefficiencies may occur.

The data show that some endpoints have a significantly higher volume of requests, particularly routes such as */docs*, */api/v1/utils/country* and */api/v1/projects/*{*project_id*}.

The */docs* endpoint accounted for a striking 44,654 out of 53,407 total requests, representing approximately 87% of all entries. Upon investigation, it was found that this route was automatically invoked every minute as a heartbeat check to verify whether the software was up and running. While technically useful, this endpoint provides no analytical value and its high frequency was significantly inflating the volume of log data.

This insight proved crucial in reducing the load on the database and led to important conclusions regarding the need to exclude operational noise from analytical processing, ensuring that performance analyses reflect truly relevant user interactions.

Excluding */docs*, the most accessed endpoints were */api/v1/utils/country* (1,297 requests) and */api/v1/projects/*{*project_id*} (651 requests). A closer look at */api/v1/utils/country* revealed that all requests were GET methods, consistent with its role in populating country selection dropdowns across multiple interface pages. As for */api/v1/projects/*{*project_id*}, the majority of calls (over 99%) were also GET requests, triggered whenever a project is opened by a user, only 0.76% of the interactions were PUT requests, confirming that update operations are comparatively less.

When analyzing the metric of average response time per endpoint, it becomes evident that endpoints related to more complex operations (such as file uploads, document loading and login processes) tend to exhibit higher response times. This is expected, given the nature of the processing involved. Nevertheless, most recorded average times remain within acceptable limits, with no evidence of excessively high delays.

As status codes 400 and 401 are analyzed in detail on the Errors and Warnings page, one of the main insights on this page was the recurring identification of status code 307, associated with temporary redirects. The data revealed that this code appeared in two specific endpoints, both following a distinct pattern: the request path included an additional trailing slash (e.g., **/projects/** instead of **/projects**). It was observed that whenever users accessed the version of the endpoint with the trailing slash, the server automatically issued a redirect to the canonical path without it, thereby generating a 307 response.

Although this behaviour is technically correct, it carries relevant practical implications, as the system treats these as additional requests, which may affect both the total count of requests and the perceived response time. This pattern, identified through the analysis of this page, underscores the importance of normalizing API paths in client-side calls to avoid unnecessary redirects and to improve both performance and system traceability.

It can thus be concluded that the dashboard effectively addresses the research question, as it enabled the identification of a consistent redirection pattern (status code 307) as well as the most frequently accessed endpoints.

These findings reinforce the relevance of endpoint-level monitoring, not only to detect system inefficiencies (such as unnecessary redirects or automated probes) but also to better understand user interaction patterns.

Overall, the page provides a clear overview of the distribution of calls across endpoints and their average performance, serving as a foundation for identifying which areas of the system handle the highest load and should therefore be prioritized in future monitoring and optimization efforts.

5.2.4 Errors and Warnings Page

The *Errors and Warnings* page is primarily designed to analyze the errors and warnings recorded during the operation of the DocGenius backend system, enabling the identification of failure patterns and their temporal evolution. This page is intended to answer the following research question: *"What are the most common errors and warnings in the system? Are there identifiable causes or patterns associated with these failures that should or could be addressed?"*

To support this analysis, the page is divided into two main sections, Request's, referring to failures in HTTP requests and Docs, which relate to errors that occurred during document processing. In both sections, events are grouped by severity level (ERROR and WARNING), accompanied by descriptive error messages and their respective frequencies. Additionally, the associated doc_id or req_id values can be expanded, enabling a more granular inspection of where each issue occurred.

At the top of the page, a time series chart displays the distribution of errors and warnings, with the possibility to toggle between three categories: Errors by invoices, Warnings by invoices and Errors by requests. This categorization was intentionally designed as "by number of entities" (e.g., invoices or requests) to allow for a relative measurement of events, rather than absolute counts, thus improving the interpretability of the visualization and avoiding distortion caused by volume fluctuations.

This temporal view enables the detection of specific error peaks and periods of increased instability, supporting the identification of recurring anomalies and their possible correlation with system updates or operational changes.

Overall, this page plays a fundamental role in identifying and categorizing the system's most relevant technical failures, serving as a key diagnostic tool to support the prioritization of system improvements. The separation between request-level and document-level errors enables a more targeted analysis per technical domain, while the temporal dimension allows tracking the consistency and progression of failures over time.

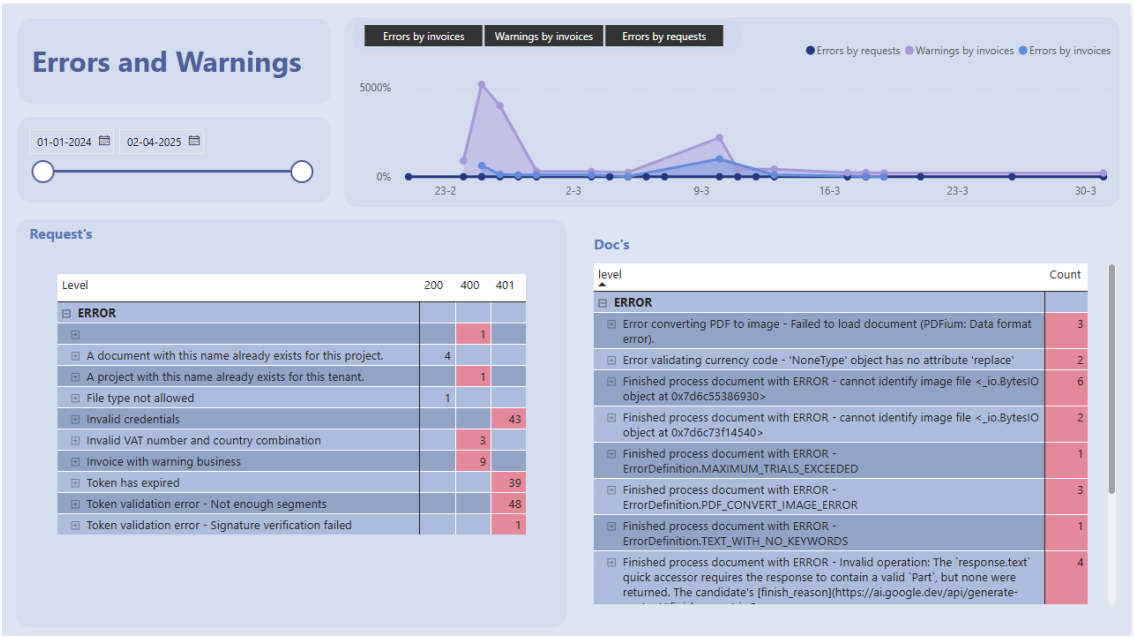


Figure 7: Errors and Warnings page of the Log Monitoring Dashboard

Analysis

From the data displayed, it is evident that the most frequent errors in HTTP requests are related to token validation and authentication mechanisms. Specifically, the errors "Token validation error – Not enough segments", "Token has expired" and "Invalid credentials" stand out as the most common issues under the Request's section. These errors are strongly associated with status code 401, indicating unauthorized access. This suggests systemic issues in session handling, token life cycle management that reflect the user behavior (e.g., repeated attempts with expired or malformed tokens).

In the Doc's section, errors are predominantly related to document conversion, image recognition and field extraction processes. Examples include "Finished process invoice with ERROR -ErrorDefinitionİNVOICE_NUMBER_ALREADY_EXISTS", "Error converting PDF to image – Failed to load document (PDFium Data format error)" and "Cannot identify image file <_ioBytesIO object>". These indicate backend limitations when processing unexpected or corrupted file formats, which can compromise the reliability of the document processing pipeline.

Regarding the warnings (which can be viewed by scrolling through the table), valuable insights emerged that contributed to the improvement of the software's processing logic. These include warnings such as "Error validating and converting number - 1.153,74€" and "Error validating and converting number - 1.852,76.Kč". Through these messages, it was possible to identify that the LLM occasionally included currency symbols in the extracted cost values, which subsequently caused conversion errors, leaving the affected fields empty at the end of the pipeline.

Another important insight comes from the message "Error validating currency code - 'NoneType' object has no attribute 'replace' ". Although technically classified as an error, this case would be more appropriately handled as a warning, since the absence of an explicit currency code on the invoice may be an expected scenario rather than a critical failure. In such cases, the software should not attempt to process a missing currency code, thus avoiding the invocation of the replace attribute on a NoneType object.

The temporal chart at the top of the page reveals clear error spikes, most notably in early March, where a sharp increase in both errors by invoices and errors by requests was observed. These peaks may correspond to system updates, increased usage or changes in external integrations and thus warrant further investigation. The decision to normalize error data "by number of invoices" or "by number of requests" allows for a more accurate detection of relative anomalies, avoiding distortion caused by absolute volume fluctuations.

In summary, the page reveals clear and recurring failure patterns, which are both consistent over time and concentrated in specific technical areas: authentication-related issues on the request side and file processing errors on the document side. These insights are essential for prioritizing corrective actions, such as refining error categorization and enhancing the robustness of the document parsing and extraction processes.

5.3 Final Remarks

The development of the two dashboards, one focused on invoice extraction quality and the other on backend log monitoring, represents a significant step toward improving the transparency of the DocGenius platform. By transforming raw system data into structured

insights, these dashboards support both technical diagnostics and strategic decision-making.

The invoice dashboard enables the identification of systematic extraction issues, recurring correction patterns and field-level performance indicators. It provides a clear overview of the software's ability to handle documents from different suppliers and highlights areas that may benefit from further refinement or automation.

In parallel, the log monitoring dashboard offers a detailed view of the system's internal behavior, enabling the detection of anomalies, tracking of processing times and identification of recurring errors.

Together, these dashboards contribute to a comprehensive performance monitoring framework, allowing stakeholders to act on empirical evidence and implement continuous improvements with confidence.

Chapter 6

Analytical Pipeline for Invoice Data Validation and Automatic Correction

This chapter introduces a complementary analytical pipeline designed to enhance the robustness of the invoice processing system.

While the previous chapters focused on performance monitoring through dashboards and log analysis, this section addresses the need for automated mechanisms that proactively validate and correct extracted invoice data.

In real-world scenarios, automated extraction systems often face issues related to inconsistent formats, missing fields or misclassified values. The goal is to support data quality assurance by identifying and resolving common extraction errors automatically, thereby reducing manual intervention and improving overall system efficiency. The following sections detail the motivation behind this pipeline, its architecture, the technologies and algorithms employed, as well as the expected benefits and limitations.

6.1 Context and Motivation

In addition to the analysis originally proposed for this project, it was decided to include this chapter to actively contribute to the improvement of the invoice extraction and validation process. Despite the enhancements provided by the analytical dashboards developed throughout this work, the validation of extracted data still relies heavily on manual intervention, particularly in cases where inconsistencies are found in the fields automatically identified by the DocGenius software.

Although the system represents a significant advancement by reducing the manual digitization effort required from the client, it still demands human validation to ensure the integrity of the final data. This dependency compromises the scalability of the process and keeps the client's workload higher than desirable.

To mitigate this limitation, this chapter proposes an automated pipeline for invoice validation and correction, based on the reuse of previously validated information and the application of similarity and prediction algorithms. The main objective is to reduce the number of invoices requiring manual validation, improve the consistency of the recorded data and consequently, lessen the effort required from the end user during the correction process.

6.2 Description of the Automated Workflow

On the DocGenius platform, clients are required to manually validate all invoices, either by confirming that the extracted data is correct and validating without changes, or by correcting the necessary fields and then validating the document.

Figure 8 presents the diagram that summarizes the proposed automated workflow for invoice validation.

The pipeline begins by checking for the existence of previously validated invoices from the same supplier. For this comparison, the supplier's tax identification number (NIF) was selected as the matching field, as invoices from the same supplier generally follow a similar structure.

If no validated invoices are associated with the same NIF, the document is immediately routed for manual validation, since there is no historical record to support automatic inference or comparison. If such records do exist, the workflow is split into two distinct branches, depending on whether the previously validated invoices required manual corrections.

In cases where the validated invoices did not undergo any manual changes, the next step is to verify the QR Code structure. This verification is essential, as the presence or absence of QR Codes can significantly impact the extraction process and the consistency of the resulting data. If the QR Code pattern in the current invoice differs from that of previously validated invoices, the document is redirected for manual validation. However, if both invoices use or do not use QR Codes and the number of extracted fields is the same, a text similarity algorithm is applied to the populated fields. If the calculated similarity score exceeds a predefined threshold, the document is automatically validated, under the assumption that the fields were correctly identified.

In cases where the number of extracted fields differs between the current invoice and previously validated documents, alternative prediction algorithms are triggered to handle those specific fields more efficiently.

When the historical data indicates that previously validated invoices did undergo manual corrections, the system attempts to detect correction patterns. If five or more invoices have been corrected in the same fields, this is interpreted as a recurring pattern. In such cases, a similarity algorithm is used again to compare the five corrected invoices to the newly submitted one. If the confidence threshold is met, the same corrections are applied automatically.

The choice of using five or more invoices as the minimum threshold for detecting correction patterns is based on a balance between operational feasibility and empirical stability. With fewer than five examples, the risk of capturing outliers or isolated corrections increases significantly, which can lead to inaccurate inferences and weak generalizations. This risk is particularly relevant in automated validation systems, where corrections applied based on inconsistent patterns may compromise data integrity.

This issue is closely related to the high variance associated with small datasets, one of the main contributors to generalization error in machine learning systems. As Domingos (2012) explains, the total error of a model consists of bias, variance and noise and variance tends to increase when the volume of data is small, leading models to learn spurious or irrelevant patterns.

On the other hand, setting a higher threshold could significantly delay the activation of the correction mechanism, especially in contexts with low document volume or infrequent suppliers. Therefore, the value of five was adopted as a practical and prudent compromise

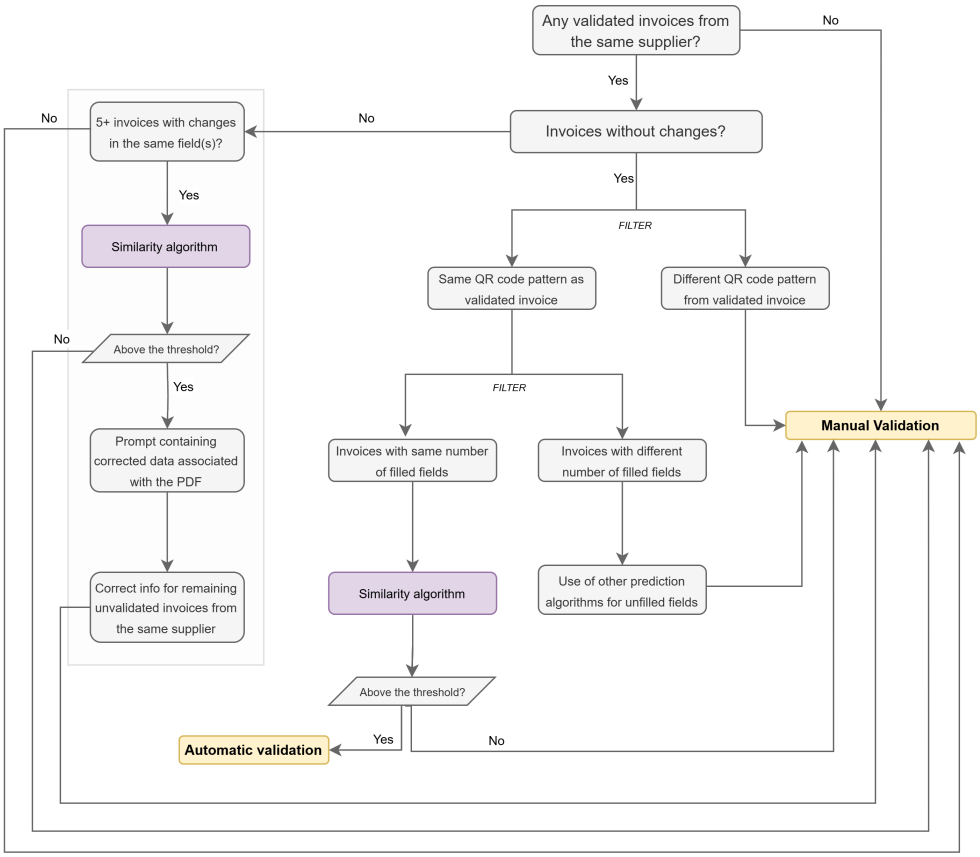


Figure 8: Pipeline for Invoice Data Validation and Automatic Correction

it provides a sufficiently robust foundation for identifying recurring patterns with greater confidence, while preserving the responsiveness needed for effective process automation.

To ensure the reliability of these automatic corrections, a Large Language Model (LLM) is integrated at this stage. A custom prompt is constructed and sent to the model, including, the original values extracted by the LLM, the manual corrections made and the image of the invoice itself.

Based on this input, the LLM generates updated values for the new invoice and for other future invoices that follow the same pattern. The model's suggestion is only applied if its confidence score exceeds a predefined threshold. Otherwise, the invoice follows the standard manual validation process.

At any point in the pipeline, if a satisfactory confidence level cannot be reached, the document is redirected to human validation.

Overall, this process significantly reduces the volume of documents awaiting validation, relying on historical evidence derived from both the structure of the documents and user behavior, thus promoting greater efficiency, consistency and automation in the invoice handling process.

6.3 Technologies and Algorithms Used and Proposed

The implementation of the automated workflow for invoice validation and correction is based on a combination of heuristic techniques and machine learning models, with particular emphasis on visual similarity algorithms and Large Language Models (LLMs).

To identify visually similar invoices, a variety of image comparison libraries were tested using the available dataset. Among the tools explored were SIFT, OpenCV (cv2) and pHash.

The SIFT (Scale-Invariant Feature Transform) algorithm, as described by (Hua, Chen, Wei, & Jiang, 2012), extracts distinctive features from both the original and resized invoice images, enabling the calculation of the distance between feature vectors to assess the degree of similarity. SIFT is robust to transformations such as translation, scaling, rotation, and affine distortion, as well as changes in lighting and image noise, making it particularly well-suited for image matching tasks involving scanned or photographed documents.

pHash is a perceptual hashing method based on frequency information. However, as discussed by Oyen, Kucer, and Wohlberg (2021), it was found to be highly sensitive to small visual differences, resulting in a considerable number of false negatives in invoice scenarios.

OpenCV (cv2) proved to be the most effective library during testing. Its strengths lie in detecting structural differences, such as the edges and contours of invoice elements, rather than focusing on textual or pixel-level content. This makes it ideal for scenarios where invoice content, such as line items, totals and descriptions, naturally varies between documents, but the overall structure remains consistent. Furthermore, OpenCV includes image alignment and perspective correction functionalities, enabling more accurate comparisons even when documents are slightly skewed or misaligned (Karapet, Savitskyi, & Vakaliuk, 2024)

As part of this architecture, one of the key innovations tested with promising results during the pilot phase was the use of LLMs via structured prompting, especially in scenarios where correction patterns were identified across multiple invoices.

The process begins with standard OCR and document parsing using Gemini (the same model used in the current implementation). The system first detects the supplier's tax ID (NIF) and if five visually similar invoices from the same supplier contain identical manual corrections in the same field, a specialized prompt is generated and sent to the LLM.

This updated prompt includes the original (initially extracted) values, the corrected values entered manually and the associated invoice image. By combining this structured context with few-shot examples, the model is able to learn from previous user behavior and apply corrections automatically to new, similar invoices.

This few-shot prompting strategy simulates the reasoning of an experienced user and allows the model to propose corrections based on contextual cues and historical patterns. As discussed by (Cheng et al., 2024), this approach aligns with the concept of "additional prompts," which are used to correct or prevent specific types of recurring extraction errors. These prompts are designed to guide the model toward more accurate outputs in situations where common extraction failures are known to occur.

The test case realized shows the efficacy of this approach. Initially, the software misidentified the "subtotal" (amount without VAT) in five structurally identical invoices, incorrectly registering the value before discount instead. After correction patterns were established, the model successfully identified and extracted the correct field in a new, previously unseen invoice, which would otherwise have resulted in an error.

This technological stack supports the creation of a flexible and scalable system, capable

of adapting to the variability of invoice data while ensuring the quality and reliability of the validation process. It also enables seamless integration with the analytical environment already established within the organization.

It is important to highlight that, although not tested during this internship, in scenarios where the number of extracted fields varies, indicating that the current model (Gemini) fails to infer certain values, alternative pre-trained language models such as GPT or LLaMA could be considered. These models are better suited for more flexible tasks, including conditional field completion and may provide improved performance when dealing with loosely structured or highly variable documents.

6.4 Expected Benefits, Limitations and Future Work

The expected benefits of the proposed workflow primarily include a significant reduction in invoice validation time, lower dependence on manual processes and an increase in the reliability and consistency of the extracted data. By automating key steps in the validation process and leveraging both historical data and contextual information, the system is designed to minimize human intervention while maintaining high-quality standards. In addition, the integration of corrective logic into the analytical pipeline enables the identification of recurrent errors and the application of standardized treatments, contributing to a more scalable and efficient data validation process.

However, although the proposed approach is promising, it still requires extensive testing with real-world data in a production environment in order to validate its effectiveness. The current implementation is based on rule-based matching and similarity algorithms that, while effective in many scenarios, may underperform when faced with ambiguous or poorly structured inputs. Due to limitations in available labeled data, several components are left for future work, most notably, the definition and fine-tuning of the similarity threshold, the selection of optimal algorithms and the evaluation of alternative LLMs for extracting values with semantic awareness.

Furthermore, the use of machine learning models inherently implies the need for continuous monitoring, maintenance and retraining, as new document structures and data patterns emerge over time. These aspects must be carefully addressed to ensure the long-term adaptability and robustness of the system. The use of active learning strategies, where user feedback is reintegrated into model training, could significantly enhance the adaptability of the correction process.

Chapter 7

Conclusion

This project aimed to design and implement a comprehensive analytical solution for monitoring and improving the performance of DocGenius, a software platform dedicated to the automatic extraction of data from commercial invoices.

The project was initially proposed to fill an existing gap, the absence of structured monitoring tools capable of assessing the accuracy and efficiency of DocGenius. Although the software already performed automated extractions, there was no integrated mechanism to quantify its performance, identify frequent errors or monitor system behavior over time. Recognizing this gap, the project set out to design and implement an analytical framework that could provide DataColab with operational visibility and support informed decision-making.

To respond to this challenge, the project proposed an analytical approach based on Business Intelligence (BI) and data engineering principles. This led to the development of two comprehensive dashboards in Power BI. These dashboards were built upon data extracted from a PostgreSQL database and log entries obtained via the Kibana interface, structured and updated through a custom pipeline orchestrated by Apache Airflow.

However, performance monitoring alone is not sufficient to enhance software reliability. For this reason, the second major contribution of this work was the development of an automated pipeline for invoice data validation and correction. This component, implemented using Python, integrates rule-based logic and predictive mechanisms to automatically detect and address inconsistencies in the extracted data. By reducing the reliance on manual user intervention, this workflow enhances data quality, promotes consistency and contributes to a more scalable and intelligent document processing pipeline.

The results obtained demonstrated that the implemented solution brought substantial benefits to DataColab. The dashboards enabled the identification of frequent field-level correction patterns, highlighted country-specific anomalies and provided a clear overview of extraction success rates over time. From the log analysis, it became possible to monitor endpoint usage, detect system bottlenecks and quantify error occurrences across different modules. These insights supported both technical and managerial decision-making, allowing for a more efficient allocation of resources and prioritization of software improvements.

In addition, the introduction of the automated pipeline for invoice validation and correction is expected to deliver long-term benefits. By systematically addressing recurring inconsistencies in extracted data, the workflow reduces the burden of manual validation and enables faster, more accurate document processing. This not only improves the scalability of the system but also lays the groundwork for intelligent automation, where feedback from

user interactions can be incorporated to refine correction rules and predictive models. As the pipeline evolves, it has the potential to support adaptive learning mechanisms, improve compliance with business rules and increase overall trust in the automated processing framework.

Moreover, the project introduced a reusable and extensible analytical architecture, establishing a foundation for future performance monitoring initiatives within the organization. It also contributed to a cultural shift toward data-driven evaluation practices, equipping the company with concrete tools to assess and improve the reliability of its document automation systems.

In conclusion, this project successfully bridged the gap identified in the original proposal by delivering a functional, scalable and interpretable solution tailored to the needs of DataCo-lab. Moreover, the additional solution to address the critical issue of requiring user validation for all documents, by automating the detection and correction of frequent inconsistencies. It stands as both a practical and methodological contribution that can serve as a reference for similar analytical challenges in the field of document automation and software performance monitoring.

Bibliography

Alemifar, H. (2025). *Empowering enterprises with lightweight large language models: Automated" rule card" extraction from grant documents (doctoral dissertation, politecnico di torino).*

Apandi, A. H. B., Shariff, A., & Khairai, K. M. (2024). *Product quality output measurement for preventive maintenance on computer numerical control (cnc) machines at an electronic manufacturing industry citation* (Vol. 2024).

Armbrust, M., Ghodsi, A., Xin, R., & Zaharia, M. (2021). Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of cidr* (Vol. 8, p. 28).

Aslan, A., & Çelebi, S. B. (2024). *Strategic advantages of business intelligence applications with olap technology in wholesale companies.* Retrieved from https://www.icfarconf.com/

Asplin, M., & Daneshtalab, M. (2025). *Vertical scaling for big data analytics and processing-a case study examiner: Abu naser masud.*

Becker, L. T., & Gould, E. M. (2019, 7). Microsoft power bi: Extending excel to manipulate, analyze, and visualize diverse data. *Serials Review*, *45*, 184-188. doi: 10.1080/00987913.2019.1644891

Benjelloun, M., El, M., & Amin, E. (2018). Impact of using snowflake schema and bitmap index on data warehouse querying. *International Journal of Computer Applications*, *180*(15), 33–35.

Bessa, J. A., Souza, G., Pessoa, L., Barreto, R., de Freitas, R., et al. (2024). Análise de desempenho de log parsers da coleção logpai em dados brutos de dispositivos android. In *Workshop em desempenho de sistemas computacionais e de comunicação (wperformance)* (pp. 37–48).

Cappelletti, A., & Maglione, S. (2020). Developing log analysis for a worldwide distributed system.

Chai, C. P. (2023). Comparison of text preprocessing methods. *Natural language engineering*, *29*(3), 509–553.

Chen, J. (2024, 1). A study on analyzing symbolism in literature by combining computer vision techniques. *Applied Mathematics and Nonlinear Sciences*, *9*. doi: 10.2478/amns-2024-2521

Cheng, Q., Chen, L., Hu, Z., Tang, J., Xu, Q., & Ning, B. (2024). A novel prompting method for few-shot ner via llms. *Natural Language Processing Journal*, *8*, 100099.

Collier, C. A. (2024). Teaching case cleaning house: A case teaching data cleaning using real-world zillow real estate data. *Journal of Information Systems Education*, *35*, 456-460. doi: 10.62273/OQXC8468

Cormier, K., Kongwen, Zhang, Padron-Uy, J., Wong, A., Gagnier, K., & Parihar, A. (2025, 2). Data warehouse design for multiple source forest inventory management and image processing. Retrieved from http://arxiv.org/abs/2502.07015

Daqqah, B. H. (2024). *Leveraging large language models (llms) for automated extraction and processing of complex ordering forms.*

Das, S., Banerjee, K., Nath, S., & Chatterjee, S. (2023). Data visualization approach for business strategy recommendation using power bi dashboard. Retrieved from www .ijlesjournal.org doi: 10.51386/25816659/ijles-v6i5p101

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87. doi: 10.1145/2347736.2347755

Feldman, R., & Sanger, J. (2007). *The text mining handbook : advanced approaches in analyzing unstructured data*. Cambridge University Press.

Georgiev, A., & Valkanov, V. (2024). Custom data quality mechanism in data warehouse facilitated by data integrity checks. *Mathematics and Education in Mathematics*, *53*, 67-75. doi: 10.55630/mem.2024.53.067-075

Göksel, G., Arslan, A., & Dinçer, B. T. (2023). A selective approach to stemming for minimizing the risk of failure in information retrieval systems. *PeerJ Computer Science*, *9*, e1175.

Gonçalves, M., Salgado, C., de Sousa, A., & Teixeira, L. (2025, 1). Data storytelling and decision-making in seaport operations: A new approach based on business intelligence. *Sustainability*, *17*, 337. Retrieved from https://www.mdpi.com/2071-1050/17/1/337 doi: 10.3390/su17010337

Gruenberg, R. (2022). *Multi-model snowflake schema creation* (Unpublished master's thesis). Miami University.

Gulnar, R. (2023). Comparative analysis of traditional vs cloud data warehouse. *Journal of Computing and Artificial Intelligence*, *1*(1).

Guo, M., Wang, Y., Yang, Q., Li, R., Zhao, Y., Li, C., … Gao, R. (2023, 9). Normal workflow and key strategies for data cleaning toward real-world data: Viewpoint. *Interactive Journal of Medical Research*, *12*, e44310. doi: 10.2196/44310

Gupta, P. (2021). *Practical data science with jupyter explore data cleaning, pre-processing, data wrangling,... feature engineering and machine learning using pyt*. BPB PUBLICATIONS.

Hamza, M. M., Ashraf, A., & Gomaa, W. H. (2024, 9). Unlocking arabic script: Ocr technology for efficient text extraction. In (p. 299-304). Institute of Electrical and Electronics Engineers (IEEE). doi: 10.1109/imsa61967.2024.10652687

Harenslak, B., & Ruiter, J. D. (2021). *Data pipelines with apache airflow*.

Hashem, O. M., Rahouma, K., Abd, N. S., & Hameed, E. (2025). *Enhancing quality factors in data warehousing through study and improvement* (Vol. 44).

Hassle, I., & Bardvall, M. (2024). *Automating invoice recognition: A comparative study of large language models and ocr/ml technologies*.

He, S., Zhu, J., He, P., & Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. In *Proc. ieee international symposium on software reliability engineering (issre)* (pp. 207–218).

Healy, K. (2024). *Data visualization: A practical introduction*. Princeton University Press.

Hua, S., Chen, G., Wei, H., & Jiang, Q. (2012). Similarity measure for image resizing using sift feature. *EURASIP Journal on Image and Video Processing*, *2012*, 1–11.

Inersjö, E. (2021). *Comparing database optimisation techniques in postgresql: Indexes, query writing and the query optimiser*.

Inojosa, H., Voigt, I., Wenk, J., Ferber, D., Wiest, I., Antweiler, D., … Ziemssen, T. (2024, 10). *Integrating large language models in care, research, and education in multiple sclerosis management*. SAGE Publications Ltd. doi: 10.1177/13524585241277376

Jalili, A., Tabrizchi, H., Mosavi, A., & Varkonyi-Koczy, A. (2024). Enhancing language model performance with a novel text preprocessing method. *Acta Physica Polonica: A*, *146*(4).

Karapet, B., Savitskyi, R., & Vakaliuk, T. (2024). Method of comparing and transforming images obtained using uav. *Radioelectronic and Computer Systems*, *2024*(1), 99–115.

Karnani, B. (2025, 2). Achieving high customer data quality: A comprehensive guide. *International Journal of Innovative Research in Science, Engineering and Technology*, *14*. Retrieved from https://www.ijirset.com/upload/2025/february/9_Achieving.pdf doi: 10.15680/IJIRSET.2025.1402009

Khan, A. (2024). *Visual analytics for dashboards*. Apress. doi: 10.1007/979-8-8688-0119-8

Konda, M. (2023). *Elasticsearch in action*.

Kwartler, T. (2017). *Text mining in practice with r*. John Wiley Sons.

Liu, S., Yun, L., Nie, S., Zhang, G., & Li, W. (2024). Iplog: An efficient log parsing method based on few-shot learning. *Electronics*, *13*(16), 3324. doi: 10.3390/electronics13163324

Liu, Y., Li, S., Deng, Y., Hao, S., & Wang, L. (2024, 8). Ssuiebert: Domain adaptation model for chinese space science text mining and information extraction. *Electronics (Switzerland)*, *13*. doi: 10.3390/electronics13152949

Luís, M., Vaz, C., & Francisco, A. P. (2023, 10). *Flowviz: An airflow based workflow middleware for computational phylogenetics*. Retrieved from https://www.preprints.org/manuscript/202310.1211/v1 doi: 10.20944/preprints202310.1211.v1

Mahadevkar, S. V., Patil, S., Kotecha, K., Soong, L. W., & Choudhury, T. (2024, 12). Exploring ai-driven approaches for unstructured document analysis and future horizons. *Journal of Big Data*, *11*. doi: 10.1186/s40537-024-00948-z

Mahdikhani, M., & Meena, P. (2024, 10). Metaverse applications and supply chain innovation: insights from text mining. *Journal of Innovation and Knowledge*, *9*. doi: 10.1016/j.jik.2024.100591

Malashin, I., Masich, I., Tynchenko, V., Gantimurov, A., Nelyub, V., & Borodulin, A. (2024, 6). Image text extraction and natural language processing of unstructured data from medical reports. *Machine Learning and Knowledge Extraction*, *6*, 1361-1377. doi: 10.3390/make6020064

Midway, S. R. (2020, 12). *Principles of effective data visualization* (Vol. 1). Cell Press. doi: 10.1016/j.patter.2020.100141

Mingsamorn, P., & Singhuang, N. (2024). *An interactive dashboard system for budget allocation using power bi* (Vol. 1).

Nesca, M., Katz, A., Leung, C. K., & Lix, L. M. (2022). A scoping review of preprocessing methods for unstructured text data to assess data quality. *International Journal of Population Data Science*, *7*(1), 1757.

Nikita Kathare, O. V. R., & Prabhu, D. V. (2022, 11). A comprehensive study of elastic search. *Journal of Research in Science and Engineering*, *4*. doi: 10.53469/jrse.2022.04(11).07

Obe, R. O., & Hsu, L. S. (2017). *Postgresql: up and running: a practical guide to the advanced open source database*. " O'Reilly Media, Inc.".

Oner, B., Hakli, O., & Zengul, F. D. (2024). A text mining and network analysis of topics and trends in major nursing research journals. *Nursing open*, *11*(1), e2050.

Oyen, D., Kucer, M., & Wohlberg, B. (2021). Vishash: Visual similarity preserving image hashing for diagram retrieval. In *Applications of machine learning 2021* (Vol. 11843, pp. 50–66).

Petraitytė, E. (2024). *Exploring efficient workflow frameworks for data management optimising data management for sponsoring consortium for open access publishing in particle physics*.

Peña, A., Morales, A., Fierrez, J., Ortega-Garcia, J., Puente, I., Cordova, J., & Cordova, G.

(2024, 8). Continuous document layout analysis: Human-in-the-loop ai-based data curation, database, and evaluation in the domain of public affairs. *Information Fusion*, *108*. doi: 10.1016/j.inffus.2024.102398

Pinho, H., & Cavique, L. (2024). *Determination and analysis of kpis in the production flow of a manufacturing environment: A power bi-based approach.*

Qian, K. Y. (2024). *Rental apartment monitoring system a report submitted to.*

Ravshanovich, A. R. (2024). *Psixologiya va sotsiologiya ilmiy jurnali database structure: Postgresql database.*

Rigueira, F., Bernardino, J., & Pedrosa, I. (2020). Extraction of information from log files using python programming and tableau. In *2020 15th iberian conference on information systems and technologies (cisti)* (pp. 1–7).

Rose, B., Claire, B., C, D. C. K. A., Abner, G. J., Ritz, L., T., L. K. C., & J, P. B. J. (2024). Understanding the role of olap and oltp in managing and interpreting student data for seait scholarship system. *International Journal of Scientific and Academic Research*, *04*, 17-28. Retrieved from https://ijsar.net/index.php/ijsar/article/view/142/84 doi: 10.54756/IJSAR.2024.18

Saout, T., Lardeux, F., & Saubion, F. (2024). An overview of data extraction from invoices. *IEEE Access*, *12*, 19872-19886. doi: 10.1109/ACCESS.2024.3360528

Saraswati, N. W. S., Yanti, C. P., Muku, I. D. M. K., & Dewi, D. A. P. R. (2025). Evaluation analysis of the necessity of stemming and lemmatization in text classification. *MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, *24*(2), 321–332.

Saxena, N., & Parveen, H. (2018). *Text extraction systems for printed images: A review.* Retrieved from https://ssrn.com/abstract=3331937

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A systematic literature review on applying crisp-dm process model. In *Procedia computer science* (Vol. 181, p. 526-534). Elsevier B.V. doi: 10.1016/j.procs.2021.01.199

Shamshiri, A., Ryu, K. R., & Park, J. Y. (2024, 2). *Text mining and natural language processing in construction* (Vol. 158). Elsevier B.V. doi: 10.1016/j.autcon.2023.105200

Shioi, T., Kambayashi, T., Arakawa, S., Kurosawa, R., Hikida, S., & Yokota, H. (2024, 9). Read-safe snapshots: An abort/wait-free serializable read method for read-only transactions on mixed oltp/olap workloads. *Information Systems*, *124*. doi: 10.1016/j.is.2024.102385

Souibgui, M., Atigui, F., Zammali, S., Cherfi, S., & Yahia, S. B. (2019). Data quality in etl process: A preliminary study. In *Procedia computer science* (Vol. 159, p. 676-687). Elsevier B.V. doi: 10.1016/j.procs.2019.09.223

Thor, A., Golovin, N., & Rahm, E. (2005). Adaptive website recommendations with awesome. *The VLDB journal*, *14*, 357–372.

Unwin, A. (2020, 1). Why is data visualization important? what is important in data visualization? *Harvard Data Science Review*. doi: 10.1162/99608f92.8ae4d525

Vallelunga, R., Scarpino, I., Martinis, M. C., Luzza, F., & Zucco, C. (2024, 12). Applications of text mining techniques to extract meaningful information from gastroenterology medical reports. *Journal of Computational Science*, *83*. doi: 10.1016/j.jocs.2024.102458

Wang, J., Liu, Y., Li, P., Lin, Z., Sindakis, S., & Aggarwal, S. (2024, 3). Overview of data quality: Examining the dimensions,antecedents, and impacts of data quality. *Journal of the Knowledge Economy*, *15*, 1159-1178. doi: 10.1007/s13132-022-01096-6

Yang, D., Kleissl, J., Gueymard, C. A., Pedro, H. T., & Coimbra, C. F. (2018, 7). History and trends in solar irradiance and pv power forecasting: A preliminary assessment and

review using text mining. *Solar Energy*, *168*, 60-101. doi: 10.1016/j.solener.2017.11.023

Younis, K., & Khateeb, A. (2017). Arabic hand-written character recognition based on deep convolutional neural networks. *Jordanian Journal of Computers and Information Technology*, *3*, 186. doi: 10.5455/jjcit.71-1498142206

Zervou, M. (2024). *Python data cleaning and preparation best practices : a practical guide to organizing and handling data from various sources and formats using python*. Packt Publishing Ltd.

Zhang, Q., Huang, V. S.-J., Wang, B., Zhang, J., Wang, Z., Liang, H., … Zhang, W. (2024, 10). *Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction*. Retrieved from http://arxiv.org/abs/2410.21169

Zhang, X., Bai, W., & Cui, H. (2023). *Review of optical character recognition for power system image based on artificial intelligence algorithm* (Vol. 120). Tech Science Press. doi: 10.32604/ee.2023.020342

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. In *Proc. ieee/acm international conference on software engineering (icse), seip track*.

Zhu, L., & Luo, D. (2023). A novel efficient and effective preprocessing algorithm for text classification. *Journal of Computer and Communications*, *11*(3), 1–14.

Zolbanin, H., & Aubert, B. (2025, 3). *A process model for design-oriented machine learning research in information systems* (Vol. 34). Elsevier B.V. doi: 10.1016/j.jsis.2024.101868

Appendix A

Appendix

A.1 Softwares

This thesis is fundamentally centered on the use of various software and tools, including Elasticsearch, Docker, Apache Airflow, PostgreSQL, Python and Power BI. Each of these technologies plays a significant role in data processing, storage, workflow automation and visualization.

Elasticsearch is a distributed document storage system built on Lucene, primarily used for searching and analyzing large volumes of data. Data is stored as JSON objects, which correspond to rows in relational database tables, enabling fast and efficient queries (Nikita Kathare & Prabhu, 2022). It is part of the Elastic Stack, which also includes Beats (data collectors), Logstash (ETL tool) and Kibana (visualization tool), forming a comprehensive ecosystem for data analysis. During indexing, Elasticsearch automatically interprets and stores the data types of each field, optimizing retrieval (Konda, 2023). Additionally, its horizontal scalability allows it to store petabytes of structured and unstructured data, making it a robust option for data storage and retrieval.

Apache Airflow is a data workflow orchestration platform that enables the definition, scheduling and monitoring of data pipelines. Its structure is based on Directed Acyclic Graphs (DAGs), providing a clear representation of tasks and their dependencies (Harenslak & Ruiter, 2021). Airflow enables automated task execution based on dependencies, workflow monitoring through a web interface, integration with various systems via operators and hooks, implementation of security mechanisms for sensitive data protection, execution of tasks within Docker and Kubernetes containers and the creation of custom components such as hooks and operators (Harenslak & Ruiter, 2021; Petraitytė, 2024)

Docker is widely used in conjunction with Apache Airflow to mitigate challenges related to dependency management. It allows workflows to run in isolated containers, ensuring consistency across development and production environments (Petraitytė, 2024). Airflow supports task execution using the DockerOperator, which facilitates the creation of dynamic and flexible DAGs (Luís, Vaz, & Francisco, 2023). Thus, the combined use of Airflow and Docker enhances scalability and reproducibility in data workflow management.

PostgreSQL is an open-source relational database management system (RDBMS) widely used for its reliability, scalability and support for SQL specifications. Its extensibility allows users to customize data types, functions and indexes according to project needs. Additionally, PostgreSQL offers one of the best supports for SQL standards, ensuring flexibility in data

manipulation. Through SQL commands, data can be added, modified and queried efficiently. Its compatibility with various tools, including Apache Airflow and Elasticsearch, makes it an ideal choice for managing and analyzing large volumes of data (Ravshanovich, 2024).

As noted by Qian (2024), Python is widely recognized as a powerful tool for data processing and manipulation, playing a critical role in tasks such as data cleaning, handling missing values and converting data types. The Pandas library, in particular, is a valuable resource for loading, manipulating and transforming data.

An additional advantage of Python is its capability to automate data preprocessing workflows through pipelines provided by libraries such as Scikit-learn, making the process more efficient and reliable. Once preprocessing is complete, the data can be transformed and formatted to facilitate further analyses. This includes integration with tools like Power BI, which offers advanced functionalities for data visualization and exploration. Such visualizations are crucial for understanding relationships between different variables, fostering deeper and more informed insights (Qian, 2024).

On the other hand, Microsoft Power BI is widely recognized as one of the leading business intelligence tools for creating dashboards. It is extensively used to integrate, analyze and visualize data, transforming raw information into actionable insights (Gonçalves et al., 2025). Power BI allows for the development of interactive and visually engaging dashboards, which simplify the interpretation of complex data and make information accessible to a wide range of users. These dashboards effectively present key performance indicators (KPIs) in a clear and concise manner, ensuring alignment with project objectives and enabling the evaluation of specific model performance (Apandi, Shariff, & Khairai, 2024; Pinho & Cavique, 2024).

A key advantage of Power BI is its ability to integrate data from multiple sources, such as SQL databases, spreadsheets, ERP systems, Jupyter Notebooks and real-time data streams. This provides a comprehensive and consolidated view of information (Apandi et al., 2024; Mingsamorn & Singhuang, 2024). The tool also excels in data modeling, organizing data into fact and dimension tables to facilitate queries and reporting. Moreover, its DAX (Data Analysis Expressions) language supports the creation of custom calculations, enabling deeper insights (Pinho & Cavique, 2024). This combination of integration, modeling and customization makes Power BI highly versatile.

Interactive visualization is another cornerstone of Power BI's functionality. By transforming complex data into intuitive and visually appealing formats such as charts and tables, Power BI enables the identification of patterns, trend analysis and comparisons across categories (Mingsamorn & Singhuang, 2024). Dashboards include features like interactive filters and drill-through capabilities, empowering users to explore detailed information as needed (Pinho & Cavique, 2024). Additionally, real-time monitoring features enhance operational efficiency by focusing on significant deviations from performance standards, enabling effective exception management (Apandi et al., 2024).

A.2 Main Database Structure

Projects
creat by
creation Date
creation Time
deleted [bolean]
update at
update by
id integration
name
tenant
id

Documents
id
Document Name
Size
Nº Pages
Nº Invoices
Nº create Invoices
Project id
Tenant
Document Error
Document status
Deleted
Creat at
Update at
Creat by
Update by

Invoice_raw
id
Nº of Invoice
Billing date
NIF Supplier
Supplier Country
Supplier name
NIF Customer
Customer Country
Customer street
Customer zip code
Customer City
Cost
Cost with taxes
Currencý
Description
Invoice Type
LLM model
Input Tokens
Output Tokens
Total time
LLM Time
Asserts Time

Invoice
id
Nº of Invoice
Billing date
NIF Supplier
Supplier Country
Supplier name
NIF Customer
Customer Country
Customer street
Customer zip code
Customer City
Cost
Cost with taxes
Currency
Description
Invoice Type
Validation Status
Update by
Create by
Update Time
Update Date
Creat Time
Creat Date
Deleted
Tenant
Document id
Warning Business
Warning Fields

A.3 Example of a Log Entry

2025-02-21 15:25:55 - app.main - INFO - [REQ_20ac...] msg: Request started | method: GET | path: /docs | ip:

INFO: - *"GET /docs HTTP/1.1" 200 OK*

WHERE user_account.idkeycloak = %(idkeycloak_1)s

2025-02-21 16:48:44,273 INFO sqlalchemy.engine.Engine [cached since 0.2267s ago] {'idkeycloak_1': 'aedc1,,,,,,'}

2025-02-24 10:27:52 - app.src_llm.media.media_main - INFO - [DOC_a1cc.....] msg: Starting process document pdf...

A.4 Example of the pre-processing applied to a request log

2025-02-21 15:43:56 - app.main - INFO - [REQ_7993d158-d127-44d7-98d6-ce489172029f] msg: Request started | method: GET | path: /docs | ip:

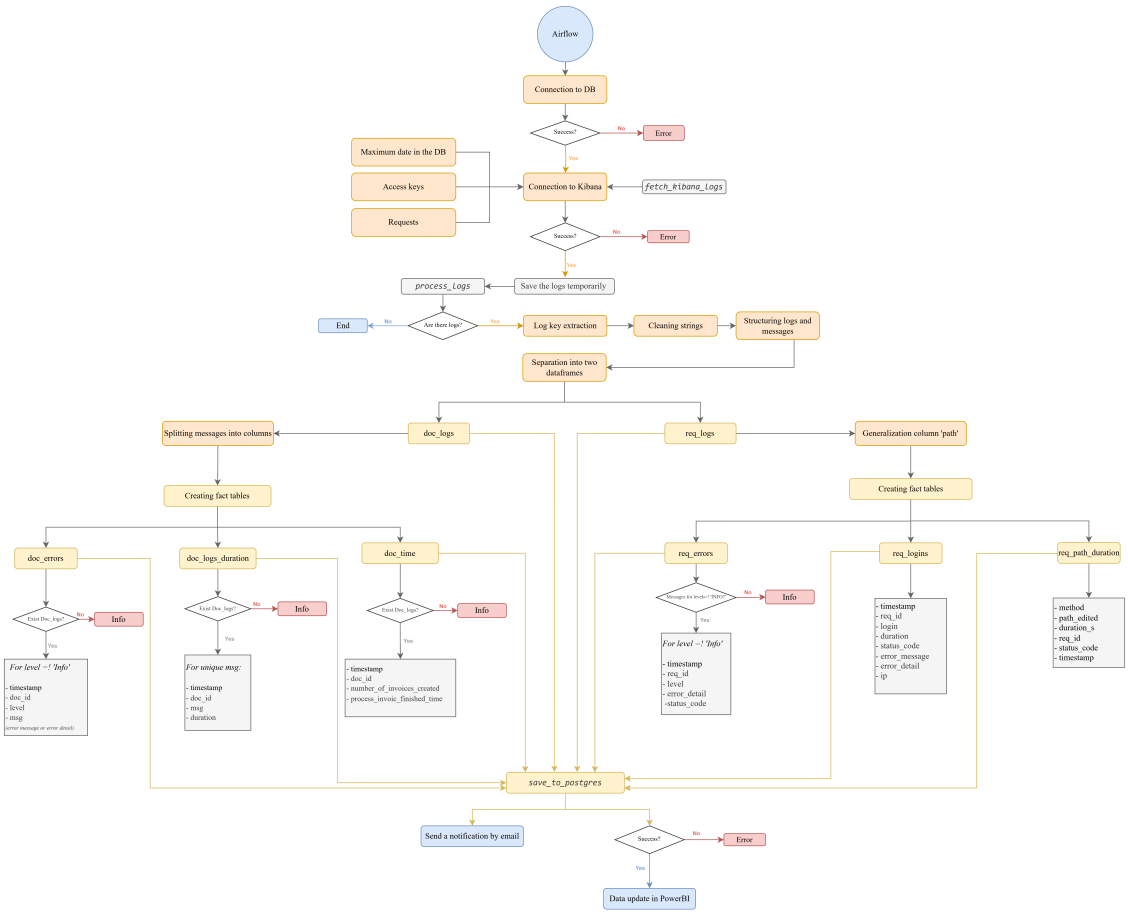
Timestamp:	2025-02-21 15:43:56
Module:	app.main
Level:	INFO
ID	[REQ_7993d158-d127-44d7-98d6-ce489172029f]
msg:	Request started
method:	GET
path:	/docs
ip:	

A.5 Example of the pre-processing applied to a document log

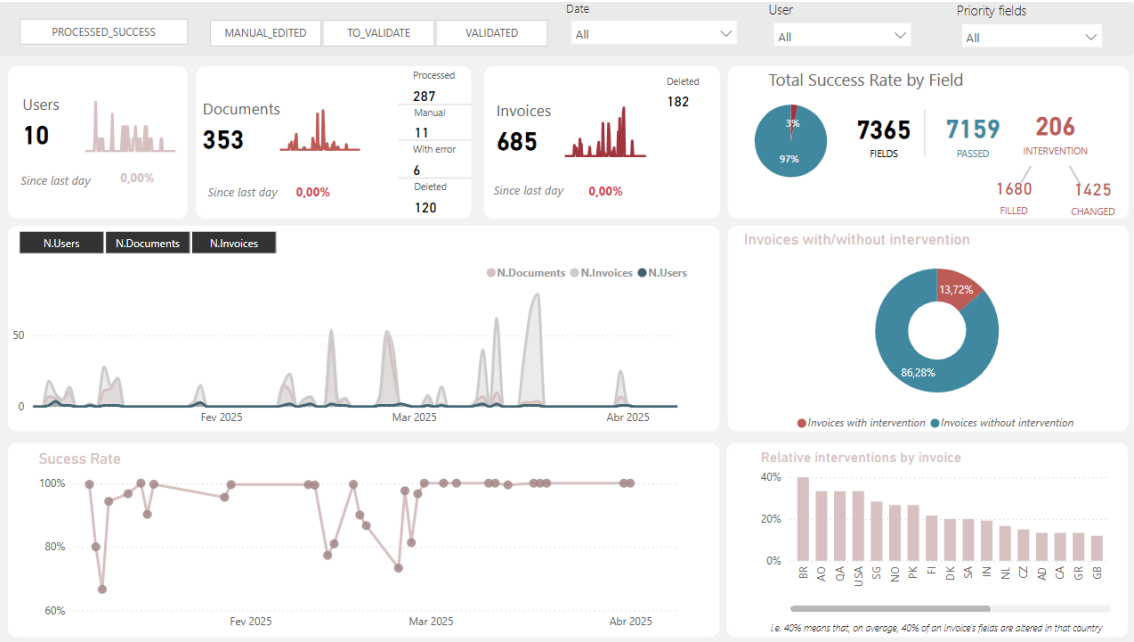
```
2025-02-24 10:28:11 - app.src_llm.main_processer - INFO - [DOC_a1cc0e59-6a81-4fde-b838-866a015f45b8] msg: Number of invoices created - 2
```

Timestamp:	2025-02-24 10:28:11
Module:	app.src_llm.main_processer
Level:	INFO
ID	[DOC_a1cc0e59-6a81-4fde-b838-866a015f45b8]
msg:	
number_of_invoices_created:	2

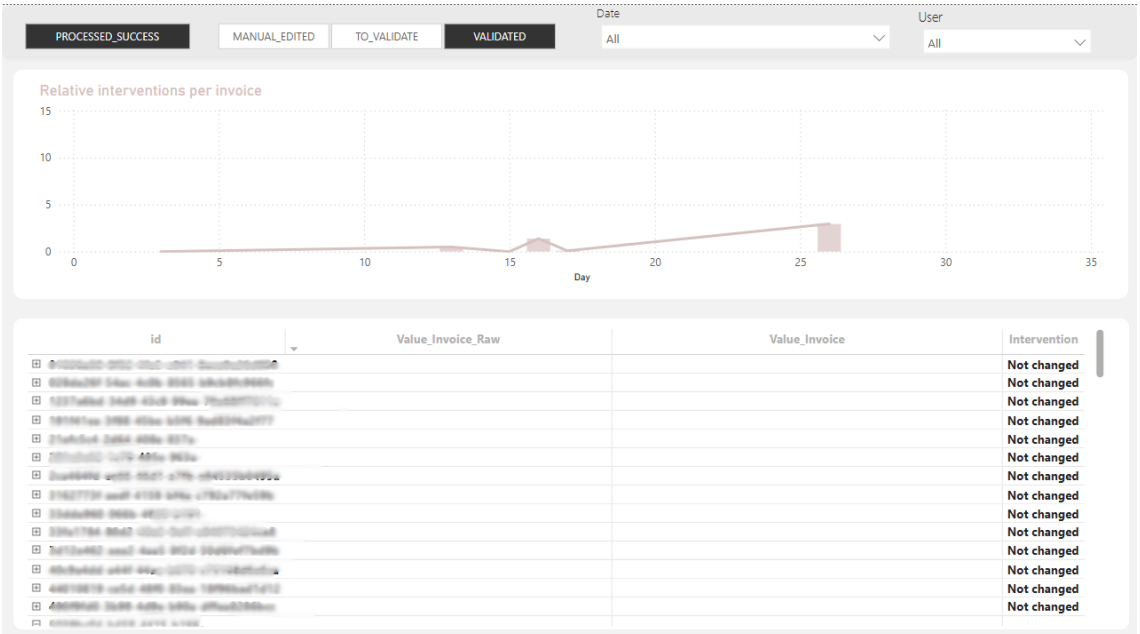
A.6 Detailed diagram of the low-level pre-processing pipeline



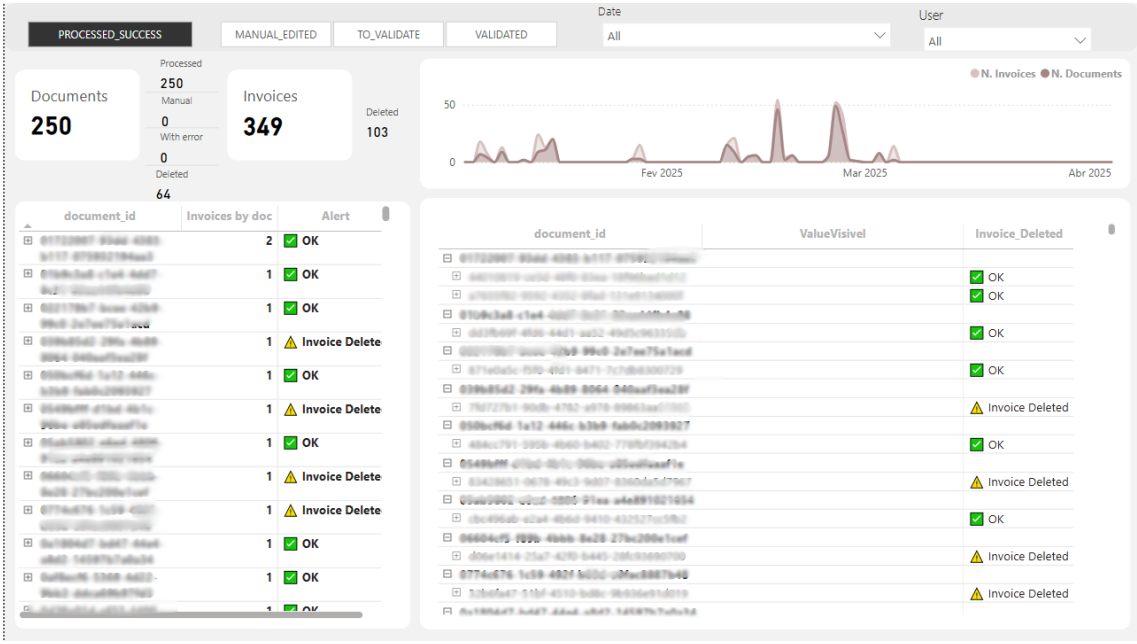
A.7 Overview Page of Invoice Extraction Quality Dashboard



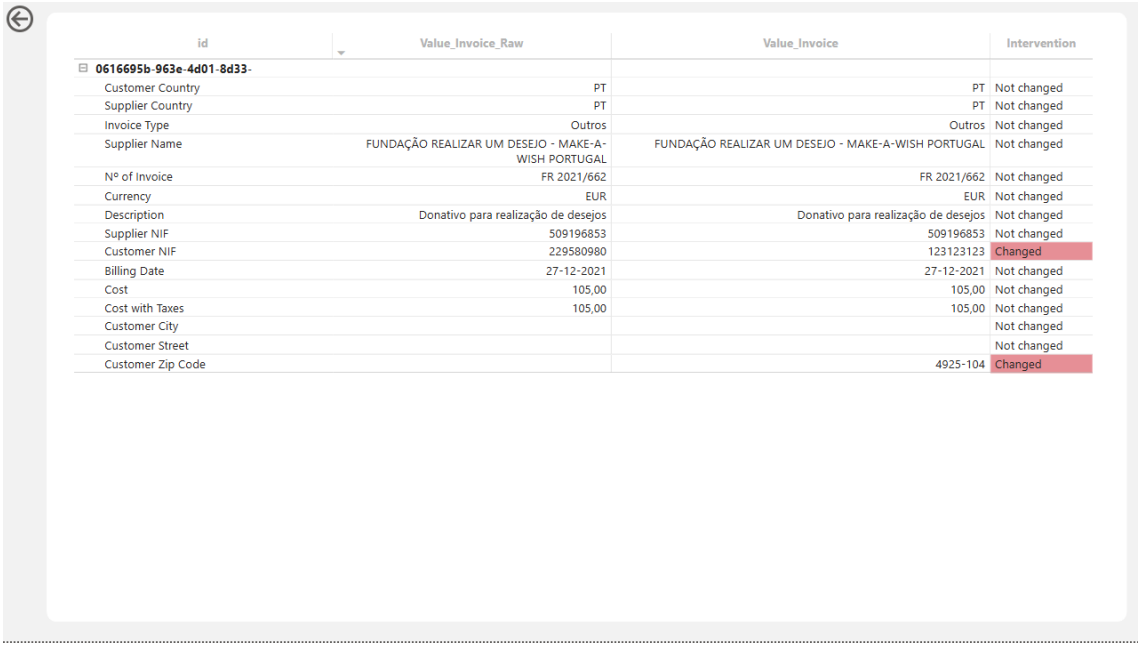
A.8 Data Level Page of Invoice Extraction Quality Dashboard



A.9 Documents vs Invoices Page of Invoice Extraction Quality Dashboard

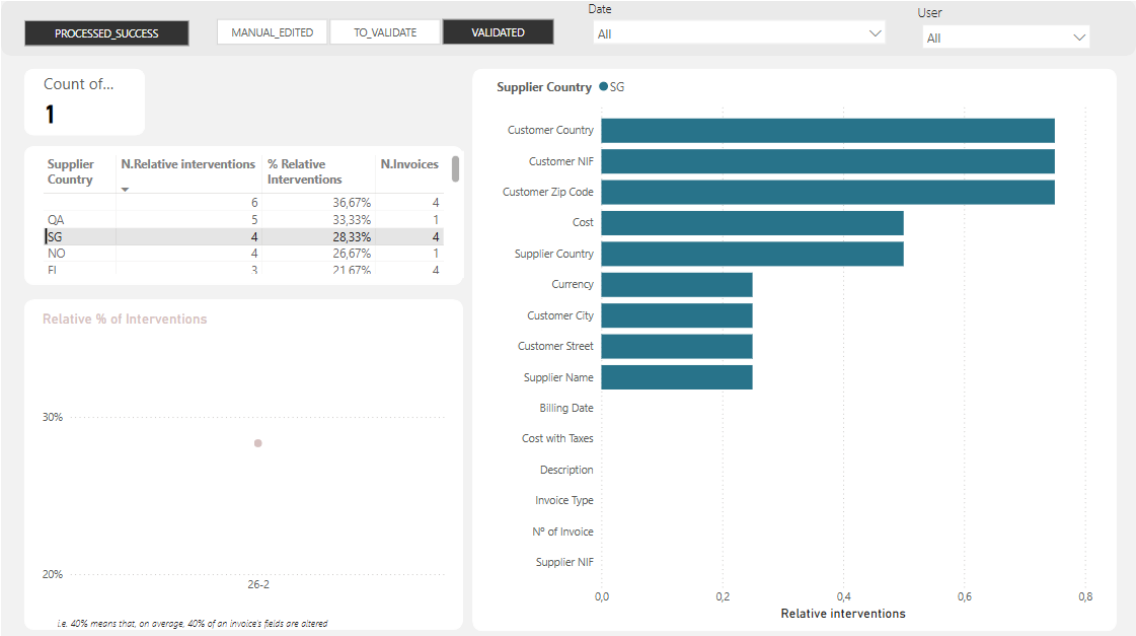


A.10 Detail Page Example of Invoice Extraction Quality Dashboard

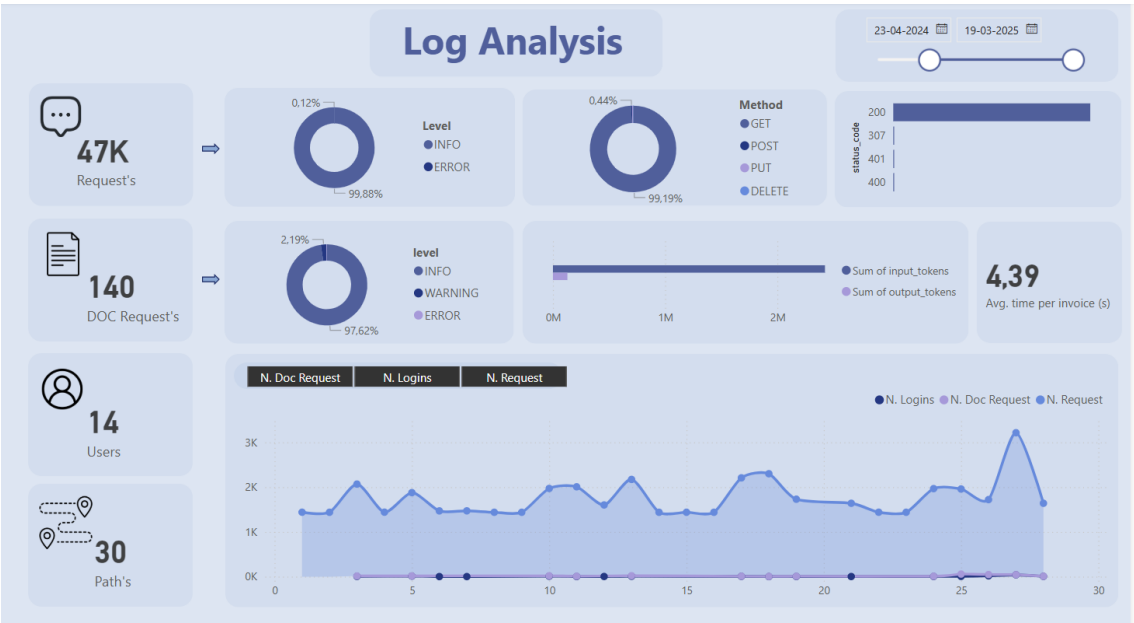


id	Value_Invoice_Raw	Value_Invoice	Intervention
0616695b-963e-4d01-8d33-			
Customer Country	PT	PT	Not changed
Supplier Country	PT	PT	Not changed
Invoice Type	Outros	Outros	Not changed
Supplier Name	FUNDAÇÃO REALIZAR UM DESEJO - MAKE-A-WISH PORTUGAL	FUNDAÇÃO REALIZAR UM DESEJO - MAKE-A-WISH PORTUGAL	Not changed
Nº of Invoice	FR 2021/662	FR 2021/662	Not changed
Currency	EUR	EUR	Not changed
Description	Donativo para realização de desejos	Donativo para realização de desejos	Not changed
Supplier NIF	509196853	509196853	Not changed
Customer NIF	229580980	123123123	Changed
Billing Date	27-12-2021	27-12-2021	Not changed
Cost	105,00	105,00	Not changed
Cost with Taxes	105,00	105,00	Not changed
Customer City			Not changed
Customer Street			Not changed
Customer Zip Code		4925-104	Changed

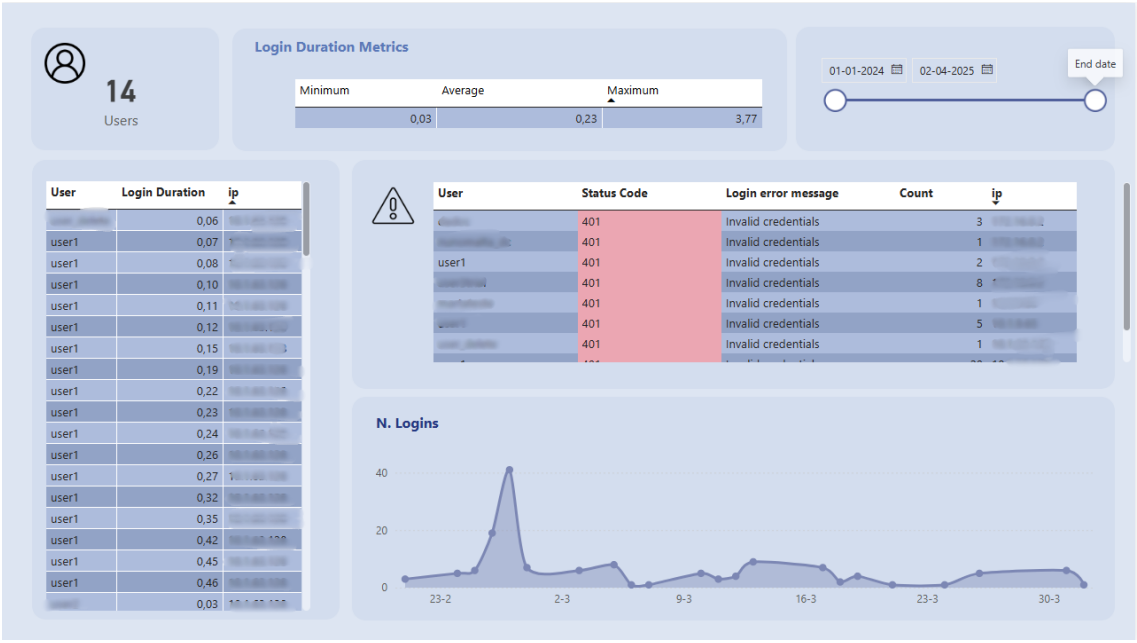
A.11 Country Level page with drill-through to Singapore in the Invoice Extraction Quality Dashboard.



A.12 Overview Page of Log Monitoring Dashboard



A.13 Login Page of Log Monitoring Dashboard



A.14 Documents Process Time Page of Log Monitoring Dashboard