GRADO EN INGENIERÍA MATEMÁTICA CON INTELIGENCIA ARTIFICIAL

TRABAJO DE FIN DE GRADO

Business Process Automation Using Artificial Intelligence at Foreworth

Autor: Hugo Albert Rodríguez de Miguel

Director: Álvaro Marinetto Sánchez

Madrid

Mayo de 2026

Declaración de originalidad

Declaro bajo mi responsabilidad que el Proyecto presentado con el título **Business Process Automation Using Artificial Intelligence at Foreworth** e la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico 2025/2026 es de mi autoría y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Uso de Inteligencia Artificial1

Declaro bajo mi responsabilidad que (indicar la opción correcta):

□ No he utilizado Inteligencia Artificial en la elaboración del presente documento.

⊗ He utilizado Inteligencia Artificial en la elaboración del presente documento y/o del Anexo B siempre en las condiciones permitidas por la Universidad Pontificia Comillas, es decir, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024): "*La IA puede utilizarse para actividades previas a la tarea, como la lluvia de ideas, la descripción y la investigación inicial. Este nivel se centra en el uso de la IA para la planificación, las síntesis y la generación de ideas, pero las evaluaciones deben hacer hincapié en la capacidad de desarrollar y refinar estas ideas de forma independiente*". En concreto, las Inteligencia Artificial ha sido empleada para:

(indicar aquí el uso concreto que se ha hecho de la Inteligencia Artificial)
He utilizado la Inteligencia Artificial como herramienta de apoyo en fases previas y auxiliares del proyecto. En concreto, la he empleado para organizar ideas iniciales, plantear posibles enfoques de solución, comparar alternativas, estructurar el trabajo, y revisar la claridad de algunas explicaciones. También la he utilizado como apoyo para la búsqueda de palabras en inglés. En todos los casos, he revisado, adaptado y validado el contenido generado. El análisis del problema, el diseño final de las soluciones, la implementación técnica, la experimentación, la obtención de resultados, la interpretación de métricas y las conclusiones han sido realizados por mí.

Firmado (alumno): Hugo Albert Rodríguez de Miguel
Fecha: 24 de mayo de 2026

Autorización para la entrega del Proyecto

[1] Esta declaración se refiere al uso de la Inteligencia Artificial generativa para realizar los documentos del Proyecto (Anexo B y Memoria). No aplica a Proyectos donde, por su naturaleza, deban emplear inteligencia artificial como parte de los mismos (aplicación de técnicas de aprendizaje automático, redes neuronales, análisis de datos…)

El Director del Proyecto	El co-Director del Proyecto (si aplica)
	
Fdo: Álvaro Marinneto Sánchez	Fdo:
Fecha: 24 de mayo de 2026	Fecha:

BACHELOR OF MATHEMATICAL ENGINEERING AND ARTIFICIAL INTELLIGENCE

BACHELOR'S FINAL PROJECT

Business Process Automation Using Artificial Intelligence at Foreworth

Author: Hugo Albert Rodríguez de Miguel

Director: Álvaro Marinetto Sánchez

Madrid

May 2026

Acknowledgements

I would like to express my gratitude to Foreworth for giving me the opportunity to carry out this project as part of my internship. Working on real production systems, with real data and real constraints, shaped the way I approached every technical problem throughout this thesis.

In particular, I want to thank Álvaro Marinetto for his supervision and guidance. His availability during meetings, his practical feedback, and his willingness to discuss both technical and operational aspects of the work made a genuine difference. The direction of the project benefited greatly from those conversations.

I also want to thank the rest of the Foreworth team for their support during my time there. The working environment was collaborative and the day-to-day interaction with the team helped me understand how engineering decisions actually get made in a production context.

I would also like to thank Rogelio Martínez Perea, my academic supervisor at Universidad Pontificia Comillas, for his guidance throughout the project from a university perspective. His feedback helped ensure the work met the academic standards expected of a final degree thesis.

AUTOMATIZACIÓN DE PROCESOS EMPRESARIALES A TRAVÉS DE LA INTELIGENCIA ARTIFICIAL EN FOREWORTH

Autor: Albert Rodríguez de Miguel, Hugo.
Director: Marinetto Sánchez, Álvaro.
Entidad Colaboradora: Foreworth

RESUMEN DEL PROYECTO

Este trabajo aborda la automatización de dos procesos analíticos clave en Foreworth, una plataforma de inteligencia de software que analiza repositorios Git para generar métricas de productividad. El primer proceso consiste en la resolución automática de identidades de desarrollador, unificando los distintos alias que un mismo desarrollador puede usar en Git. El segundo consiste en la generación automatizada de informes de rendimiento mediante modelos de lenguaje de gran escala. Ambos procesos buscan reducir el trabajo manual, mejorar la escalabilidad y aumentar la fiabilidad de los resultados entregados a los clientes.

Palabras clave: Jaro-Winkler, Random Forest, LLM, prompt engineering, identidad de desarrollador, generación de informes

1. Introducción

Foreworth es una plataforma de análisis de software que extrae datos de repositorios Git para generar métricas objetivas sobre productividad y calidad del código. La fiabilidad de estas métricas depende directamente de la calidad del proceso de tratamiento de datos, lo que convierte la automatización de ciertos procesos analíticos en una necesidad operativa.

Un desarrollador puede contribuir bajo múltiples identidades: distintas direcciones de correo, nombres de usuario o alias. Si estas identidades no se unifican correctamente, las métricas quedan fragmentadas y las conclusiones sobre rendimiento individual resultan incorrectas. Por otro lado, la generación manual de informes analíticos para clientes requiere un esfuerzo considerable por parte de los consultores, limitando la capacidad de escalar el servicio.

2. Objetivos

El objetivo principal es automatizar estos dos procesos mediante técnicas de aprendizaje automático e inteligencia artificial generativa. En el mapeo de identidades, se busca aumentar la tasa de resolución automática manteniendo precisión elevada, priorizándola sobre el recall dado que una fusión incorrecta corrompe directamente las métricas de productividad. Se comparan un enfoque heurístico basado en Jaro-Winkler con modelos supervisados como Random Forest, XGBoost y LightGBM, y aproximaciones con redes Siamesas. En la generación de informes, el objetivo es producir informes coherentes en español a partir de datos estructurados, directamente entregables a clientes con supervisión mínima.

3. Descripción del modelo/sistema/herramienta

Ambos procesos comparten una arquitectura de datos común. Los datos en bruto procedentes de los repositorios de los clientes se almacenan en Amazon S3 y son consultables mediante Amazon Athena. Los datos procesados se almacenan en bases de

datos MySQL alojadas en local, que actúan como fuente principal para las aplicaciones de análisis.

Para el mapeo de identidades se ha diseñado un marco de decisión en tres zonas: pares con puntuación alta se fusionan automáticamente, pares intermedios se envían a revisión manual, y pares con puntuación baja se descartan. El sistema utiliza una versión mejorada de Jaro-Winkler que amplía las comparaciones incluyendo cruce entre nombre y alias en ambas direcciones, normalización de nombres en formato «apellido, nombre», detección de cuentas no humanas y un umbral dinámico que se adapta a la longitud de las cadenas. En paralelo se entrenaron modelos supervisados con características de similitud textual y variables de comportamiento. Las redes Siamesas con capas LSTM no alcanzaron consistencia suficiente en datos no vistos y fueron descartadas.

Para la generación de informes se construyó un pipeline en dos etapas: primero cada dimensión de datos se resume individualmente mediante un prompt dedicado, y después los resúmenes se consolidan en un informe final. La interfaz en Next.js permite al consultor seleccionar tenant, tipo de informe y modelo LLM disponible a través de Amazon Bedrock. El backend está implementado en C#, coherente con la arquitectura de producción de Foreworth.

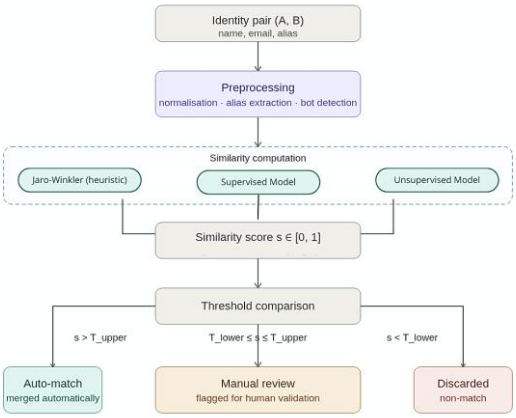

Figura 1. Estructura del mapeo de identidades (Fig. 2 en TFG)

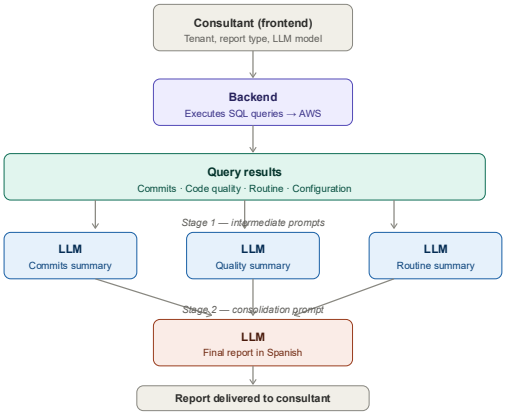

Figura 2. Estructura de la generación de informes (Fig. 4 en TFG)

4. Resultados

El enfoque heurístico mejorado alcanzó una precisión de 0,97 y un recall de 0,72 sobre datos etiquetados de tenants reales, superando al sistema original. Dentro de los modelos supervisados, Random Forest obtuvo mejores resultados en test. Entre estos dos, en la comparación sobre tenants no vistos, Jaro-Winkler mejorado obtuvo una precisión de 0,97 frente al 0,94 del Random Forest, aunque este último alcanzó un recall superior (0,81 frente a 0,72). El análisis de importancia de variables confirmó que las puntuaciones de Jaro-Winkler dominan la predicción, lo que explica que la diferencia entre ambos enfoques sea relativamente pequeña. El impacto operativo fue significativo: el esfuerzo de validación manual durante la incorporación de nuevos clientes se redujo aproximadamente un 75%, de cuatro días de trabajo a uno.

Model	Accuracy	Precision	Recall	F1-score
Enhanced Jaro-Winkler	0.9931	0.97	0.72	0.83
Random Forest	0.9948	0.94	0.81	0.87

Tabla 1. Comparación de métricas entre el enfoque Jaro-Winkler mejorado y Random Forest sobre tenants no vistos (Table 9 en TFG)

En la generación de informes, los consultores no pudieron distinguir consistentemente entre informes automáticos y manuales en comparaciones ciegas. El tiempo de producción se redujo de una media de 45 minutos a menos de dos, una reducción superior al 95%.

5. Conclusiones

Los dos sistemas desarrollados demuestran que una automatización bien diseñada y orientada a producción puede generar un valor operativo medible sin necesidad de recurrir a las técnicas más complejas disponibles. En el caso del mapeo de identidades, el enfoque heurístico mejorado fue seleccionado para producción por su mayor precisión en datos no vistos, su ausencia de requisitos de reentrenamiento y su menor riesgo operativo. Esta decisión refleja un principio que atraviesa todo el proyecto: un modelo más complejo no es siempre una mejor solución en producción. En el caso de la generación de informes, el pipeline de dos etapas con prompts estructurados permitió eliminar el cuello de botella que antes limitaba la entrega de informes a escala.

Como trabajo futuro, la integración del modelo Random Forest en producción es la extensión natural más inmediata, una vez que se disponga de suficiente experiencia operativa y datos de nuevos tenants. Para la generación de informes, la dirección más relevante es la expansión del catálogo de tipos de informe, posible sin cambios en el código gracias al diseño modular del sistema.

6. Referencias

[1] T. Fry, T. Dey, A. Karnauch, and A. Mockus, "A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits," in *Proc. 17th Int. Conf. Mining Software Repositories (MSR '20)*, Seoul, Republic of Korea, Oct. 2020. [Online]. Available: https://arxiv.org/abs/2003.08349

[2] S. Amreen, A. Mockus, C. Bogart, Y. Zhang, and R. Zaretzki, "ALFAA: Active Learning Fingerprint Based Anti-Aliasing for Correcting Developer Identity Errors in Version Control Data," *Empirical Software Engineering*, vol. 25, pp. 1136–1167, 2020. [Online]. Available: https://arxiv.org/pdf/1901.03363

[3] S. Schulhoff et al., "The Prompt Report: A Systematic Survey of Prompting Techniques," arXiv:2406.06608, 2024. [Online]. Available: https://arxiv.org/abs/2406.06608

[4] E. Fons, E. Kochkina, R. Kaur, Z. Zeng, B. Hlavaty, C. Smiley, S. Vyetrenko, and M. Veloso, "AI Analyst: Framework and Comprehensive Evaluation of Large Language Models for Financial Time Series Report Generation," arXiv:2507.00718, 2025. [Online]. Available: https://arxiv.org/pdf/2507.00718

[5] S. Dhuliawala et al., "Chain-of-Verification Reduces Hallucination in Large Language Models," arXiv:2309.11495, 2023. [Online]. Available: https://arxiv.org/abs/2309.11495

BUSINESS PROCESS AUTOMATION USING ARTIFICIAL INTELLIGENCE AT FOREWORTH

Author: Albert Rodríguez de Miguel, Hugo.
Director: Marinetto Sánchez, Álvaro.
Collaborating Entity: Foreworth

ABSTRACT

This project addresses the automation of two analytical processes at Foreworth, a software intelligence platform that analyses Git repository data to generate developer productivity metrics. The first process focuses on developer identity mapping, consolidating the multiple Git aliases a single developer may use into a unified profile. The second focuses on automated report generation using Large Language Models. Both processes aim to reduce manual effort, improve scalability, and increase the reliability of the insights delivered to clients.

Keywords: Jaro-Winkler, Random Forest, LLM, prompt engineering, developer identity, report generation

1. Introduction

Foreworth is a software analytics platform that processes data from client Git repositories to generate objective metrics related to developer productivity and code quality. The reliability of these metrics depends directly on the quality of the underlying data processing pipeline, which makes the automation of certain analytical tasks an operational necessity rather than a purely technical exercise.

A developer may contribute to repositories under multiple identities: different email addresses, usernames, or naming conventions. If these are not correctly unified, productivity metrics become fragmented and conclusions about individual performance are distorted. At the same time, producing analytical reports for clients manually requires significant consultant effort and limits the platform's ability to scale across a growing number of clients and developers.

2. Objectives

The main objective of this project is to automate both processes through the application of machine learning and generative artificial intelligence. For identity mapping, the goal is to increase the automatic resolution rate while maintaining high precision, which is prioritised over recall given that an incorrect merge directly corrupts downstream productivity metrics. The project compares an enhanced Jaro-Winkler heuristic with supervised models including Random Forest, XGBoost, and LightGBM, as well as embedding-based approaches using Siamese networks. For report generation, the objective is to produce coherent Spanish-language reports from structured data that can be delivered to clients with minimal post-processing.

3. Description of the proposed model/system/tool

For identity mapping, the solution is structured around a three-zone decision framework: high-confidence pairs are automatically merged, intermediate pairs are sent to a manual

review queue, and low-confidence pairs are discarded. The core method is an enhanced version of Jaro-Winkler similarity that extends the original system by comparing names and email aliases in both cross-field directions, normalising names written in last-name-first format, detecting non-human accounts, and applying a dynamic similarity threshold that tightens for shorter strings to reduce spurious matches. Supervised models were also trained using the four Jaro-Winkler scores alongside behavioural features such as commit time, timezone offset, and most common programming language. Siamese LSTM networks were explored but did not generalise consistently to unseen tenants and were not selected for production.

For report generation, a two-stage pipeline was built on top of Amazon Bedrock. In the first stage, each data dimension is summarised independently using a dedicated prompt. In the second stage, the intermediate summaries are consolidated into a single coherent report. A Next.js frontend allows consultants to select the tenant, report type, and LLM model, while the backend logic is implemented in C#, consistent with Foreworth's existing production architecture.

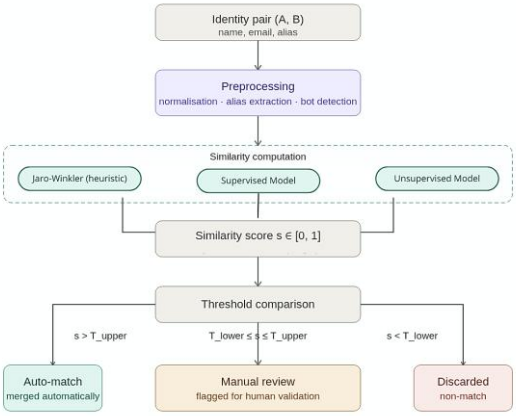

Figure 1. Framework of identity mapping

(Fig. 2 in TFG)

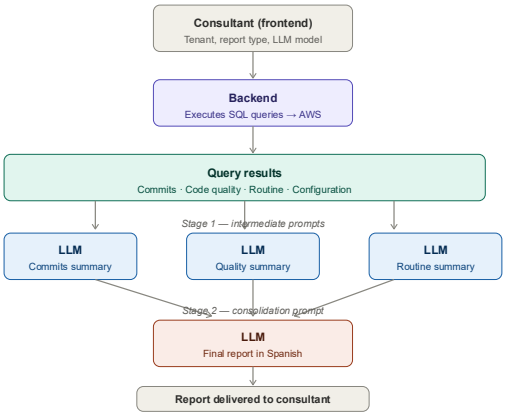

Figure 2. Pipeline of report generation

(Fig. 4 in TFG)

4. Results

The enhanced Jaro-Winkler approach achieved a precision of 0.97 and a recall of 0.72 on labelled data from real production tenants, improving the original system. Corresponding the supervised models, Random Forest achieved the best results on test. On unseen tenants, the heuristic maintained a precision of 0.97 against 0.94 for Random Forest. Feature importance analysis confirmed that the Jaro-Winkler similarity scores dominate the model's predictions, which explains why the performance gap between the two approaches is relatively narrow: both are primarily drawing on the same underlying signal. The operational impact was substantial, reducing manual mapping effort during client onboarding by approximately 75%, from around four working days to one.

Model	Accuracy	Precision	Recall	F1-score
Enhanced Jaro-Winkler	0.9931	0.97	0.72	0.83
Random Forest	0.9948	0.94	0.81	0.87

Table 1. Metric comparison between the enhanced Jaro-Winkler approach and Random Forest on unseen tenants (Table 9 in TFG)

For the report generation process, consultants were unable to consistently distinguish between automatically generated and manually written reports in blind comparisons. Report production time decreased from an estimated average of 45 minutes to under two minutes, a reduction of over 95%.

5. Conclusions

The two systems developed demonstrate that well-designed, production-oriented automation can generate measurable operational value without resorting to the most complex techniques available. In the case of identity mapping, the improved heuristic approach was selected for production due to its higher precision on unseen data, its lack of retraining requirements, and its lower operational risk. This decision reflects a principle that runs throughout the entire project: a more complex model is not always a better solution in production. In the case of report generation, the two-stage pipeline with structured prompts made it possible to eliminate the bottleneck that had previously limited report delivery at scale.

As future work, integrating the Random Forest model into production is the most immediate natural extension, once sufficient operational experience and data from new tenants are available. For report generation, the most relevant direction is expanding the catalog of report types, which is possible without any code changes thanks to the system's modular design.

6. References

[1] T. Fry, T. Dey, A. Karnauch, and A. Mockus, "A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits," in *Proc. 17th Int. Conf. Mining Software Repositories (MSR '20)*, Seoul, Republic of Korea, Oct. 2020. [Online]. Available: https://arxiv.org/abs/2003.08349

[2] S. Amreen, A. Mockus, C. Bogart, Y. Zhang, and R. Zaretzki, "ALFAA: Active Learning Fingerprint Based Anti-Aliasing for Correcting Developer Identity Errors in Version Control Data," *Empirical Software Engineering*, vol. 25, pp. 1136–1167, 2020. [Online]. Available: https://arxiv.org/pdf/1901.03363

[3] S. Schulhoff et al., "The Prompt Report: A Systematic Survey of Prompting Techniques," arXiv:2406.06608, 2024. [Online]. Available: https://arxiv.org/abs/2406.06608

[4] E. Fons, E. Kochkina, R. Kaur, Z. Zeng, B. Hlavaty, C. Smiley, S. Vyetrenko, and M. Veloso, "AI Analyst: Framework and Comprehensive Evaluation of Large Language Models for Financial Time Series Report Generation," arXiv:2507.00718, 2025. [Online]. Available: https://arxiv.org/pdf/2507.00718

[5] S. Dhuliawala et al., "Chain-of-Verification Reduces Hallucination in Large Language Models," arXiv:2309.11495, 2023. [Online]. Available: https://arxiv.org/abs/2309.11495

Memory Index

Figure Index

Table Index

CHAPTER 1. INTRODUCTION

1.1 CONTEXT AND MOTIVATION

In recent years, organizations have increasingly relied on data-driven approaches to improve their internal processes and optimize decision-making. In the context of software development, this trend has led to a growing interest in tools capable of analyzing engineering activity, measuring productivity, and identifying opportunities for improvement. The automation of business processes using advanced data analysis, machine learning, and artificial intelligence has therefore become a key factor for scalability and operational efficiency.

Foreworth is a software intelligence platform that supports organizations by analyzing data from Git repositories to generate objective metrics related to developer productivity and software quality. These metrics allow companies to gain visibility into their development processes, improve resource allocation, and make informed strategic decisions. However, the generation of reliable insights depends not only on accurate data processing but also on the effective automation of key analytical and reporting tasks.

This project addresses two complementary challenges within Foreworth's platform.

The first concerns the optimization of developer identity mapping. A major challenge arises from the fact that developers often contribute to code under multiple identities, such as different email addresses, usernames, or naming conventions. If these identities are not properly unified, the resulting metrics may be distorted, leading to inaccurate conclusions about developer productivity. The existing process was already partially automated but still required extensive manual validation, which becomes a significant operational bottleneck as the number of analyzed repositories and contributors grows.

The second challenge concerns the generation of analytical reports for clients. These reports summarize developer performance and highlight areas for improvement, but producing them manually limits the platform's ability to scale and respond quickly to client needs, especially when customization is required.

Together, these two processes represent critical opportunities to improve both the analytical foundation of the platform and the efficiency with which insights are delivered to clients.

1.2 OBJECTIVES

The main objective of this project is to analyze and implement the automation of key business processes at Foreworth through the application of advanced data analysis

techniques, algorithmic optimization, machine learning models, and generative artificial intelligence. More specifically, the project pursues the following objectives:

First, to optimize the developer identity mapping process by improving the existing system's ability to correctly unify multiple Git identities belonging to the same individual. The goal is to increase the automatic matching rate while maintaining high precision and reducing the number of cases that require manual validation.

Second, to design and implement an automated report generation system based on Large Language Models, capable of transforming structured analytical data into coherent, natural-language reports summarizing developer performance and identifying areas for improvement.

Third, to evaluate the impact of both solutions on operational efficiency and result quality, ensuring that the proposed approaches are suitable for integration into Foreworth's production environment.

1.3 ALIGNMENT WITH THE SDGS

This project is primarily aligned with **Sustainable Development Goal 9: Industry, Innovation and Infrastructure**, as it focuses on the application of advanced technologies to improve the efficiency and scalability of digital infrastructures within organizations. By leveraging data analysis techniques, machine learning models, and generative artificial intelligence, the project contributes to the development of more innovative and resilient software analytics systems. The optimization of developer identity mapping and the automation of reporting processes support more robust, data-driven infrastructures that enable organizations to make informed decisions based on reliable and consistent information.

In addition, the project relates to Sustainable Development Goal 5: Gender Equality and **Sustainable Development Goal 10: Reduced Inequalities** through its approach to automated report generation. The reports produced by the Large Language Model are designed to be based exclusively on objective performance data extracted from code repositories, avoiding subjective judgments or personal characteristics unrelated to the work itself. By relying solely on measurable contributions and standardized metrics, the system reduces the risk of bias in performance evaluation and helps ensure that all developers are assessed under the same criteria. By minimizing human subjectivity in both data processing and report generation, the project promotes fairer and more transparent evaluation practices.

Although the project does not directly target social inequalities, its emphasis on objective, data-driven assessment contributes indirectly to more inclusive and equitable decision-making processes within organizations. In this way, the project demonstrates how

technological innovation can support not only industrial development but also responsible and unbiased use of artificial intelligence in professional environments.

1.4 STRUCTURE OF THE WORK

The remainder of this document is organized as follows. Chapter 2 reviews the state of the art in developer identity resolution and automated report generation, covering relevant techniques, existing solutions, and their limitations. Chapter 3 describes the system developed, structured around the two main processes addressed in this project: for each process, the problem is formally defined, the proposed solution is designed, and the implementation is described in detail. Chapter 4 presents the results obtained for both processes, including the evaluation setup, performance metrics, and a critical analysis of the findings. Chapter 5 discusses the conclusions drawn from the project and outlines potential directions for future work. Finally, Chapter 6 contains the bibliographic references cited throughout the document.

CHAPTER 2. STATE OF ART

2.1 PROCESS 1: DEVELOPER IDENTITY MAPPING

The problem of consolidating developer identities across version control systems has been widely studied in the software engineering and data mining literature. Developers frequently contribute to repositories under multiple logins, email addresses, or naming conventions, which leads to fragmented identity data and distorted productivity metrics if not handled correctly.

Traditional approaches to this problem rely on deterministic rules, such as exact matches on names or email addresses. While simple and computationally efficient, these methods are highly sensitive to data inconsistencies. Small variations in spelling, the presence of accented characters, or changes in email domain are enough to break an exact match, leaving many identities unresolved.

To address these limitations, string similarity metrics have been widely adopted. The Jaro-Winkler similarity measure, originally proposed in the context of record linkage for census data, has become a standard technique for comparing short strings such as names and identifiers. It computes a similarity score between 0 and 1 based on the number of matching characters and transpositions, with an additional bonus for strings sharing a common prefix. This makes it particularly well-suited for comparing human names, where typographical variations and abbreviated forms are common. Despite its effectiveness, Jaro-Winkler has known limitations: its performance degrades with longer strings, and it relies solely on surface-level character patterns without capturing semantic or contextual information.

Several works have proposed more comprehensive approaches to the identity resolution problem in software repositories. [1] addressed the challenge at scale, processing over 38 million author IDs extracted from 2 billion Git commits. Their approach combines a blocking strategy to reduce the candidate space with a supervised machine learning model to predict whether two author IDs belong to the same developer. They found that around 14.8 million IDs had at least one alias, highlighting the magnitude of the problem in real-world ecosystems. Similarly, the ALFAA system proposed by [2] augments author strings with behavioral fingerprints — including timezone frequencies, sets of modified files, and commit message embeddings — and applies active learning to reduce the labeling effort required to train supervised models on large datasets.

Beyond string similarity and supervised classification, embedding-based approaches have also been explored. Pre-trained language models and document embeddings such as Doc2Vec have been applied to entity resolution tasks, capturing semantic similarity between

textual representations that goes beyond character-level matching. These methods have shown promise in identifying relationships between identities that share no obvious textual overlap but exhibit similar behavioral or contextual patterns. However, their integration into production pipelines introduces additional complexity in terms of computational cost, model maintenance, and explainability.

A recurring limitation across existing approaches is the tension between automation rate and reliability. Methods that maximize the number of automatic matches tend to generate more false positives, which in the context of developer analytics have a direct and difficult-to-reverse negative impact on metric accuracy. Methods that prioritize precision, such as conservative threshold-based heuristics, leave a large proportion of cases unresolved and dependent on manual validation. This tradeoff between precision and recall, and its operational consequences, is a central concern in the design of any production-ready identity resolution system.

2.2 PROCESS 2: AUTOMATED REPORT GENERATION

The automated generation of natural language reports from structured data has been an active research area for decades, but the landscape has shifted substantially with the emergence of Large Language Models. Traditional Natural Language Generation systems relied on hand-crafted templates and rule-based pipelines that required significant engineering effort to customize for new domains or report types. While effective for standardized reporting tasks, these approaches lack flexibility and do not scale well when reports must be adapted to different clients, datasets, or contexts.

Early data-to-text systems addressed this by separating content planning from surface realization, allowing structured data to be first interpreted and then expressed in natural language. These systems worked well in narrow domains such as weather forecasting or sports reporting, but required domain-specific knowledge engineering that limited their generalizability.

The introduction of large-scale pre-trained language models has fundamentally changed what is possible in automated reporting. Models such as GPT-4 and Claude can receive structured data as input and produce coherent, context-aware narratives without domain-specific training. Prompt engineering has emerged as the primary mechanism for controlling the output of these models. [3] provide a comprehensive taxonomy of prompting techniques,

covering structured approaches such as role prompting and output formatting, which form the basis of the prompt design strategy adopted in this project.

A particularly relevant application is that of [4], who propose AI Analyst, a framework that leverages LLMs to generate analytical reports directly from time series data. Their work demonstrates that LLMs can produce structured, interpretable narratives from numerical data, and provides a systematic evaluation of how prompt design and model choice affect report quality. Although their focus is on financial time series, the underlying challenges, grounding the model's output in specific numerical values, ensuring consistency across reports, and evaluating factual accuracy, are directly applicable to the developer productivity reporting context addressed in this project.

However, the integration of LLMs into enterprise reporting workflows introduces its own set of challenges. Hallucination, the tendency of language models to generate plausible-sounding but factually incorrect content, is a significant concern when reports must accurately reflect specific numerical data. [5] demonstrate this through Chain-of-Verification, showing that without deliberate mechanisms to ground model outputs, hallucination rates in factual generation tasks remain substantial. In this project, hallucination is addressed through prompt design constraints rather than post-generation verification: the prompts explicitly instruct the model to rely exclusively on the data provided as input, without introducing figures or conclusions not present in the query results.

In the context of software analytics, these challenges are particularly relevant. Reports describing developer productivity must be grounded exclusively in measurable data, since any subjective or inconsistent framing could undermine the credibility of the platform and introduce unfair bias into performance evaluations. This makes the design of the prompt engineering pipeline and the validation strategy critical components of any LLM-based reporting system in this domain. The current state of the art thus points to a clear opportunity: LLMs offer the flexibility and scalability that traditional systems lack, but their deployment in production environments requires careful engineering to ensure reliability, objectivity, and factual grounding.

CHAPTER 3. SYSTEM AND MODEL

3.1 SYSTEM OVERVIEW

Foreworth is a software analytics platform that collects and processes data from client code repositories in order to generate developer productivity metrics and reports. Understanding the underlying infrastructure is relevant to both processes developed in this project, as the data they rely on flows through two distinct storage layers before becoming available for analysis. Figure 1 shows the architecture in detail.

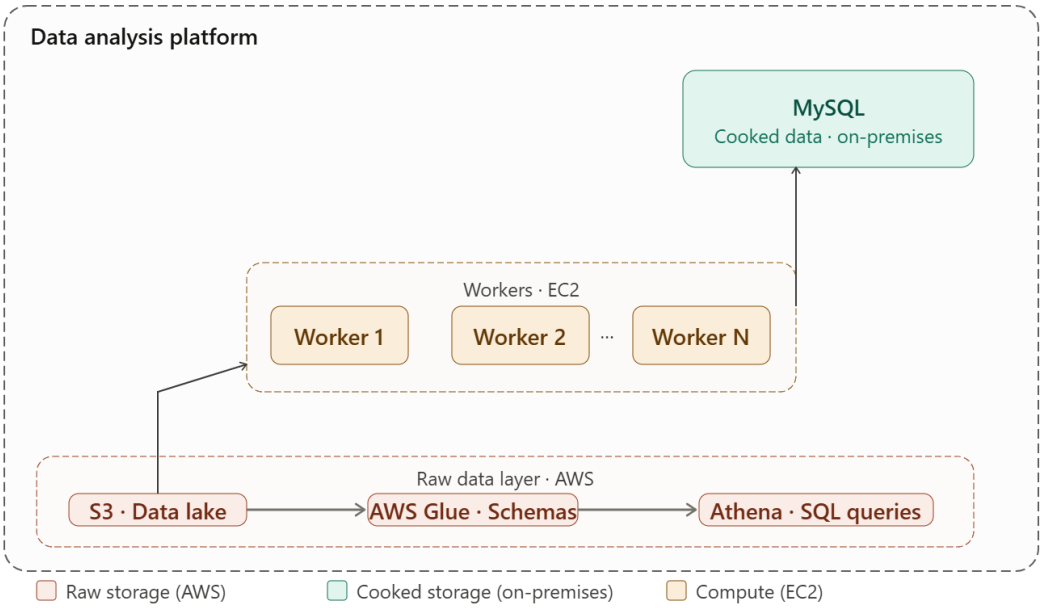

Figure 1. Data analysis platform architecture

The first layer consists of Amazon S3 buckets, which act as the platform's primary data lake. All raw data ingested from client repositories lands here. AWS Glue manages the schema definitions on top of these buckets, and Amazon Athena provides a serverless SQL interface for querying this data directly. This layer prioritises scalability and completeness, the original data is preserved in full, and the schema can evolve as new variables are introduced or processing logic changes.

The second layer is a set of MySQL relational databases hosted on-premises, which store the processed, or "cooked", data produced after the analysis pipelines have run. Unlike the

AWS layer, this one is optimised for fast, repeated queries and low-latency access. It serves as the primary data source for the web application and any connected BI tooling. The separation between these two layers is deliberate: raw data stays in S3 for durability and reprocessing capability, while MySQL exposes only the clean, aggregated metrics that downstream consumers need.

3.2 PROCESS 1: DEVELOPER IDENTITY MAPPING

3.2.1 PROBLEM STATEMENT

To generate accurate development metrics, Foreworth analyzes the code repositories where engineering teams conduct their work. A critical step in this process is the consolidation of user identities: since individual developers often contribute to repositories under multiple Git credentials, failing to unify these identities would directly distort key metrics such as productivity per developer or contribution analysis.

The complete analysis of a tenant within the Foreworth platform is divided into two main phases. The first phase corresponds to a pre-analysis stage, whose objective is to define and apply mapping rules that group multiple user identities under a single developer profile. Once this automatic grouping is completed, a manual validation process is carried out to ensure the correctness of the mappings. The second phase focuses on the computation of developer effort metrics, including estimations of productive days based on factors such as programming language complexity and lines of code, using the Kokoma algorithm as the core computational model.

The work developed in this project focuses on the pre-analysis phase, specifically on the following tables stored in Foreworth's on-premises MySQL database:

The **Original Authors** table contains all users who have contributed to the analyzed repositories. Each entry includes a unique identifier, a name, and an email address, as shown in Table 1.

Column Name	**Data Type**
Id	bigint

Name	varchar(512)
Email	varchar(512)

Table 1. Original Authors Table Structure

The **Mapping Rules** table includes both manual and automatic mapping rules associated with each original author. Each rule defines a pattern and a target, indicating which original author identity should be mapped to which developer profile. A key field in this table is the *needs revision* flag, which marks rules that require manual validation, as shown in Table 2.

Column Name	**Data Type**
Id	bigint
application_source_id	bigint
mapping_rule_type_id	bigint
mapping_rule_match_type_id	bigint
Order	int
valid_from	bigint
valid_to	bigint
pattern	varchar(1024)
target	varchar(1024)
repository_id	bigint
needs_revision	bit(1)
modified_on	datetime
modified_by	varchar(256)
modified_from	varchar(255)

Table 2. Mapping Rules Table Structure

The **Developers** table represents the final list of developers after the mapping process has been applied, consolidating all resolved identities into unified profiles, as shown in Table 3.

Column Name	Data Type
Id	bigint
Name	varchar(512)
disabled	tinyint(1)
needs_revision	bit(1)
inactive_from	int
start_date	date
modified_on	datetime
modified_by	varchar(256)
modified_from	varchar(255)

Table 3. Developers Table Structure

Exploratory Analysis

Before designing any solution, an exploratory analysis of the data was conducted to better understand the scale and nature of the problem.

First, the tenants with the highest number of mapping rules were identified in order to select representative cases for experimentation. These tenants typically present higher complexity and a larger number of ambiguous mappings, making them particularly relevant for evaluating the performance of the proposed methods. As shown in Table 4, the largest tenants include organizations such as BBVA and Indra, with over 20,000 mapping rules each.

Tenant	Number of Mapping Rules
BBVA	20820
Indra	20460
Seguros Bolívar	5024
Cepsa	4608

Emeal NTT Data	3028
Hispatec	2567
Madrid	2552
Veolia	2434
Mutua	2422
Goalsystems	2082

Table 4. Tenants with highest number of mapping rules

Second, the most frequent developer identities across the dataset were analyzed. As shown in Table 5, this analysis revealed a significant presence of non-human accounts such as GitHub, root, or dependabot. These automated accounts introduce noise into the mapping process and must be handled separately from real developer identities, reinforcing the need for a dedicated bot detection mechanism.

Developer	**Count**
GitHub	32
dependabot[bot]	17
Root	16
unknown	11
Administrator	9
Ubuntu	7
EC2 Default User	7
Your Name	6
admin	6
usuario	5
Daniel Lopez	5
github-actions[bot]	5

Daniel	5
María	4
Javier Perez	4
Alex	4
Developer	4
DEVOPS	4
Miguel Garcia Puche	4
System Administrator	4
Daniel Diaz	4
Javier Ruiz	4
Renovate Bot	3
Usuario de los responsables del aplicativo	3
bitbucket-pipelines	3
semantic-release-bot	3
=	3

Table 5. Most common developer identities

Illustrative Example

To facilitate understanding of the problem, a representative sample dataset is presented. Table 6 shows a set of original authors that would typically appear when analyzing the repositories of a tenant, illustrating the variability in naming conventions and email usage that can correspond to the same individual.

Name	**Email**
hrodriguez	hrodriguez@fw.com
hrodríguez	hrodriguez@fw.com

hrodriguez	hrodriguez@ali.com
hrodríguez	hrodriguez@ali.com
arodriguez	arodriguez@fw.com
arodríguez	arodriguez@fw.com
arodriguez	arodriguez@ali.com
arodríguez	arodriguez@ali.com
halbertrodriguez	halbertrodriguez@fw.com
halbertrodriguez	hrodriguez@fw.com
arodríguez	arodriguezpaco@ali.com
arodriguezpaco	arodriguezpaco@fw.com
Root	root@ali.com
GitHub	github@noreply.com

Table 6. Example of original authors

Table 7 presents a comparison between the mapping approach previously implemented by Foreworth and the improved approach proposed in this work. The previous algorithm primarily relied on exact matches between names and emails, which led to fragmented mappings when variations in spelling, accents, or email domains were present. As a result, multiple identities corresponding to the same individual were often treated as separate developers. In contrast, the proposed approach aims to unify all variations that correspond to the same real-world individual into consolidated developer profiles. Additionally, the system incorporates a mechanism to automatically detect and classify non-human accounts, labelling them as BOT and separating them from human contributors.

Pattern	**Previous Target**	**Goal Target**
hrodriguez |#| hrodriguez@fw.com	halbertrodríguez	Hugo Rodríguez
hrodríguez |#| hrodriguez@fw.com	halbertrodríguez	Hugo Rodríguez
hrodriguez |#| hrodriguez@ali.com	Hugo Rodríguez	Hugo Rodríguez

Hugo Rodríguez |#| hrodriguez@ali.com	Hugo Rodríguez	Hugo Rodríguez
halbertrodriguez |#| halbertrodriguez@fw.com	halbertrodriguez	Hugo Rodríguez
halbertrodríguez |#| hrodriguez@fw.com	halbertrodríguez	Hugo Rodríguez
arodriguez |#| arodriguez@fw.com	arodriguez	Álvaro Rodríguez
arodriguez |#| arodriguez@ali.com	Álvaro Rodríguez	Álvaro Rodríguez
Álvaro Rodríguez |#| arodriguez@ali.com	Álvaro Rodríguez	Álvaro Rodríguez
arodríguez |#| arodriguezpaco@ali.com	arodríguez	Álvaro Rodríguez
arodriguezpaco |#| arodriguezpaco@fw.com	arodriguezpaco	Álvaro Rodríguez
root |#| root@ali.com	root	BOT
GitHub |#| github@noreply.com	GitHub	BOT

Table 7. Comparison between previous mapping and proposed identity mapping

3.2.2 SOLUTION DESIGN

The solution proposed for the developer identity mapping problem is structured around a unified decision framework that separates the question of *how to compute similarity* from the question of *how to act on it*. Regardless of the method used to produce a similarity score, all approaches evaluated in this project feed into the same three-zone classification logic, which determines the outcome for each identity pair.

The three zones are defined as follows. Pairs whose score exceeds T_upper are considered confirmed matches and are automatically merged into a single developer profile without any human intervention. Pairs whose score falls between T_lower and T_upper are flagged as suggested matches and added to a review queue, where a member of the Foreworth team must manually validate whether the two identities should be unified or kept separate. Pairs whose score falls below T_lower are automatically classified as non-matches and discarded from further consideration.

This design is intentional and directly addresses the precision requirements of the system. False positives, incorrectly merging two distinct developers, have a significantly higher operational cost than false negatives, since they directly distort productivity metrics in a way that is difficult to reverse. By reserving the automatic match zone for only the highest-

confidence pairs, the system ensures that the metrics produced by the platform remain reliable, while still reducing the manual workload substantially compared to the previous approach. Figure 2 illustrates the complete framework.

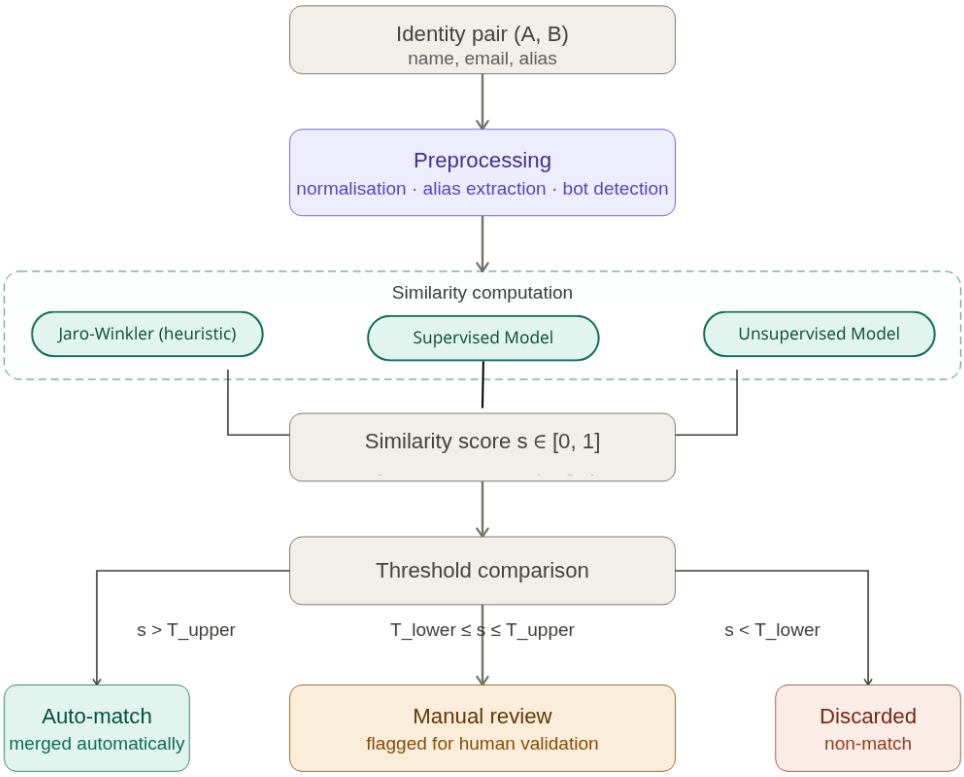

Figure 2. Three zone decision framework

Heuristic-Based Approach

The starting point of the solution is an enhanced version of the Jaro-Winkler similarity measure, which was already partially used in Foreworth's original system. As described in [1], the Jaro-Winkler metric computes a similarity score between 0 and 1 based on character matches and transpositions, with an additional bonus for strings sharing a common prefix, making it particularly well-suited for comparing human names and identifiers.

The original system applied Jaro-Winkler exclusively between cleaned names and only within the suggestion mechanism. The enhanced approach expands the comparison strategy to include four additional combinations:

- Name A vs. Name B
- Alias A vs. Alias B, where the alias is defined as the portion of the email address before the '@' symbol
- Name A vs. Alias B

- Alias A vs. Name B

By incorporating these cross-comparisons, the system is able to identify a substantially larger number of true positive mappings that the original approach would have missed. Both names and aliases are preprocessed before comparison, removing special characters and converting all letters to lowercase to ensure uniformity and reduce noise.

Additionally, a name normalization step is applied to handle the common convention of recording names in "last name, first name" format. For example, an author recorded as "Rodríguez, Hugo" and another as "Hugo Rodríguez" would fail to match under standard string comparison despite referring to the same individual. By detecting the presence of a comma and reordering the tokens accordingly, placing the first name before the last name, the system ensures that these variations are correctly identified as potential matches.

Finally, the similarity threshold used with this method is dynamic as it adapts according to the minimum length of the two strings being compared. This threshold is defined by a linear ramp function, allowing for more flexible and context-sensitive matching criteria. For very short strings, a higher threshold is required to avoid spurious matches, while longer strings tolerate a slightly lower threshold. This design choice is particularly important for reducing false positives in cases where short names or abbreviations could otherwise be incorrectly merged. Figure 3 illustrates the dynamic threshold as a function of string length.

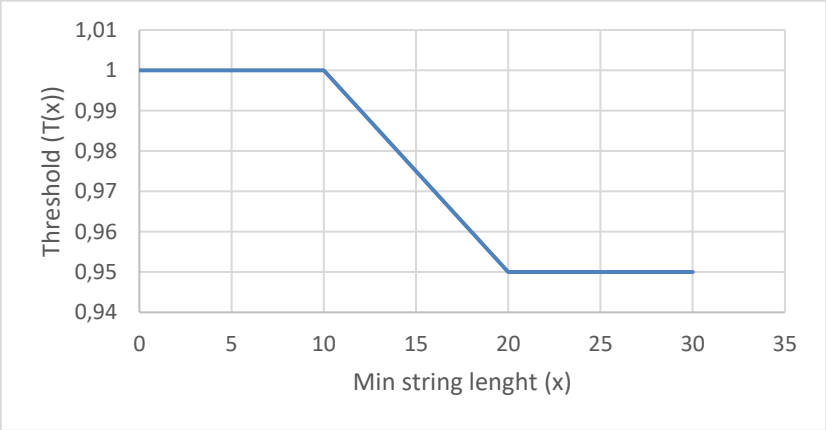

Figure 3. Dynamic similarity threshold as a function of string length

Supervised Learning Approach

In addition to the heuristic method, supervised machine learning approaches were explored to assess whether a data-driven solution could improve the matching process and better handle ambiguous cases, following the direction taken by recent work in identity resolution at scale [1].

The problem is formulated as a binary classification task: given a pair of original authors, the model must predict whether both identities correspond to the same developer or not.

Several models were evaluated, including Logistic Regression, Random Forest, XGBoost, and LightGBM. These models were selected due to their different theoretical properties and their suitability for binary classification tasks.

Logistic Regression provides a simple and interpretable baseline but is limited in capturing non-linear relationships. Random Forest improves robustness through the aggregation of multiple decision trees, making it particularly effective at reducing overfitting. XGBoost and LightGBM, both based on gradient boosting techniques, offer higher modeling capacity and efficiency, although they require careful tuning to avoid overfitting.

To enable these models, a set of features was constructed for each pair of original authors capturing different aspects of similarity between user identities:

- String similarity: Jaro-Winkler score of Name A vs. Name B
- String similarity: Jaro-Winkler score of Alias A vs. Alias B
- String similarity: Jaro-Winkler score of Name A vs. Alias B
- String similarity: Jaro-Winkler score of Alias A vs. Name B
- Most common timezone offset similarity:

$$sim_{offset} = 1 - \frac{|mode_{offset_A} - mode_{offset_B}|}{24}$$

- Average commit time similarity:

$$sim_{committime} = 1 - \frac{|avg_{committime_A} - avg_{committime_B}|}{1440}$$

- Most common programming language similarity:

$$sim_{committime} \rightarrow 1 \; if \; mode_{language_A} == mode_{language_B}$$
$$sim_{committime} \rightarrow 0 \; otherwise$$

The inclusion of behavioral features such as timezone offset, commit time, and programming language is motivated by the observation that developers working in the same team tend to share similar working patterns, even when their textual identifiers differ significantly.

Unsupervised Learning Approach

Finally, embedding-based representations were investigated as an alternative way of capturing similarity between developer identities without relying on manually engineered string features. The motivation was that character-level metrics like Jaro-Winkler, while effective, are inherently limited to surface-level string matching and cannot capture semantic relationships between identities that share no obvious lexical overlap. As shown in [2], augmenting author strings with behavioral fingerprints and embedding-based

representations of commit messages can improve identity resolution beyond what string similarity alone achieves.

Two model architectures were considered: a Siamese network, in which two identical subnetworks independently generate embeddings for a pair of identities and learn to predict whether they belong to the same developer, and an encoder-decoder model, in which the encoder produces an embedding that the decoder attempts to map to the correct target name. The Siamese approach was pursued further given its more natural fit to the pairwise similarity framing of the problem.

Several architectures were explored progressively, starting with linear layers and ReLU activations, then incorporating recurrent layers, and finally LSTM-based networks trained with Contrastive Loss measured over Euclidean distance between embedding pairs. The training data was constructed from existing validated mapping rules extracted from production tenants, using name and email alias fields as inputs.

3.2.3 IMPLEMENTATION

The final system integrates the enhanced Jaro-Winkler approach as the core decision mechanism, deployed within Foreworth's production pipeline. Although supervised and unsupervised learning models were also evaluated as part of this project, the heuristic approach was selected as the production solution. The reasoning behind this decision is discussed in detail in Chapter 4, once the experimental results are available to support it.

The backend is implemented in C#, consistent with Foreworth's existing production architecture, and runs inside a Docker containerized environment. Database access is managed through DBeaver, connecting to the MySQL on-premises database where the processed developer and mapping rule data is stored. The algorithm was developed and debugged using Visual Studio, with OpenVPN providing secure remote access to the production infrastructure. For the experimental and evaluation phases of the project, Python was used to prototype similarity approaches, train and evaluate the supervised models, and analyze results before any logic was integrated into the production backend. Version control follows a fork-based Git workflow, with features developed and validated in isolated branches before being merged into the main codebase.

The implementation follows a three-zone decision framework explained previously in Section 3.2.2.

In practice, the system is triggered in two scenarios. The first is during the initial analysis of a new client, where all original authors extracted from the repositories are processed for the first time. The second is on a recurring basis each month, when new logins detected in the repositories are evaluated against the existing developer profiles. In both cases, high-confidence matches are resolved automatically, while ambiguous cases are surfaced through the frontend interface as actionable suggestions, displaying the relevant identity information alongside each candidate pair so that the team can make an informed decision without needing to consult the underlying data directly.

Since the enhanced Jaro-Winkler approach is a deterministic algorithm, its outputs are fully reproducible given the same input data and threshold configuration. Unlike a trained machine learning model, there is no stochastic component and no dependency on a serialized model file or training dataset. Any change to the threshold parameters is made directly in the codebase and immediately traceable through version control, ensuring that the behaviour of the system at any point in time can be reconstructed exactly.

3.3 PROCESS 2: AUTOMATED REPORT GENERATION

3.3.1 PROBLEM STATEMENT

Foreworth's core product delivers developer productivity insights to its clients by analyzing software repositories. A key component of this value delivery is the generation of analytical reports that summarize individual developer performance, highlight behavioral patterns, and identify areas for improvement. Traditionally, these reports were produced manually by consultants, a process that is both time-consuming and difficult to scale as the number of clients and developers grows.

The goal of this second process is to automate the generation of these reports using Large Language Models, and in doing so, establish a reusable protocol that allows consultants to produce reports independently and efficiently. The objective is not limited to generating a single report type, but to define a general framework that covers multiple report categories and can be extended over time.

Concretely, the system must be capable of generating the following report types, each drawing on different data sources:

Commit and technology report: analyzes the developer's total commits, excluded commits, non-code commits, merge commits, added lines of code, and technologies used, in order to infer their likely role and contribution profile.

Code quality report: analyzes technical debt and code reliability scores extracted from the platform, as well as the proportion of commits that include test files, in order to assess the quality of the developer's output.

Work routine report: analyzes productivity patterns across days of the week and hours of the day, as well as available information about remote work and Copilot usage.

Individual developer report: consolidates the three previous reports into a single narrative for a given developer.

Bottom performer report: a variant of the individual report targeted at developers with low productivity, which includes an explicit disclaimer reminding the consultant to verify the quality of the underlying data before drawing conclusions.

The system must satisfy the following requirements. All reports must be written in Spanish, since that is the language used by Foreworth's clients and consultants. The reports must be grounded exclusively in the data provided through SQL queries, without introducing fabricated or hallucinated figures. The language and level of detail must be appropriate for a non-technical audience, typically a CEO, project manager, or consultant, who understands the business context but is not familiar with the technical details of software engineering. Finally, the outputs must be consistent and structured enough to be delivered to clients with minimal post-processing.

The evaluation of the system combines human review by Foreworth consultants, who assessed the reports against real developer data, with LLM-as-a-judge evaluation, in which a separate model scores the generated reports on dimensions such as factual grounding, clarity, and tone.

3.3.2 SOLUTION DESIGN

The automated report generation system follows a modular pipeline structured around three layers: a consultant-facing interface, a data retrieval layer, and a generation layer based on Large Language Models.

Frontend and user interaction

The entry point is an internal web tool designed for Foreworth consultants. Through this interface, the consultant selects the target tenant and the type of report to generate. Depending on the report type, the system requests the specific inputs required: for example, generating a bottom performer report requires the name of the developer in question, while a general developer report requires a date range. The interface also allows the consultant to select the LLM to use from the set of models available through Amazon Bedrock, giving the team flexibility to experiment with different models without modifying the underlying pipeline.

Data retrieval

Once the consultant submits the request, the backend determines which queries to execute based on the selected report type. All queries are routed to AWS-based data sources. Each report type draws on a different combination of data: commit volume and technology breakdown, code quality metrics such as technical debt and code reliability scores, work routine patterns across days of the week and hours of the day, and developer configuration data such as Copilot usage and remote work status. The queries return structured tabular results that are passed directly to the generation pipeline.

Two-stage generation pipeline

The generation pipeline follows a two-stage prompting strategy. In the first stage, each query result is passed individually to the LLM along with a dedicated prompt that instructs the model to produce a concise analytical summary of that specific data slice. This produces a set of intermediate reports, one per data dimension. In the second stage, all intermediate summaries are combined and passed to a consolidation prompt that instructs the model to produce a single coherent final report following the structure required for that report type.

This two-stage design was chosen over a single large prompt for two reasons. First, it reduces the risk of the model losing relevant details when processing large volumes of raw data in a single context window. Second, it makes the pipeline more modular: each intermediate prompt can be adjusted independently without affecting the rest of the pipeline, which is important in a production environment where requirements evolve.

Figure 4 illustrates an example of the complete pipeline for a developer report, showing how the consultant's inputs trigger a sequence of SQL queries, each processed by the LLM in the first stage to produce intermediate summaries, which are then consolidated into a final report in the second stage.

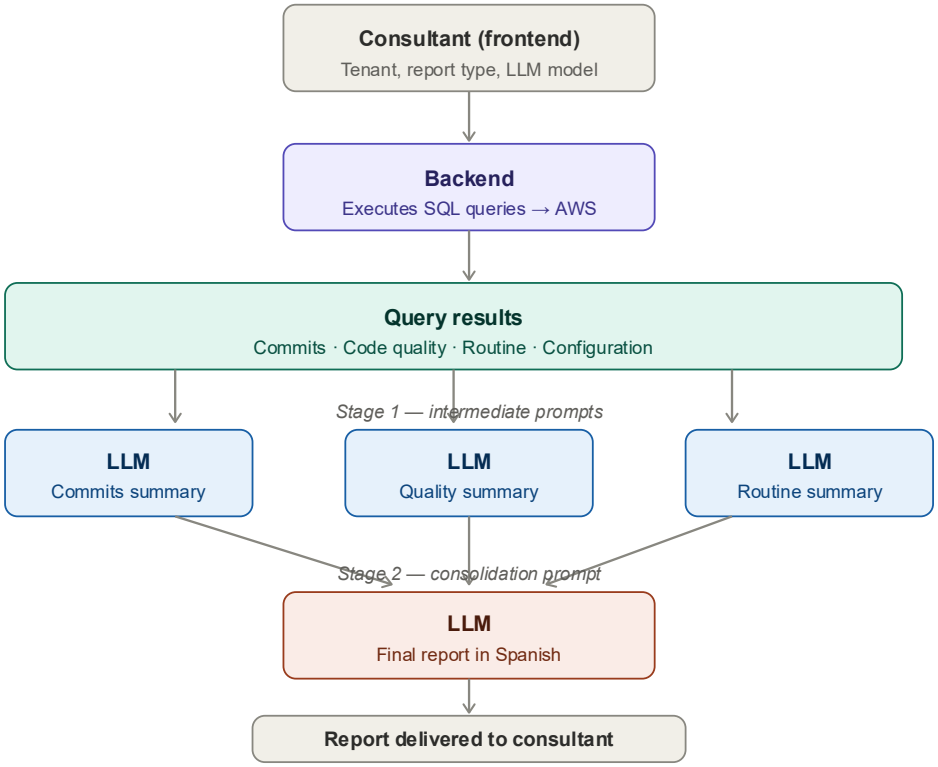

Figure 4. Example of a complete pipeline

Prompt design

All prompts follow a consistent structure with four clearly delimited sections using XML-style tags: role, objective, context, and step-by-step instructions. The role section establishes the model's persona as an expert data analyst working at Foreworth, familiar with the platform's core metric, Achieved Days. The objective section describes the specific analytical task for that prompt. The context section explains the schema of each data table passed as input, including the meaning of each column and any relevant thresholds or

business rules. Finally, the instructions section specifies the exact structure and content the model must produce, including which metrics to mention, how to interpret specific values, and any fixed elements that must always appear in the output.

A notable example of a fixed element is the disclaimer included at the top of every bottom performer report, which explicitly instructs the consultant to verify the quality of the developer's data before drawing any conclusions. This was introduced to prevent consultants from delivering reports based on incorrectly mapped identities or incomplete commit data, and reflects a broader principle in the system: the LLM generates the analysis, but the consultant remains responsible for validating its inputs.

The consolidation prompt in the second stage follows the same structure, but its input is the set of intermediate summaries rather than raw query results. It instructs the model to produce a flowing narrative in Spanish, without section headers, addressed to a non-technical reader such as a CEO or project manager.

3.3.3 IMPLEMENTATION

The report generation system was implemented using the following technology stack. The frontend is built with Next.js, providing the consultant-facing interface through which tenants, report types, and LLM models are selected. The backend logic is implemented in C#, consistent with the rest of Foreworth's production architecture. Data retrieval is performed through SQL queries against AWS-hosted databases, specifically Amazon Athena for the raw repository data and MySQL for the processed on-premises dataset. The LLM integration is handled through Amazon Bedrock, AWS's managed service for accessing foundation models, which provides a secure and scalable API to several models including those from the Claude and Titan families. Version control follows a fork-based Git workflow, with features developed and validated in isolated branches before being merged into the main codebase. For the exploratory and evaluation phases of the project, Python was used to prototype prompt variations, analyze query outputs, and assess report quality before integrating the final logic into the production backend.

The implementation is organized around report type handlers, each of which encapsulates the full pipeline for a given report. All report types, including those used in production, are defined entirely through the frontend interface, where both the SQL queries and the prompts for each data dimension are specified. This applies to developers and consultants alike, meaning there is no separate configuration layer in the codebase for report definitions. This design centralizes report management in a single interface and makes the system fully

extensible without requiring code changes. The Bedrock API integration is abstracted into a shared service that accepts a prompt and a model identifier and returns the generated text, allowing the consultant to switch between available models from the frontend without any changes to the report logic.

The execution pipeline follows these steps for every report request. First, the backend identifies the report type and collects the required inputs provided by the consultant through the frontend. Second, it executes the predefined SQL queries against the appropriate AWS data sources and collects the results. Third, each query result is passed to its corresponding intermediate prompt, and the LLM generates a structured analytical summary. Fourth, all intermediate summaries are assembled and passed to the consolidation prompt, producing the final report. The output is then returned to the frontend and displayed to the consultant.

Since all report definitions, including queries and prompts, are managed through the frontend interface, reproducibility depends on the consultant documenting and preserving the configurations used for each report type. The choice of LLM model is logged alongside each generated report, ensuring that the generation conditions are traceable. The SQL queries are fully parameterised in all cases, with tenant, developer name, and date range as explicit inputs, so any report can be re-executed given the same parameters and the same stored configuration.

CHAPTER 4. RESULTS

4.1 PROCESS 1: DEVELOPER IDENTITY MAPPING

4.1.1 EVALUATION SETUP

To evaluate the performance of the proposed approaches for developer identity mapping, a structured experimental setup was designed, including dataset construction, model training, and evaluation procedures.

Dataset Construction

The training dataset was built using pairs of developers and their associated features. Each pair was labeled with a binary target indicating whether both identities correspond to the same developer or not. This formulation transforms the identity resolution problem into a supervised classification task.

To ensure both realism and computational feasibility, pairs were generated within the same tenant. This allows the dataset to include both positive cases (different aliases belonging to the same developer) and negative cases (distinct developers), while avoiding unrealistic cross-tenant matches. The dataset was constructed by combining data from multiple tenants, including:

- BBVA
- Indra
- Seguros Bol´ıvar
- Cepsa
- Emeal NTT Data
- Hispatec
- Madrid
- Veolia
- Mutua
- Goalsystems

In addition to the training dataset, external tenants not included during training (such as Inditex, Wolters Kluwer, and Once) were used for testing. This allows evaluating the generalization capability of the models in unseen real-world scenarios.

Before training, the data was preprocessed and cleaned using an ETL pipeline, including normalization of names and emails and feature extraction.

Training Procedure

Model training and evaluation were performed using the scikit-learn library. To ensure robustness and reduce overfitting, a cross-validation strategy with 5 folds was applied on the training dataset.

Hyperparameter tuning was performed using grid search over key parameters, including:

- Maximum depth of the trees (*max_depth*)
- Learning rate (*learning_rate*)
- Number of estimators (*n_estimators*)
- Subsampling ratio (*subsample*)
- Feature sampling ratio (*colsample_bytree*)

For example, in the case of LightGBM, one of the best-performing configurations was:

- learning rate = 0.3
- max depth = 20
- n estimators = 300

Evaluation Metrics

Model performance was evaluated using standard classification metrics:

- Accuracy
- Precision
- Recall
- F1-score

Special emphasis was placed on precision, as false positives (incorrectly merging different developers) have a significant negative impact on the quality of the final metrics generated by the platform. Therefore, minimizing incorrect matches is prioritized over maximizing recall.

Feature Analysis

In addition to performance metrics, feature importance was analyzed using the Random Forest model. This allows identifying which features contribute most to the decision-making process, providing interpretability and insights for further improvements.

4.1.2 PERFORMANCE METRICS

Heuristic-Based Approach

The enhanced Jaro-Winkler approach was validated using the same labelled pairs described in the evaluation setup. Four functional test cases were defined to systematically assess the

behaviour of the system: merging identities with matching emails, merging identities with matching names, merging identities with similar names across different emails, and merging identities with similar email aliases across different names. The original baseline only passed the first two. After extending the comparison strategy to include cross-field combinations and adjusting the thresholds, the enhanced system passed all four.

On the full labelled dataset, the enhanced approach achieved a precision of 0.97 and a recall of 0.72, with an overall accuracy of 0.9931. These results confirm that the heuristic improvements substantially increased the matching rate without sacrificing the precision that is critical for this task.

Supervised Learning Approach

Table 8 presents the performance of the evaluated models on the test dataset. Overall, ensemble-based methods significantly outperform simpler approaches, achieving accuracy values close to 1. In particular, Random Forest and XGBoost both reach an accuracy above 0.996.

However, given the nature of the problem, accuracy alone is not sufficient to assess model performance. Special attention must be paid to the metrics associated with the positive class, that is, correctly identifying when two identities belong to the same developer. Precision is especially critical here, as false positives lead to incorrect merges that directly distort downstream productivity metrics.

Random Forest achieves the highest precision (0.97), making it the most reliable model in avoiding incorrect matches, while maintaining a strong recall of 0.76. XGBoost provides a slightly higher recall (0.79) at the cost of a small reduction in precision, indicating a more aggressive matching behaviour. LightGBM, despite achieving a relatively high recall, shows a very low precision (0.23), which makes it unsuitable for this task. Logistic Regression performs significantly worse across all metrics, confirming that linear models are not capable of capturing the complexity of the problem.

Model	Accuracy	Precision	Recall	F1-score

Logistic Regression	0.7949	0.04	0.61	0.07
Random Forest	0.9965	0.97	0.76	0.85
XGBoost	0.9965	0.93	0.79	0.86
LightGBM	0.9627	0.23	0.79	0.36

Table 8. Comparison of machine learning models for developer identity mapping

Among the supervised models, Random Forest was selected as the strongest machine learning candidate, as it provided the best trade-off between precision and recall within the supervised learning family. However, this does not imply that it was selected as the final production solution. The final decision was made after comparing this model with the enhanced Jaro-Winkler approach on unseen tenants.

After selecting Random Forest, feature importance was analyzed to better understand which variables contributed most to the classification task. As shown in Figure 5, the most influential features are the Jaro-Winkler similarity scores computed between names and aliases, while contextual variables such as commit time, timezone offset, and programming language play a secondary role. This is consistent with the nature of the problem, since identity resolution primarily depends on textual similarity between identifiers. In particular, the highest importance is assigned to cross-comparisons between aliases and names, suggesting that combining both sources of information is more informative than relying on a single field. It is also worth noting that the dominance of Jaro-Winkler-based features in the model explains in part why the gap between the supervised approach and the enhanced heuristic is relatively small, as both methods are fundamentally drawing on the same underlying signal.

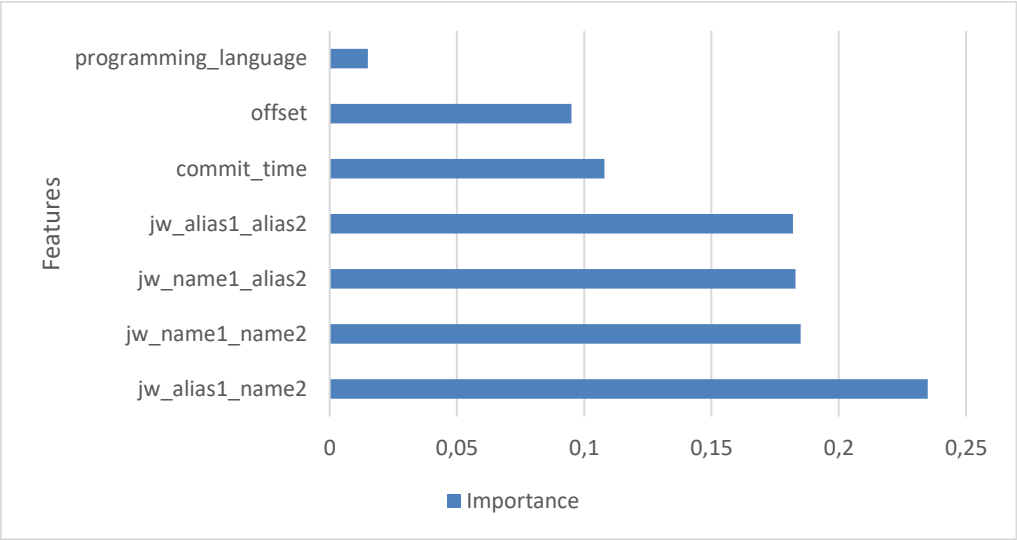

Figure 5. Feature importance obtained from the selected Random Forest model

Unsupervised Learning Approach

The Siamese LSTM network described in the solution design was trained and evaluated on the same labelled dataset used for the supervised models. The network was trained using Contrastive Loss measured over Euclidean distance between embedding pairs, and performance was assessed across a range of distance thresholds to understand the precision-recall trade-off.

The results showed a clear dependency on the chosen threshold. At a threshold of 0.5, the model achieved an accuracy of 0.91 and a precision of 0.95, which are competitive figures. However, when evaluated on tenants not included in the training data, performance dropped noticeably and did not consistently match the results obtained by the enhanced Jaro-Winkler approach. This suggests that the embeddings learned by the model did not generalise sufficiently well to unseen naming conventions and email patterns.

Additionally, increasing the maximum token length from 5 to 32 characters, which corresponds to the 95th percentile of string lengths in the real dataset, produced only marginal improvements. Adding attention layers and dropout did not lead to consistent gains either, and the simpler LSTM architecture performed comparably to more complex variants.

Given these results, the embedding-based approach was not selected for integration into the production pipeline. The enhanced Jaro-Winkler system offered similar or better precision on unseen data, with considerably lower operational complexity and higher interpretability.

4.1.3 COMPARISON WITH BASELINE

In order to evaluate the robustness of the approaches in a realistic scenario, an additional test was performed using tenants that were not included during the training phase. This setup simulates the arrival of a new client to the platform, where no prior information has been used to train the model, and therefore represents the most demanding evaluation condition.

The best-performing supervised model, Random Forest, was compared against the enhanced Jaro-Winkler approach on this unseen dataset. As shown in Table 9, Random Forest achieves a slightly higher overall accuracy and recall, indicating a better ability to identify matching identities. However, the enhanced Jaro-Winkler approach obtains a higher precision, which is the most critical factor in this problem.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Enhanced Jaro-Winkler	0.9931	0.97	0.72	0.83
Random Forest	0.9948	0.94	0.81	0.87

Table 9. Comparison between the selected supervised model and the enhanced Jaro-Winkler approach on unseen tenants

From a business perspective, false positives have a significantly higher cost than false negatives, as they directly distort the productivity metrics provided by the platform. In this context, the higher precision of the enhanced Jaro-Winkler approach makes it a safer and more reliable option. Furthermore, the feature importance analysis shows that the most relevant variables in the Random Forest model are precisely the Jaro-Winkler similarity scores, suggesting that the supervised model is largely leveraging the same underlying information while adding limited value through additional features. Beyond the metrics, the operational cost of deploying a machine learning model is considerably higher than that of a deterministic algorithm: it requires serialising and loading a trained model, managing versioning, and periodically retraining as new tenants are onboarded. The enhanced Jaro-Winkler approach requires none of this, and any parameter adjustment can be made directly in code without retraining. Considering the trade-off between performance, interpretability, and operational complexity, the enhanced Jaro-Winkler approach is selected as the preferred solution for the production system.

The impact of the enhanced mapping system extends beyond classification metrics. Prior to this work, the manual validation workload associated with the mapping process represented a significant operational cost during client onboarding. On average, approximately four out of every twenty-five working days required for onboarding a new client were dedicated to reviewing and correcting developer mappings manually, accounting for around 16% of the total onboarding time. The enhanced system has reduced this to approximately one day, a 75% reduction in manual mapping effort. This figure is expected to improve further as the system accumulates operational experience with new tenants and the automatic matching rate continues to increase.

4.2 PROCESS 2: AUTOMATED REPORT GENERATION

The system was developed iteratively, with each version of the prompts refined based on feedback from the consultant team. Early versions of the reports presented two main issues: a tendency to include generic commentary not grounded in the specific data provided, and a tone that felt too technical for the intended audience. Both issues were addressed progressively through prompt refinement, tightening the instructions around which metrics to mention and explicitly specifying the target reader profile in the role and objective sections.

By the final iteration, the reports reached a quality level at which consultants were unable to consistently distinguish between manually written reports and automatically generated ones when presented with both side by side. This blind comparison, conducted using existing manually produced reports as reference, served as the primary qualitative validation of the system.

The LLM-as-a-judge evaluation complemented this human assessment by providing a more structured scoring across the report catalogue. Reports were scored on factual accuracy, that is whether all figures cited matched the underlying query data, on clarity and readability for a non-technical audience, and on tone consistency across different report types. All report types scored positively on factual accuracy, which was the most critical dimension given the requirement that the system never fabricate data. The bottom performer report, which includes a mandatory disclaimer reminding the consultant to verify data quality before delivery, was consistently rated as the most practically useful format by the consultant team.

From an efficiency standpoint, the automated pipeline reduced the time required to produce a complete individual developer report from an estimated average of 45 minutes manually to under two minutes, including query execution and generation time. This represents a reduction of over 95% in report production time, and more importantly, it removes the bottleneck that previously prevented consultants from generating reports at scale across large client tenants.

The system was tested across multiple Amazon Bedrock models, with selection criteria based on a combination of output quality, response latency, and cost per token. The final configuration used in production was chosen as the best balance across these three dimensions, with no single model dominating on all criteria simultaneously.

CHAPTER 5. CONCLUSION AND FUTURE WORKS

This project set out to automate two key analytical processes at Foreworth: the resolution of developer identities across Git repositories, and the generation of developer productivity reports using Large Language Models. Both objectives were met, and both systems have been integrated into or are ready for integration into Foreworth's production environment.

For the developer identity mapping process, the main contribution was an enhanced heuristic approach based on Jaro-Winkler similarity, extended with cross-field comparisons, name normalization, bot detection, and a dynamic similarity threshold. The system was validated against labelled data from real production tenants and achieved a precision of 0.97 on unseen data, which is the metric that matters most in this context given the high operational cost of false positives. Alongside the heuristic approach, supervised and unsupervised learning models were evaluated, with Random Forest achieving comparable accuracy and higher recall, but lower precision. However, the heuristic approach was selected for production due to its greater interpretability, lower operational complexity, reduction of false positives, and absence of retraining requirements. This decision reflects a broader engineering principle that runs throughout the project: a more complex model is not always a better production solution. The operational impact was substantial, the enhanced system reduced the manual mapping workload during client onboarding by 75%, from approximately four working days to one.

For the automated report generation process, the main contribution was a modular two-stage pipeline built on top of Amazon Bedrock, structured around explicit prompt design with role, context, and instruction sections. The system generates five report types covering commit activity, code quality, work routines, individual performance, and low-productivity cases. By the final iteration, consultants were unable to consistently distinguish between manually written and automatically generated reports in blind comparisons. From an efficiency standpoint, report generation time was reduced from an estimated 45 minutes per report to under two minutes, a reduction of over 95%, effectively removing the bottleneck that previously limited report delivery at scale.

Taken together, both processes demonstrate that well-designed, production-oriented automation can deliver measurable operational value without necessarily relying on the most complex available techniques. The emphasis throughout has been on reliability, explainability, and maintainability, qualities that matter as much as benchmark performance in a real production environment.

5.1 FUTURE WORKS

Several natural directions for future work emerge from the results and limitations identified in this project.

The most immediate extension for the identity mapping process is the integration of the supervised Random Forest model into production. The current heuristic system was chosen as the initial deployment solution due to its simplicity, higher precision and lower operational risk, but the machine learning approach is a natural next step once sufficient operational experience has been gathered and a larger volume of validated training data is available from new tenants. The feature importance analysis suggests that expanding the feature set beyond Jaro-Winkler-based similarities, for example, incorporating repository overlap or richer behavioral signals, could further improve recall without sacrificing precision.

For the report generation system, the most relevant direction is the expansion of the report catalogue. The pipeline is designed to be fully extensible through the frontend interface without requiring code changes, which makes adding new report types a relatively low-cost operation. As Foreworth's product evolves and new data dimensions become available, the system can grow accordingly. A related direction is the evaluation methodology: the current approach combines human review with LLM-as-a-judge scoring, but a more systematic benchmark, with standardised test cases and reproducible scoring criteria, would allow more rigorous comparisons between prompt versions and model choices as the system matures.

Finally, both processes could benefit from a more unified monitoring framework that tracks system performance over time as new tenants are onboarded. For the mapping system, this means monitoring the proportion of automatic matches, review cases, and false positives across new clients. For the report generation system, it means tracking consistency and factual accuracy as the underlying data and prompt configurations evolve.

CHAPTER 6. BIBLIOGRAPHY

[1] T. Fry, T. Dey, A. Karnauch, and A. Mockus, "A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits," in *Proc. 17th Int. Conf. Mining Software Repositories (MSR '20)*, Seoul, Republic of Korea, Oct. 2020. [Online]. Available: https://arxiv.org/abs/2003.08349

[2] S. Amreen, A. Mockus, C. Bogart, Y. Zhang, and R. Zaretzki, "ALFAA: Active Learning Fingerprint Based Anti-Aliasing for Correcting Developer Identity Errors in Version Control Data," *Empirical Software Engineering*, vol. 25, pp. 1136–1167, 2020. [Online]. Available: https://arxiv.org/pdf/1901.03363

[3] S. Schulhoff et al., "The Prompt Report: A Systematic Survey of Prompting Techniques," arXiv:2406.06608, 2024. [Online]. Available: https://arxiv.org/abs/2406.06608

[4] E. Fons, E. Kochkina, R. Kaur, Z. Zeng, B. Hlavaty, C. Smiley, S. Vyetrenko, and M. Veloso, "AI Analyst: Framework and Comprehensive Evaluation of Large Language Models for Financial Time Series Report Generation," arXiv:2507.00718, 2025. [Online]. Available: https://arxiv.org/pdf/2507.00718

[5] S. Dhuliawala et al., "Chain-of-Verification Reduces Hallucination in Large Language Models," arXiv:2309.11495, 2023. [Online]. Available: https://arxiv.org/abs/2309.11495