CIÊNCIAS EMPRESARIAIS
ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

ANDRÉ FILIPE PÉ-CURTO GUERREIRO

RELATÓRIO DE ESTÁGIO: MODERNIZAÇÃO DE SOLUÇÕES DE ENGENHARIA DE DADOS NO SETOR LOGÍSTICO

Relatório de estágio do Mestrado em Ciências de Dados para Empresas

ORIENTADOR

Professor Doutor David Alexandre Mendes da Silva Simões

Professora Doutora Maria da Graça Rodrigues Gomes da Costa

Dezembro de 2025

ANDRÉ FILIPE
PÉ-CURTO
GUERREIRO

RELATÓRIO DE ESTÁGIO: MODERNIZAÇÃO DE SOLUÇÕES DE ENGENHARIA DE DADOS NO SETOR LOGÍSTICO

JÚRI

Presidente: Professor Doutor Victor Manuel Meneses Barbosa, IPS
Orientador: Professor Doutor David Alexandre Mendes da Silva Simões, IPS
Vogal: Professor Doutor Hernâni Raul Vergueiro Monteiro Cidade Mourão, IPS

Dezembro 2025

Resumo

Este relatório descreve o estágio curricular realizado na MixMove entre novembro de 2024 e maio de 2025, no âmbito do Mestrado em Ciência de Dados. O principal foco foi a modernização de soluções de engenharia de dados aplicadas ao setor logístico, com especial ênfase na integração de sistemas heterogéneos, na automação de pipelines ETL e na aplicação de técnicas de *data masking*, alinhadas com os requisitos da norma ISO 27001.

Ao longo do estágio, os principais objetivos passaram pelo desenvolvimento de integrações robustas para clientes como a Volkswagen UK, Enfega, Freja e Rangel; pela modernização dos sistemas de validação de dados; pela automação dos processos de extração, transformação e carregamento de dados; e pela implementação de mecanismos de segurança e monitorização eficazes.

O estágio surgiu no contexto de uma infraestrutura tecnológica envelhecida, onde as validações se encontravam dispersas, havia necessidade de trabalhar com dados sensíveis em ambientes de teste, e existiam desafios relacionados com a escalabilidade dos sistemas, que gerem grandes volumes de dados distribuídos por mais de vinte tabelas operacionais.

A abordagem metodológica adotada combinou princípios de DataOps com a *framework Scrum*, através de ciclos iterativos de duas semanas. A recolha de informação baseou-se na análise de documentação técnica, análise de *logs* de desempenho e numa revisão sistemática da literatura relevante. O tratamento da informação incluiu engenharia reversa de sistemas obsoletos, validações em várias camadas com recurso à biblioteca FluentValidation, e a construção de pipelines ETL automatizados e escaláveis.

As fontes de dados abrangeram desde os sistemas operacionais da Volkswagen UK, até ficheiros Excel, Csv e Json, passando por APIs externas como a da Routyn e documentação de sistemas obsoletos.

Os resultados alcançados foram bastante positivos, tendo-se verificado uma redução de 85% no tempo de processamento, a eliminação total de erros manuais e uma melhoria significativa na qualidade dos dados processados. A aplicação de técnicas de *data masking* permitiu assegurar a conformidade com requisitos regulamentares, enquanto a nova arquitetura modular, baseada em microserviços, trouxe ganhos claros em termos de capacidade de manutenção e escalabilidade.

Palavras-chave: Engenharia de Dados, DataOps, Qualidade de Dados, ETL, Data Warehouse

Abstract

This report presents the curricular internship carried out at MixMove between November 2024 and May 2025, as part of the Master's in Data Science. The main focus was the modernization of data engineering solutions for the logistics sector, with special emphasis on the integration of heterogeneous systems, automation of ETL pipelines, and the implementation of data masking techniques in compliance with ISO 27001 standards.

Throughout the internship, the primary goals were to develop robust integrations for clients such as Volkswagen UK, Enfega, Freja, and Rangel; to modernize data validation systems; to automate extraction, transformation, and loading processes; and to implement effective security and monitoring mechanisms.

The project was carried out in the context of an outdated technological infrastructure, where validations were scattered, sensitive data had to be handled in test environments, and the systems faced scalability challenges due to high volumes of data spread across more than twenty operational tables.

The adopted methodological approach combined DataOps principles with the Scrum framework, through two-week iterative development cycles. Data collection involved technical documentation analysis, performance log inspection, and a systematic literature review. Information processing included reverse engineering of legacy systems, multi-layer validation using the FluentValidation library, and the construction of automated and scalable ETL pipelines.

The data sources ranged from Volkswagen UK’s operational systems to Excel, CSV, and JSON files, as well as external APIs such as Routyn and legacy system documentation.

The results were highly positive, showing an 85% reduction in processing time, complete elimination of manual errors, and a significant improvement in data quality. The implementation of data masking ensured regulatory compliance, while the new modular microservices-based architecture improved system maintainability and scalability.

Keywords: Data Engineering, DataOps, Data Quality, ETL, Data Warehouse

Lista de Símbolos e de siglas

API - *Application Programming Interface* (Interface de Programação de Aplicações)

CSV - *Comma-Separated Values* (Valores Separados por Vírgula)

DDD - *Domain-Driven Design* (Design Orientado ao Domínio)

ETL - *Extract, Transform, Load* (Extrair, Transformar, Carregar)

HTTP - *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)

JSON - *JavaScript Object Notation* (Notação de Objetos JavaScript)

PGP - *Pretty Good Privacy* (Privacidade Bastante Boa)

PTA - *Predicted Time of Arrival* (Tempo Previsto de Chegada)

PTD - *Predicted Time of Departure* (Tempo Previsto de Partida)

REST - *Representational State Transfer* (Transferência de Estado Representacional)

SFTP - *Secure File Transfer Protocol* (Protocolo de Transferência Segura de Arquivos)

SOLID - *Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion* (Princípios de design de software)

SQL - *Structured Query Language* (Linguagem de Consulta Estruturada)

SSCC - *Serial Shipping Container Code* (Código Serial de Contentor de Envio)

VWUK - Volkswagen United Kingdom (Volkswagen Reino Unido)

XML - *Extensible Markup Language* (Linguagem de Marcação Extensível)

Introdução

No contexto atual, caracterizado por uma crescente digitalização e pela criação exponencial de dados, a ciência de dados emerge como uma área fundamental para organizações que transformam informação em valor. O presente relatório descreve o estágio curricular realizado no âmbito do Mestrado em Ciência de Dados para Empresas, na empresa MixMove. Uma organização especializada em soluções de logística e gestão de transportes.

A MixMove posiciona-se como uma empresa inovadora no setor de logística, oferecendo plataformas que integram diferentes sistemas e fontes de dados para otimizar operações de transporte. O uso de dados para melhorar a eficiência e a sustentabilidade das cadeias por abastecimento, alinha-se perfeitamente com os objetivos da ciência de dados, tornando-a um ambiente ideal para aplicar os conhecimentos adquiridos durante o mestrado.

Durante o período de seis meses de estágio, entre novembro de 2024 e maio de 2025, houve a oportunidade de trabalhar no departamento de Desenvolvimento de Integrações, com foco na migração, desenvolvimento e manutenção de diversas integrações para clientes da empresa. Este departamento é responsável pela criação e manutenção de soluções que permitem a interoperabilidade entre os vários sistemas utilizados pelos clientes e a plataforma da MixMove, de maneira a funcionar como elo de ligação entre os dados provenientes de diferentes fontes e a plataforma central.

O principal objetivo deste estágio consistiu, no desenvolvimento e modernização de soluções de engenharia de dados, com foco em integrações de sistemas, automação de pipelines e garantia de qualidade de dados. Durante este período, existiu a oportunidade de trabalhar em projetos reais com impacto significativo, sendo aplicados conceitos teóricos em contextos práticos e sendo superados desafios que exigiram adaptação e aprendizagem contínua.

Entre as atividades desenvolvidas, destacam-se: a criação de uma solução completa de integração de dados para a Volkswagen UK; o desenvolvimento e migração de integrações para sistemas de reservas, com especial atenção à integração da Enfega; a implementação de um novo sistema de validação de dados, baseado na biblioteca FluentValidation; a automação de pipelines de dados para processamento eficiente de informações de rotas de transporte; o desenvolvimento de APIs e serviços web para facilitar a comunicação entre diferentes sistemas; e a elaboração de documentação técnica abrangente.

O presente relatório está estruturado em quatro capítulos principais. Após esta introdução, será apresentado o enquadramento teórico, onde serão discutidos os conceitos fundamentais

de engenharia de dados, qualidade de dados, DataOps e outras metodologias relevantes. No segundo capítulo, são descritos os objetivos e a metodologia utilizada durante o estágio. No terceiro capítulo, são detalhadas as atividades desenvolvidas durante o estágio, incluindo os desafios enfrentados e as soluções implementadas. O quarto capítulo é dedicado à análise crítica da experiência, destacando as aprendizagens e competências adquiridas. Por fim, são apresentadas as conclusões do estágio e sugeridos possíveis desenvolvimentos futuros.

A relevância desta experiência para a área de Ciência de Dados é muito importante, uma vez que o trabalho desenvolvido abrangeu várias dimensões cruciais neste domínio: desde a aquisição e validação de dados de diferentes fontes, passando pelo seu processamento e transformação, até à disponibilização em formatos adequados para análise e visualização. Adicionalmente, o contexto empresarial contribuiu para compreender os princípios teóricos da ciência de dados que são aplicados em ambientes de produção reais, com todas as complexidades e restrições inerentes.

Como referem Provost e Fawcett (2013), a ciência de dados não se limita apenas à aplicação de algoritmos avançados, mas abrange todo o ciclo de vida dos dados, desde a sua criação, até à sua utilização final para a criação de valor. Neste sentido, os projetos desenvolvidos durante o estágio, proporcionaram uma visão abrangente do papel do cientista de dados no contexto organizacional, com particular ênfase na componente de engenharia de dados, fundamental para garantir a qualidade e disponibilidade dos dados para análises subsequentes.

I- Enquadramento Teórico

I.1 Engenharia de Dados e Pipelines

A engenharia de dados é, atualmente, uma das áreas mais importantes no mundo da ciência de dados. Funciona como a espinha dorsal que suporta todo o processo analítico, desde a recolha de dados até à aplicação de modelos de *machine learning*. Sem uma infraestrutura bem desenhada, os dados ficam dispersos, desorganizados e muitas vezes, inúteis para quem precisa de tomar decisões com base neles.

Segundo Kleppmann (2017), o grande objetivo da engenharia de dados é, construir e manter sistemas que assegurem que os dados são recolhidos, armazenados, processados e disponibilizados de forma eficiente e fiável. Estes sistemas, designam-se normalmente de *pipelines* de dados e funcionam como "autoestradas de informação", que transportam os dados desde as suas fontes, sejam bases de dados, *APIs* ou sensores, até aos locais onde são analisados ou transformados em *insights*.

Na prática, isto quer dizer que, por trás de cada *dashboard*, relatório ou modelo preditivo, existe um trabalho invisível, mas essencial da engenharia de dados. Esta área assegura que os dados chegam no formato certo, no momento certo e com a qualidade necessária para que cientistas de dados e analistas possam fazer o seu trabalho sem preocupações técnicas.

Na atualidade, os *pipelines* de dados constituem sistemas cada vez mais complexos, sobretudo porque as empresas lidam com uma enorme variedade de fontes e formatos, desde bases de dados tradicionais até *APIs* externas, ficheiros distribuídos e fluxos em tempo real. A própria AWS (s.d.) reforça essa ideia, destacando que essa diversidade exige soluções para manter a integridade e consistência dos dados durante todo o processo. O desafio é ainda maior quando falamos de processamento em tempo real, que obriga a arquiteturas capazes de lidar não só com dados armazenados (*batch*), mas também com dados em tempo real (*streaming*).

Construir um *pipeline* eficiente não é só uma questão técnica, é também uma questão de arquitetura modular. Como referem De Bie et al. (2021), a modularidade é fundamental: decomposição do pipeline em componentes independentes e reutilizáveis, facilita a manutenção e a evolução do sistema. Além disso, é essencial ter mecanismos de monitorização contínua e tratamento de falhas, já que estes sistemas operam em ambientes distribuídos, onde problemas de rede ou inconsistências nos dados podem surgir a qualquer momento.

As integrações entre sistemas funcionam como pipelines que não só transferem dados, mas também os transformam e sincronizam, eliminando incompatibilidades de forma a garantir que a informação faz sentido no destino. Este processo é fundamental para assegurar que os dados de diferentes origens chegam de forma coerente, fiável e útil.

I.2 Qualidade de Dados e Validação

Num mundo onde os dados alimentam decisões críticas, encontrar um equilíbrio entre velocidade de entrega e integridade da informação torna-se um desafio fundamental. A qualidade dos dados constitui uma prioridade essencial, pois de que serve uma análise sofisticada se os dados de entrada forem incompletos ou imprecisos?

Batini e Scannapieco (2016) definem qualidade de dados como "a adequação dos dados ao seu propósito". Um conceito simples, mas que abrange dimensões críticas como precisão, integridade, consistência, completude, atualidade e validade. Esta definição multidimensional da qualidade dos dados reflete a complexidade inerente à avaliação e manutenção de padrões elevados de qualidade em sistemas de informação modernos.

A precisão refere-se ao grau em que os dados refletem corretamente a realidade que pretendem representar. A integridade assegura que os dados mantêm a sua estrutura e relações lógicas ao longo do tempo. A consistência garante que os dados não apresentam contradições internas ou com outras fontes de dados. A completude mede se todos os valores necessários estão presentes, enquanto a atualidade avalia se os dados são suficientemente recentes para o propósito pretendido.

As abordagens multifacetadas para validação de dados alinham-se com as recomendações de Schelter et al. (2018), que defendem a validação contínua como forma de detetar anomalias antes que se propaguem pelo sistema. Estes autores propõem um *framework* de validação automática que inclui verificações estatísticas, regras de negócio e deteção de anomalias baseada em padrões históricos.

Nas integrações entre sistemas, a qualidade dos dados é ainda mais sensível. Quando informações transitam entre plataformas distintas, o risco de incompatibilidades aumenta significativamente. Dasu e Johnson (2003) comparam este processo a uma linha de montagem: "se um componente defeituoso entra no sistema, o produto final sairá comprometido". Esta analogia sublinha a importância de implementar controlos de qualidade em cada etapa do pipeline, uma prática que não só mitiga riscos como também reforça a confiança dos utilizadores finais nos dados.

A validação de dados pode ser classificada em várias categorias: validação sintática (formato, tipo de dados), validação semântica (significado e contexto), validação de domínio (valores dentro de intervalos aceitáveis) e validação de consistência (relações entre diferentes campos ou registos). Cada categoria requer técnicas específicas e deve ser aplicada em diferentes pontos do pipeline de dados.

I.3 DataOps, Integração de Sistemas e APIs

Na era atual, a gestão do ciclo de vida dos dados de forma eficaz, exige muito mais do que ferramentas avançadas, requer uma fusão eficaz de estratégias ágeis, automação e uma visão integrada dos sistemas. Foi neste contexto que, durante o estágio, trabalhou-se com *DataOps*, uma metodologia que combina a agilidade do *DevOps* com as particularidades do ciclo de vida dos dados.

De acordo com Ereth (2018), o *DataOps* pode ser definido como "um conjunto de práticas, processos e tecnologias que integram uma visão orientada para os dados com automação e metodologias da engenharia de *software*". Esta abordagem não só permite melhorar a qualidade e a velocidade do processamento, como também reforça a colaboração entre equipas técnicas e de negócio. Algo muito importante considerando a complexidade cada vez maior dos sistemas de dados nas empresas.

O DataOps fundamenta-se em vários princípios: colaboração contínua entre equipas, automação de processos, monitorização em tempo real, gestão de versões para dados e código e testes automatizados. Estes princípios visam reduzir o tempo entre a identificação de uma necessidade de dados e a entrega de insights acionáveis.

A automação, conforme destacado por Atwal (2019), constitui um pilar fundamental do DataOps. Ao implementar pipelines que processam automaticamente ficheiros, validam conteúdos e notificam utilizadores sobre inconsistências, é possível reduzir significativamente erros humanos e acelerar o ciclo de desenvolvimento. Esta prática alinha-se perfeitamente com os princípios de DataOps e permite às equipas evoluir para arquiteturas baseadas em microserviços.

A arquitetura de microserviços, tal como proposta por Gregor, Chandra Kruse e Seidel (2020), oferece uma modularidade que se revela particularmente vantajosa para garantir escalabilidade e facilitar a manutenção do sistema. Nesta abordagem, cada serviço é responsável por uma função específica e pode ser desenvolvido, implementado e escalado independentemente.

A integração de sistemas heterogéneos constitui um dos desafios mais significativos em ambientes empresariais modernos. Massé (2011) alerta precisamente para esta dificuldade, especialmente em contextos com sistemas obsoletos. As APIs RESTful, cuja eficácia é bem documentada por Richardson e Ruby (2007), tornaram-se elementos-chave para superar este obstáculo, permitindo uma comunicação segura e escalável entre sistemas distribuídos.

As APIs (*Application Programming Interfaces*) funcionam como contratos bem definidos que especificam como diferentes componentes de software podem interagir. O estilo arquitetural REST (*Representational State Transfer*) oferece princípios para o design de APIs web escaláveis, incluindo a utilização de métodos HTTP padronizados, URLs como identificadores de recursos, e representações de estado sem sessão.

A monitorização contínua, tal como defendida por Stemplinger, Heinisch e Bodendorf (2023), constitui um pilar essencial das operações de dados modernas. Estes autores propõem *frameworks* de monitorização que não só detetam precocemente problemas nos pipelines de dados, como também identificam oportunidades de otimização através de métricas de desempenho e qualidade.

I.4 Data Warehousing e ETL

Os *data warehouses* e os processos ETL (*Extract, Transform, Load*) constituem componentes fundamentais na gestão moderna de dados empresariais. Kimball e Ross (2013) descrevem o *data warehouse* como um repositório que reúne dados de diversas fontes, organizando-os de forma a facilitar análises e apoiar a tomada de decisão estratégica.

A arquitetura de *data warehousing* tradicional baseia-se no conceito de modelação dimensional, que organiza os dados em tabelas de factos e dimensões. As tabelas de factos contêm medidas numéricas do negócio, enquanto as tabelas de dimensões fornecem o contexto descritivo para essas medidas. Esta abordagem facilita consultas analíticas complexas e oferece desempenho otimizado para relatórios e análises.

Inmon (2005) propõe uma abordagem alternativa, defendendo a criação de um *data warehouse* normalizado que serve como repositório central de dados integrados, a partir do qual são criados *data marts* específicos para diferentes áreas de negócio. Esta metodologia, conhecida como "*top-down*", contrasta com a abordagem "*bottom-up*" de Kimball, que começa pelos *data marts* e os integra posteriormente.

Os processos ETL desempenham um papel crucial na arquitetura de *data warehousing*. A fase de extração envolve a recolha de dados de múltiplas fontes heterogéneas, incluindo bases de dados transacionais, ficheiros planos, APIs web, e sistemas obsoletos. A transformação inclui limpeza de dados, validação, conversão de formatos, cálculos derivados, e aplicação de regras de negócio. O carregamento refere-se ao processo de inserir os dados transformados no *data warehouse*, que pode ser incremental ou completo.

A evolução dos paradigmas ETL levou ao surgimento do ELT (*Extract, Load, Transform*), onde os dados são primeiro carregados no destino e depois transformados, fazendo uso da capacidade de processamento do sistema de destino. Esta abordagem é particularmente adequada para ambientes de *big data* e *cloud computing*, onde o armazenamento é económico e a capacidade de processamento pode ser escalada conforme necessário.

A implementação de soluções ETL modernas envolve considerações sobre processamento em lote versus tempo real, gestão de metadados, versionamento de esquemas, e recuperação de falhas. Ferramentas como Apache Airflow, Talend, e Azure Data Factory oferecem capacidades avançadas para orquestração de fluxos de trabalho complexos.

I.5 Monitorização e Governação de Dados

A governação de dados constitui um *framework* abrangente que define políticas, procedimentos e responsabilidades para garantir a gestão eficaz dos dados ao longo do seu ciclo de vida. Weber, Otto e Österle (2009) destacam que a verdadeira governação de dados vai além do mero armazenamento, envolvendo uma gestão ativa da disponibilidade, usabilidade, integridade e segurança dos dados.

O *framework* de governação de dados tipicamente inclui várias componentes: políticas de dados que definem regras e standards, arquitetura de dados que estabelece a estrutura e relações, gestão de *metadados* que documenta o significado e linhagem dos dados, qualidade de dados que assegura precisão e consistência, e gestão do ciclo de vida que controla a criação, manutenção e eliminação de dados.

A implementação de sistemas abrangentes de *logging* e monitorização contínua permite acompanhar em tempo real todo o fluxo de dados. Esta abordagem revela-se indispensável para identificar precocemente diversos tipos de problemas: desde falhas pontuais em integrações até inconsistências mais subtis nos processos de transformação de dados.

Ladley (2019) enfatiza que uma governação eficaz precisa de mecanismos proactivos que antecipam problemas em vez de apenas os corrigirem posteriormente. Estes mecanismos incluem alertas automáticos, *dashboards* de monitorização, auditorias regulares, e processos de escalamento bem definidos.

A monitorização de dados pode ser categorizada em várias dimensões: monitorização técnica (desempenho, disponibilidade, erros), monitorização de qualidade (precisão, completude, consistência), monitorização de utilização (padrões de acesso, volume de consultas), e monitorização de conformidade (aderência a políticas e regulamentos).

Os sistemas modernos de governação de dados incorporam igualmente aspetos de machine learning para deteção automática de anomalias, classificação automática de dados sensíveis, e recomendações de melhorias na qualidade dos dados. Estas capacidades inteligentes permitem uma gestão mais eficiente e proactiva dos ativos de dados organizacionais.

I.6 Metodologias de Desenvolvimento

As metodologias ágeis revolucionaram o desenvolvimento de software e, mais recentemente, têm sido adaptadas para projetos de dados e análise. O *framework Scrum*, como referido por Sutherland e Schwaber (2017) no *Scrum Guide*, organiza o trabalho em ciclos iterativos e incrementais (sprints), incentivando um feedback contínuo que permite adaptações rápidas às necessidades emergentes.

O *Scrum* baseia-se em três pilares fundamentais: transparência (todos os *aspetos* do processo devem ser visíveis), inspeção (artefactos e progresso devem ser inspecionados frequentemente), e adaptação (ajustes devem ser feitos quando desvios são detetados). Estes pilares são suportados por cinco valores: compromisso, coragem, foco, abertura e respeito.

A aplicação de metodologias ágeis a projetos de dados apresenta desafios únicos. Ao contrário do desenvolvimento de software tradicional, os projetos de dados frequentemente envolvem exploração e descoberta, onde os requisitos podem não estar completamente definidos à partida. Esta incerteza requer adaptações específicas dos *frameworks* ágeis.

As reuniões regulares do *Scrum*, incluindo *daily stand-ups*, *sprint plannings* e retrospectivas, criam um ritmo de trabalho consistente e melhoram substancialmente a comunicação entre os membros da equipa. A natureza colaborativa do *Scrum* mostra-se particularmente valiosa em projetos de integração de dados e sistemas.

Lean Analytics, conforme descrito por Croll e Yoskovitz (2013), oferece uma abordagem complementar focada na medição e aprendizagem contínua. Esta metodologia enfatiza a importância de definir métricas-chave, realizar experiências controladas, e tomar decisões baseadas em dados.

A metodologia CRISP-DM (*Cross-Industry Standard Process for Data Mining*) fornece um *framework* estruturado para projetos de ciência de dados, que inclui fases de compreensão do negócio, compreensão dos dados, preparação dos dados, modelação, avaliação e implementação.

I.7 Segurança e Governação da Informação

A segurança e a governação da informação constituem processos fundamentais para qualquer sistema que lide com dados, sobretudo em ambientes empresariais onde a informação pode ter valor estratégico ou conter detalhes sensíveis. Smallwood (2019) define a governação de dados como um conjunto de regras, processos e mecanismos que garantem que os dados são geridos de forma coerente e adequada ao longo de todo o seu ciclo de vida.

No campo da ciência de dados, a governação ganha ainda mais relevância, uma vez que está diretamente ligada à qualidade, consistência e fiabilidade da informação utilizada. Uma boa governação não deve ser vista como um entrave à inovação, mas sim como uma base essencial para garantir que os dados utilizados são confiáveis e cumprem os seus propósitos.

A segurança da informação, como parte integrante da governação, concentra-se em proteger os dados contra acessos indevidos, perdas ou alterações não autorizadas. Calder (2021) defende que uma estratégia sólida de segurança deve assentar em três pilares fundamentais:

- **Confidencialidade** – apenas utilizadores autorizados podem aceder aos dados
- **Integridade** – os dados não podem ser alterados sem permissão
- **Disponibilidade** – a informação deve estar acessível quando necessária

Estes princípios, conhecidos como a tríade CIA (*Confidentiality, Integrity, Availability*), formam a base conceptual para a maioria dos *frameworks* de segurança da informação. A implementação prática destes princípios requer uma abordagem em camadas, que incluem controlos físicos, técnicos e administrativos.

A gestão de identidade e acesso (IAM - *Identity and Access Management*) constitui um componente crítico da segurança da informação. Esta disciplina envolve a definição de políticas para autenticação (verificação de identidade), autorização (concessão de

permissões), e auditoria (registo de atividades). Sistemas IAM modernos suportam conceitos como *single sign-on* (SSO), autenticação *multi-factor* (MFA), e princípio do menor privilégio.

Em ambientes de integração de dados, Bertino e Ferrari (2018) destacam que é crucial implementar controlos de acesso rigorosos e registos de auditoria detalhados. Estas medidas incluem autenticação baseada em *tokens*, encriptação de dados sensíveis em trânsito e em repouso, e monitorização contínua de atividades suspeitas.

A conformidade regulamentar adiciona uma camada adicional de complexidade à segurança da informação. Regulamentos como o RGPD (Regulamento Geral sobre a Proteção de Dados) na União Europeia, HIPAA nos Estados Unidos, e SOX para empresas cotadas impõem requisitos específicos para a proteção e gestão de dados sensíveis.

I.8 Data Masking e Segurança de Dados

O *data masking* é uma técnica essencial no âmbito da segurança e proteção da informação, sobretudo em contextos empresariais que lidam com dados sensíveis. Bertino e Ferrari (2018) definem esta técnica como a substituição de dados reais por valores fictícios, mas plausíveis, mantendo os formatos e propriedades estatísticas dos dados originais.

O objetivo primário do *data masking* é permitir a utilização de dados em ambientes não produtivos, como desenvolvimento ou teste, sem comprometer a privacidade ou a segurança da informação. Esta prática é fundamental para organizações que precisam de dados realistas para testes, mas que devem proteger informações sensíveis de clientes, empregados ou operações comerciais.

No contexto da Ciência de Dados, o *data masking* assume particular importância quando se trabalha com dados pessoais, financeiros ou estratégicos. Mohanty, Jagadeesh e Srivatsa (2013) observam que a aplicação destas técnicas não representa apenas uma boa prática, mas constitui uma exigência imposta por várias regulamentações de proteção de dados.

A norma ISO 27001, referida por Calder (2021), define um conjunto de diretrizes rigorosas para a gestão da segurança da informação, que incluem controlos específicos para proteger dados utilizados em ambientes de teste e desenvolvimento. Esta norma exige que os dados nesses contextos sejam cuidadosamente selecionados, devidamente protegidos e controlados.

Existem diversas técnicas de *data masking*, cada uma adequada a diferentes tipos de dados e níveis de sensibilidade:

1. **Substituição** – substituição dos valores reais por dados fictícios provenientes de dicionários predefinidos (como nomes, moradas, entre outros)
2. **Embaralhamento (*shuffling*)** – reorganização dos valores dentro da mesma coluna, mantendo a distribuição estatística, mas partindo a correspondência com outras colunas
3. **Encriptação** – transformação dos dados através de algoritmos que apenas permitem a reversão com uma chave específica
4. **Tokenização** – substituição dos dados sensíveis por *tokens* sem qualquer significado intrínseco, mantendo o mapeamento original num local seguro
5. **Mascaramento parcial** – ocultação parcial da informação sensível (por exemplo, apresentar um número de cartão como "4*** **** **** 1234")
6. **Generalização** – redução da complexidade dos dados, como substituir uma morada completa apenas pela cidade
7. **Adição de ruído** – introdução de variações aleatórias pequenas, mas consistentes nos dados numéricos
8. **Síntese de dados** – gerar conjuntos de dados completamente artificiais que preservam as características estatísticas dos dados originais

A seleção da técnica apropriada depende de vários fatores: o tipo de dados (numéricos, categóricos, texto livre), o nível de sensibilidade, os requisitos de preservação de relações entre campos, e as necessidades específicas dos utilizadores dos dados mascarados.

Ferramentas modernas de *data masking* oferecem capacidades avançadas como mascaramento consistente (o mesmo valor real é sempre mascarado para o mesmo valor fictício), preservação de integridade referencial, e mascaramento condicional baseado em regras de negócio complexas.

A implementação eficaz de *data masking* requer uma compreensão profunda dos dados e dos seus padrões de utilização, bem como uma análise cuidadosa dos riscos de re-identificação. Técnicas de mascaramento inadequadas podem criar vulnerabilidades de segurança ou tornar os dados inúteis para os propósitos pretendidos.

II- Objetivos e Metodologia

II.1 Objetivos do Estágio

O presente estágio teve como propósito central desenvolver e modernizar soluções de engenharia de dados na empresa MixMove, com foco em integrações de sistemas, automação de pipelines e garantia de qualidade de dados.

O primeiro objetivo enfrentado consistiu em criar e implementar soluções de integração de dados para clientes específicos, como a Volkswagen UK e a Enfega. Este objetivo envolveu o desenvolvimento de sistemas capazes de extrair, processar e sincronizar dados provenientes de múltiplas fontes heterogéneas, garantindo a compatibilidade entre diferentes formatos e estruturas de dados. A implementação destas soluções exigiu uma análise detalhada dos requisitos específicos de cada cliente, bem como a criação de mecanismos de transferência e transformação de dados.

O segundo objetivo focou-se na modernização de sistemas de validação de dados existentes, melhorando a qualidade e confiabilidade das informações processadas. Esta modernização implicou a substituição de métodos de validação dispersos e inconsistentes por um sistema centralizado e estruturado, capaz de detetar e corrigir erros de forma automática. O objetivo incluía também a implementação de múltiplas camadas de validação, desde verificações básicas de formato até validações semânticas complexas.

O terceiro objetivo centrou-se no desenvolvimento de APIs RESTful e serviços web para facilitar a comunicação entre diferentes sistemas. Este trabalho envolveu a conceção e implementação de interfaces padronizadas que permitissem a interoperabilidade entre plataformas distintas, seguindo as melhores práticas de design de APIs. O objetivo incluía também a documentação completa destas interfaces, garantindo a sua facilidade de utilização e manutenção a longo prazo.

O quarto objetivo visou automatizar processos de extração, transformação e carregamento de dados (ETL). Esta automatização tinha como propósito reduzir significativamente a intervenção manual nos fluxos de dados, minimizando erros humanos e aumentando a eficiência operacional. O objetivo incluía a criação de pipelines que pudessem processar grandes volumes de dados de forma escalável e fiável, com mecanismos adequados de tratamento de falhas e recuperação.

O quinto objetivo procurou estabelecer mecanismos de monitorização e validação de dados. Este objetivo envolveu a implementação de sistemas de *logging* detalhados, alertas automáticos e *dashboards* de monitorização que permitissem acompanhar em tempo real o

estado dos processos de dados. A monitorização incluía também a definição de métricas de qualidade e indicadores de desempenho que facilitassem a identificação proactiva de problemas.

O sexto objetivo centrou-se na elaboração de documentação técnica para suportar a manutenção das soluções desenvolvidas. Este trabalho incluía a criação de documentação de arquitetura, manuais de utilizador, guias de instalação e configuração, bem como documentação de APIs. O objetivo era garantir que as soluções desenvolvidas pudessem ser facilmente compreendidas, mantidas e evoluídas por outros membros da equipa.

O sétimo e último objetivo visou aplicar metodologias ágeis no contexto de projetos de engenharia de dados. Este objetivo envolveu a adaptação de práticas do *Scrum* para o desenvolvimento de soluções de dados, incluindo a organização do trabalho em sprints, a realização de reuniões regulares de acompanhamento e a implementação de ciclos de feedback contínuo. A aplicação destas metodologias tinha como propósito aumentar a eficiência do desenvolvimento e garantir o alinhamento constante com as necessidades dos *stakeholders*.

II.2 Metodologia

Para atingir os objetivos propostos, adotou-se uma abordagem metodológica mista, combinando elementos de metodologias ágeis como o *Scrum* (Sutherland & Schwaber, 2017) com práticas específicas de engenharia de dados e *DevOps*.

O estágio foi realizado no departamento de Desenvolvimento de Integrações da MixMove, durante um período de seis meses (novembro de 2024 a maio de 2025), onde houve envolvimento em diversos projetos de integração de dados e sistemas.

Fundamentação da Abordagem Metodológica

A escolha de uma abordagem metodológica mista justifica-se pela natureza complexa e multifacetada dos projetos de engenharia de dados em ambiente empresarial. Esta abordagem permite combinar as vantagens de diferentes paradigmas, proporcionando uma compreensão mais holística dos fenómenos estudados. No contexto da engenharia de dados, esta metodologia revela-se particularmente adequada, uma vez que permite integrar métodos quantitativos de análise de desempenho com métodos qualitativos de avaliação de requisitos e necessidades dos utilizadores.

A fundamentação teórica da abordagem adotada baseia-se nos princípios da investigação-ação, caracterizada pela integração entre teoria e prática, onde o investigador não é um

observador externo, mas participa ativamente no processo de mudança e melhoria. Esta abordagem permitiu não apenas observar e analisar os processos existentes, mas também implementar melhorias e avaliar o seu impacto em tempo real.

A metodologia adotada alinha-se também com os princípios do *Design Science Research*, particularmente adequado para investigação em sistemas de informação, pois foca na criação e avaliação de artefactos inovadores que resolvem problemas organizacionais identificados. Os artefactos desenvolvidos incluem sistemas de integração, APIs, pipelines de dados e *frameworks* de validação.

Estratégia de Investigação

A estratégia de investigação seguiu uma abordagem de estudo de caso múltiplo. Cada integração desenvolvida (Volkswagen UK, Enfega, Freja, Rangel) constituiu um caso de estudo individual, permitindo a análise comparativa de diferentes contextos, requisitos e soluções técnicas. Esta abordagem possibilitou a identificação de padrões comuns e a generalização de boas práticas aplicáveis a contextos similares.

A estratégia incorporou também elementos de investigação participativa, onde o investigador assume simultaneamente os papéis de observador e participante ativo no processo de desenvolvimento. Essa combinação de papéis enriquece significativamente a compreensão dos fenómenos estudados e aumenta a validade dos resultados obtidos.

Fase de Diagnóstico e Planeamento

Esta fase inicial baseou-se numa metodologia estruturada de análise de requisitos que enfatiza a importância de uma compreensão profunda das necessidades dos *stakeholders* antes de qualquer desenvolvimento técnico. A análise das necessidades de integração dos clientes seguiu uma abordagem que incluiu entrevistas com utilizadores-chave, análise documental de sistemas existentes e observação direta de processos operacionais.

O levantamento dos requisitos técnicos e funcionais baseou-se na distinção entre requisitos funcionais (o que o sistema deve fazer) e não-funcionais (como o sistema deve preformar). Esta distinção revelou-se crucial para garantir que as soluções desenvolvidas não apenas satisfaziam as necessidades funcionais, mas também cumpriam critérios de qualidade como desempenho, segurança e capacidade de manutenção.

A definição da arquitetura de solução seguiu os princípios do *Domain-Driven Design*, conforme proposto por Evans (2003), que enfatizam a importância de alinhar a estrutura técnica com o domínio de negócio. Esta abordagem facilitou a comunicação entre equipas técnicas e de

negócio, garantindo que as soluções desenvolvidas refletiam adequadamente os processos e terminologias organizacionais.

A identificação de riscos e planeamento de mitigação seguiu uma abordagem sistemática que incluiu a identificação proactiva de riscos técnicos, organizacionais e temporais, bem como a definição de estratégias específicas de mitigação para cada categoria de risco identificado.

Fase de Desenvolvimento e Implementação

O desenvolvimento iterativo seguindo ciclos de sprint de duas semanas alinha-se com os princípios fundamentais do desenvolvimento ágil. Esta abordagem permite adaptação contínua às necessidades emergentes e facilita a deteção precoce de problemas técnicos ou de alinhamento com requisitos.

As reuniões diárias (*daily stand-ups*) para alinhamento do progresso e identificar bloqueios seguem as práticas recomendadas pelo *Scrum Guide* (Sutherland & Schwaber, 2017). Estas reuniões não apenas facilitam a coordenação da equipa, mas também promovem a transparência e a identificação proactiva de impedimentos ao progresso.

As revisões de código através de *pull requests*, garantindo qualidade e consistência do código, baseiam-se nas melhores práticas de desenvolvimento colaborativo. Esta prática não apenas melhora a qualidade técnica do código, mas também facilita a partilha de conhecimento entre membros da equipa.

Os testes contínuos, incluindo testes unitários, de integração e funcionais, seguem uma abordagem estratificada que garante cobertura adequada de testes em diferentes níveis de granularidade, desde componentes individuais até fluxos completos de sistema.

Fase de Validação e Refinamento

A validação das soluções implementadas junto aos *stakeholders* seguiu uma metodologia de avaliação participativa. Esta abordagem reconhece que a validação efetiva de sistemas de informação requer não apenas verificação técnica, mas também confirmação de que as soluções atendem às necessidades reais dos utilizadores finais.

O refinamento e ajustes com base no feedback recebido incorporaram princípios de melhoria contínua. Esta abordagem enfatiza a importância de ciclos rápidos de feedback e melhoria incremental.

A monitorização do desempenho das integrações em ambiente de produção baseou-se em práticas sistemáticas que garantem que os sistemas não apenas funcionam corretamente após a implementação, mas mantêm níveis adequados de desempenho e disponibilidade ao longo do tempo.

Técnicas de Recolha e Análise de Dados

A análise documental de especificações técnicas e manuais de sistemas obsoletos seguiu uma metodologia sistemática que permitiu extrair informação relevante de documentação frequentemente fragmentada e inconsistente, característica comum de sistemas legados.

A análise de *logs* e métricas de desempenho para avaliação das soluções implementadas baseou-se em técnicas de análise de dados operacionais. Esta análise quantitativa forneceu evidência objetiva da eficácia das soluções implementadas.

A revisão sistemática de literatura para fundamentar as decisões técnicas seguiu diretrizes rigorosas para revisões em engenharia de software. Esta abordagem garantiu que as decisões técnicas fossem fundamentadas em evidência científica sólida.

Ferramentas e Tecnologias Utilizadas

A seleção das ferramentas e tecnologias utilizadas baseou-se numa análise comparativa que considerou fatores como maturidade tecnológica, suporte da comunidade, adequação aos requisitos específicos, e alinhamento com a arquitetura organizacional existente.

Desenvolvimento: .NET Core, C#, Entity Framework, Dapper

APIs: ASP.NET Core, Swagger/OpenAPI

Base de dados: SQL Server, Azure Synapse Analytics

ETL e processamento de dados: Azure Data Factory, bibliotecas para processamento de Excel e CSV

Versionamento e CI/CD: Git, Azure DevOps

Validação de dados: FluentValidation

Segurança: Autenticação baseada em tokens, criptografia PGP, SFTP

Considerações de Qualidade

O estágio seguiu princípios rigorosos, incluindo confidencialidade de dados organizacionais, respeito pela propriedade intelectual, e transparência na comunicação de resultados. Todos os dados utilizados foram anonimizados quando necessário, tendo as soluções desenvolvidas respeitado políticas organizacionais de segurança e privacidade.

A qualidade da investigação foi assegurada através de triangulação metodológica, utilizando múltiplas fontes de dados e métodos de validação. A credibilidade dos resultados foi reforçada

através de validação por pares e feedback contínuo de supervisores académicos e organizacionais.

Esta metodologia permitiu não só desenvolver as soluções técnicas necessárias, mas também garantir a qualidade, capacidade de manutenção e alinhamento com as necessidades de negócio, aplicando na prática os conceitos teóricos da engenharia de dados e ciência de dados estudados durante o mestrado.

III- Atividades Desenvolvidas

Este capítulo apresenta de forma detalhada as principais atividades desenvolvidas durante o estágio na MixMove, abrangendo quatro grandes áreas de intervenção: integração de dados para clientes específicos, modernização de sistemas de validação, automação de pipelines de dados, e desenvolvimento de APIs e serviços web. Cada secção descreve os desafios enfrentados, as soluções implementadas e os resultados alcançados, demonstrando a aplicação prática dos conceitos teóricos da engenharia de dados em contexto empresarial real.

III.1 Integração de Dados para a Volkswagen UK

Um dos projetos mais desafiadores foi o desenvolvimento de uma solução de integração e processamento de dados para a Volkswagen UK (VWUK).

Este projeto foi essencial para modernizar as operações logísticas da empresa, tendo sido criada uma infraestrutura que permitisse extrair, processar e disponibilizar dados de forma eficiente.

Contexto e Desafios

A Volkswagen UK, necessitava uma solução que integrasse dados de várias fontes operacionais num *data warehouse* centralizado, facilitando análises avançadas e decisões estratégicas. Os principais desafios eram:

- **Dispersão de dados**: Informações espalhadas por mais de 20 tabelas operacionais, cada uma com estruturas diferentes.
- **Segurança e conformidade**: Dados sensíveis que exigiam proteção rigorosa, seguindo normas internas e regulamentos.
- **Volume e escalabilidade**: Necessidade de processar grandes quantidades de dados sem comprometer a performance.
- **Relatórios periódicos**: Gerar automaticamente os relatórios em formatos específicos, sem margem para erros.
- **Integração com sistemas obsoletos**: Compatibilidade com plataformas antigas.

A Figura 1 apresenta o diagrama da arquitetura implementada, ilustrando a distribuição das tabelas operacionais da Volkswagen UK e os fluxos de dados estabelecidos entre os diferentes componentes do sistema.

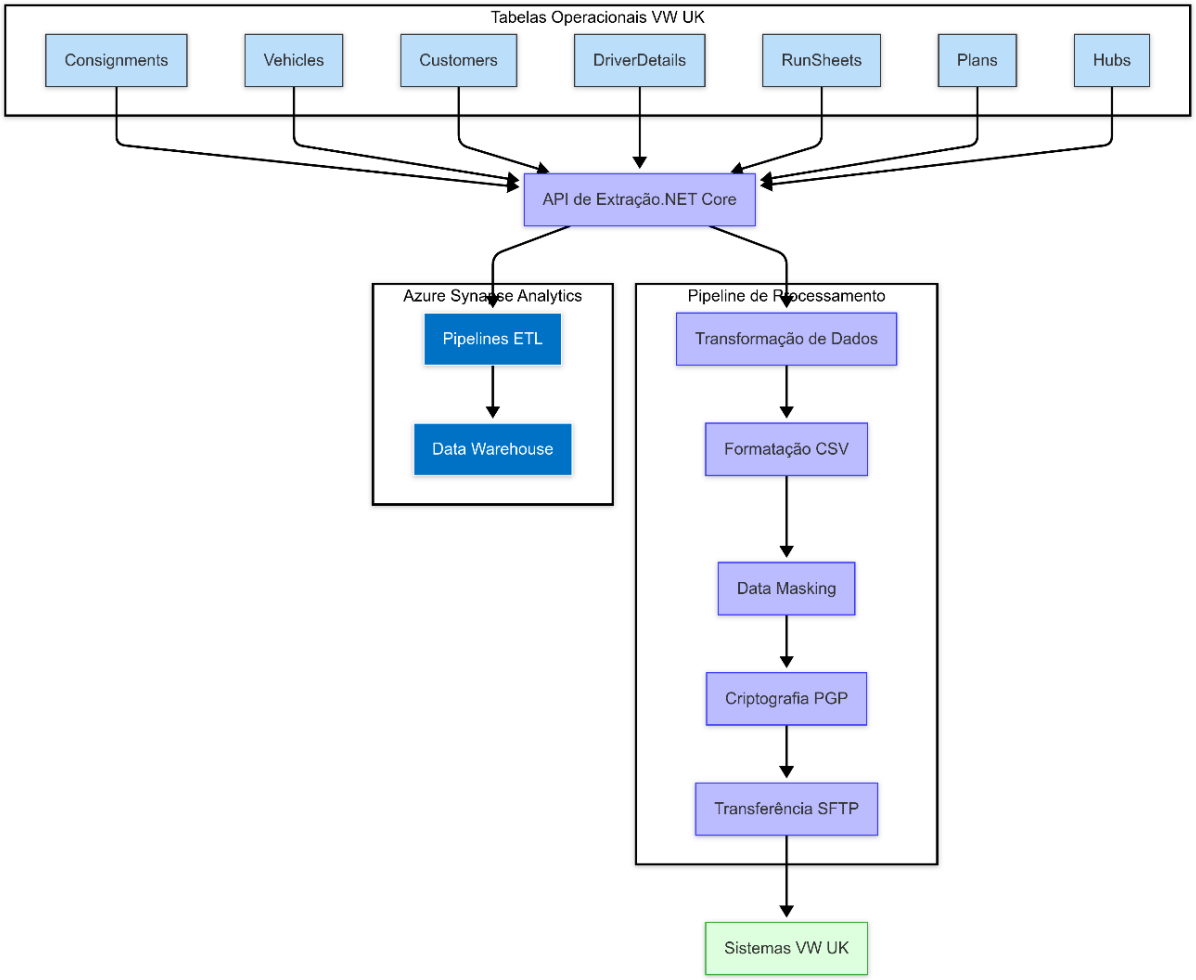

Figura 1 - Diagrama da arquitetura com as distribuições das tabelas operacionais da Volkswagen UK. Fonte: Autor (2025)

Solução Implementada

Para resolver estes problemas, foi desenvolvida uma arquitetura de integração completa, composta por:

1. ***API RESTful* para Extração de Dados**: Foi criada uma API em .NET Core responsável por extrair e processar vários conjuntos de dados. A API expõe *endpoints* específicos para cada tipo de entidade (*vehicles*, *consignments*, runs, etc.) com parâmetros para filtragem e paginação. A Figura 2 demonstra um exemplo de *endpoint* utilizado para extrair dados de *consignments*.

```
1. GET /Reports/ExecutionReport?startDate=2024-10-01&endDate=2024-10-31
```

Figura 2 - Exemplo de endpoint para extração de dados de consignments. Fonte: Código desenvolvido pelo autor

2. **Camada de Acesso a Dados Otimizada**: Foram implementados repositórios específicos utilizando Dapper para otimizar as consultas SQL e garantir alto desempenho, mesmo com grandes volumes de dados. A Figura 3 apresenta um exemplo da consulta SQL implementada para a tabela de *consignments*, incluindo *joins* com várias tabelas relacionadas e filtros temporais.

```
1. SELECT c.Id, c.ClientReference, c.ReferenceNumber, c.Status, c.ShippedOn,
2.        c.DeliveredOn, c.ShipFromName, c.ShipFromAddress, c.ShipToName,
3.        c.ShipToAddress, c.Weight, c.Volume, h.Name as HubName
4. FROM [dbo].[Consignments] c
5. INNER JOIN [dbo].[Hubs] h ON c.ShipFromHubId = h.Id
6. WHERE c.ShippedOn BETWEEN @StartDate AND @EndDate
7.   AND h.SubscriptionId = @SubscriptionId
8. ORDER BY c.ShippedOn DESC;
```

Figura 3 - Consulta SQL otimizada para extração de dados de consignments. Fonte: Código desenvolvido pelo autor

Esta consulta demonstra a otimização implementada para extrair dados de *consignments* com filtros temporais e *joins* eficientes, garantindo alto desempenho mesmo com grandes volumes de dados.

3. ***Pipeline* de Processamento Seguro**: Foi criado um fluxo automatizado que incluía:
    - Extração de dados com filtros personalizáveis (como intervalos de datas).
    - Transformação e formatação em *CSV*, conforme exigido pela Volkswagen.
    - Criptografia *PGP* para garantir a segurança durante a transferência.
    - Envio automático via *SFTP* para os servidores da VWUK.

A Figura 4 ilustra de forma detalhada o fluxograma do pipeline de processamento implementado, mostrando todas as etapas desde a extração até ao envio seguro dos dados.

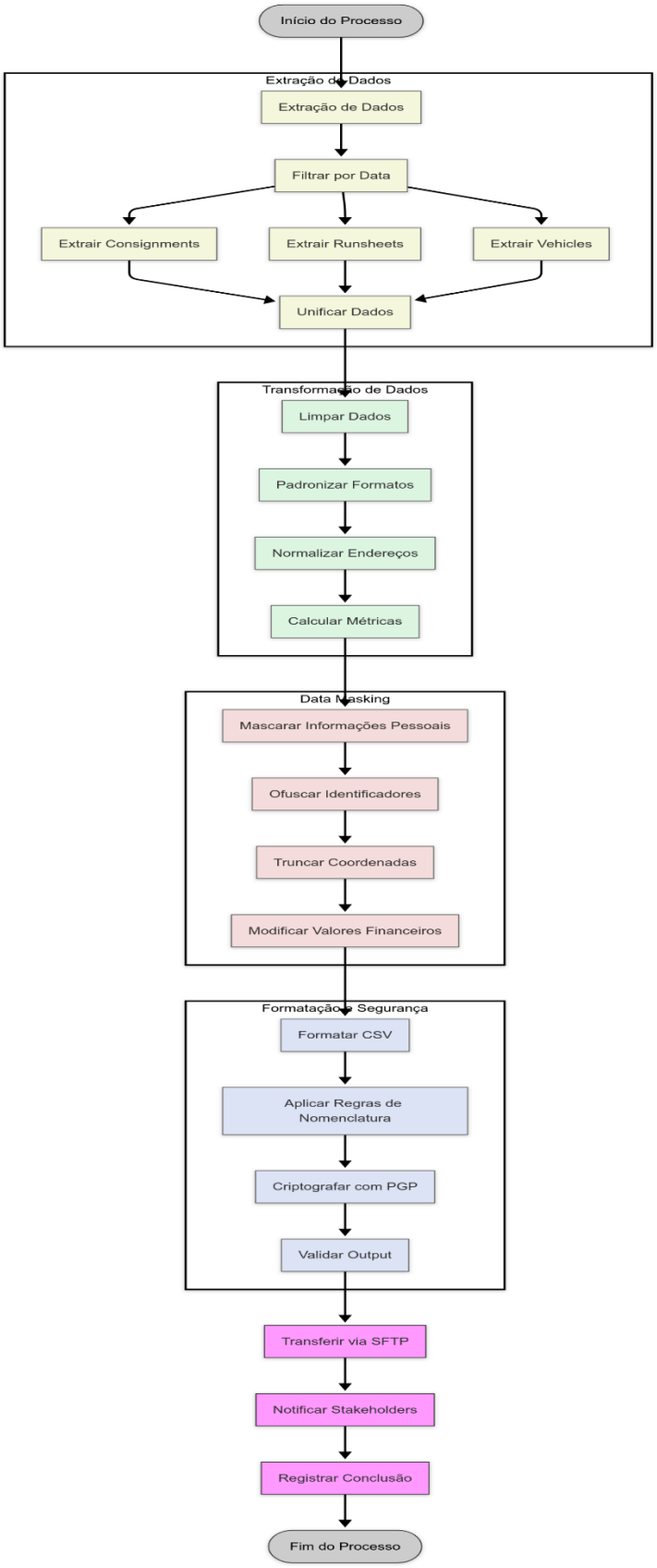

Figura 4 - Fluxograma detalhado do pipeline de processamento. Fonte: Autor (2025)

O pipeline representado na Figura 4 demonstra o fluxo completo de processamento, desde a validação inicial dos parâmetros de entrada até ao envio encriptado dos dados para os servidores da Volkswagen UK, de forma a garantir integridade e segurança em todas as etapas.

4. **Implementação de *Data Masking* para Ambientes de Teste**: Um aspeto crítico do projeto constitui na implementação de técnicas de *data masking* para permitir a utilização de dados em ambientes de desenvolvimento e teste sem comprometer informações sensíveis. Esta prática alinha-se com os requisitos da ISO 27001 para proteção de dados.

O processo de mascaramento incluiu:

- Ofuscação de dados de clientes e identificadores de veículos
- Substituição de coordenadas geográficas por aproximações
- Alteração de valores financeiros preservando a escala e distribuição estatística

A Figura 5 apresenta um exemplo concreto de transformação de dados, ilustrando como um registo original é convertido através das técnicas de mascaramento implementadas.

```
// Dados originais
{
  "driverName": "João Silva",
  "driverPhone": "+351965456547",
  "vehicleReg": "AB-51-CD",
  "clientReference": "VW-12345-XYZ",
  "deliveryAddress": "10 Downing Street, London",
  "coordinates": [51.5034, -0.1276]
}

// Dados mascarados
{
  "driverName": "D*****23",
  "driverPhone": "351***456***",
  "vehicleReg": "AB-XX-XX",
  "clientReference": "VW-99999-XXX",
  "deliveryAddress": "London Area",
  "coordinates": [51.50, -0.12]
}
```

Figura 5 - Exemplo de transformação de dados através de data masking. Fonte: Autor (2025)

O exemplo apresentado na Figura 5 demonstra como os dados sensíveis são sistematicamente ofuscados mantendo o formato e a utilidade para fins de teste, cumprindo os requisitos de segurança estabelecidos.

A Tabela 1 apresenta uma análise comparativa detalhada dos diferentes tipos de dados antes

Tipo de Dados	**Dados Originais**	**Dados Mascarados**	**Técnica Utilizada**
Nome	João Silva	J*** S***	Ofuscação parcial
Telefone	351 965 456 547	351***456***	Ofuscação parcial com preservação de prefixo
Matrícula	AB-51-CD	AB-XX-XX	Substituição por padrão
Referência do Cliente	VW-12345-XYZ	VW-99999-XXX	Substituição com preservação de formato
Endereço	10 Downing Street, London	London Area	Generalização
Coordenadas	[51.5034, -0.1276]	[51.50, -0.12]	Truncamento
Email	j.smith@example.com	j***@example.com	Ofuscação parcial com preservação de domínio
Valor de Transporte	£1,245.67	£1,2XX.XX	Ofuscação de precisão com preservação de escala
ID da Reserva	BOOK-2024-10157	BOOK-2024-XXXXX	Ofuscação parcial com preservação de prefixo/data
Código Postal	SW1A 2AA	SW** ***	Preservação de área com ofuscação

Tabela 1 - Técnicas de data masking aplicadas por tipo de dados. Fonte: Autor (2025)

e depois da aplicação das técnicas de mascaramento, especificando a técnica utilizada para cada tipo de informação

A análise da Tabela 1 revela que as técnicas de mascaramento foram selecionadas criteriosamente para cada tipo de dados, assegurando a preservação das propriedades estatísticas e funcionais necessárias para os ambientes de teste, enquanto protegem eficazmente as informações sensíveis.

5. **Padronização de Nomes e Formatos**: Implementou-se um sistema de nomenclatura padronizada, conforme as diretrizes da Volkswagen, de forma a garantir a consistência

nos ficheiros gerados, para facilitar a integração com outros sistemas. Os ficheiros CSV gerados seguem um formato específico:

- VW_[TIPO_DADO]_[DATA]_[TIMESTAMP].csv

6. **Integração com *Azure Synapse***: Foi utilizado o *Azure Synapse Analytics* para implementar *pipelines* de *ETL* mais robustos, que permitiam:
    - Extração de dados de múltiplas fontes
    - Transformar dados usando *SQL* e carregá-los eficientemente no *data warehouse*.
    - Agendar e monitorizar processos automaticamente.

Esta integração com *Azure Synapse* revelou-se particularmente valiosa por permitir combinar processamento de *big data* com *data warehousing* tradicional num único serviço.

A Figura 6 apresenta uma captura de ecrã do *Azure Synapse Analytics*, mostrando o pipeline de ETL configurado para a integração da Volkswagen UK, com os vários componentes e transformações implementadas

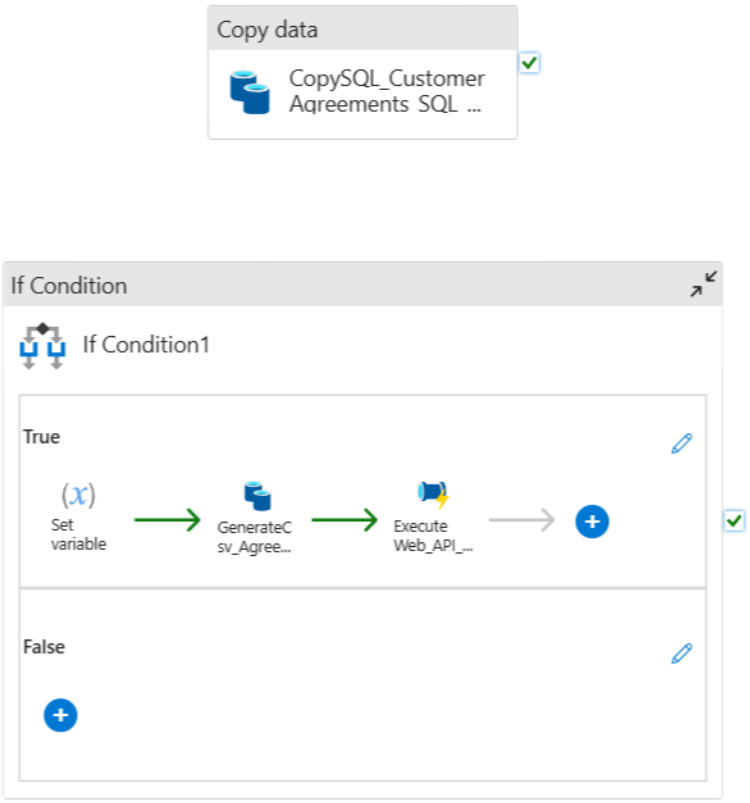

Figura 6 - Pipeline de ETL no Azure Synapse Analytics para a integração da Volkswagen UK. Fonte: Autor (2025)

O pipeline ilustrado na Figura 6 demonstra a complexidade e robustez da solução implementada, integrando múltiplas fontes de dados e aplicando transformações sofisticadas antes do carregamento no *data warehouse* centralizado.

Resultados e Impacto

A implementação desta solução, trouxe benefícios significativos para a Volkswagen UK:

- **Processos mais rápidos:** A criação de relatórios, que anteriormente demorava horas, passou a ser feita em minutos.
- **Dados mais confiáveis:** Eliminaram-se erros humanos, o que permitiu maior precisão nas análises.
- **Eliminação completa de erros humanos na transferência de dados**, devido à automação do processo
- **Conformidade total com os requisitos de segurança e privacidade de dados**, através da implementação de criptografia *PGP* e transferência segura via *SFTP*

- **Melhoria na visibilidade das operações logísticas em 16 centros de distribuição** da Volkswagen no Reino Unido

Este projeto demonstrou a importância da aplicação de princípios fundamentais da engenharia de dados e *DataOps* dentro do contexto empresarial real, tendo resultado numa solução robusta e escalável que continua a suportar as operações de uma das maiores empresas automobilísticas do mundo.

Implementação de Parâmetros de Data para Extração Seletiva

Posteriormente, a solução foi otimizada para permitir extrações seletivas com base em parâmetros temporais (datas concretas, como dia, mês e ano). Isto evitou reprocessar dados desnecessariamente, tornando o sistema ainda mais eficiente.

Este projeto foi um exemplo claro de como a engenharia de dados bem aplicada pode transformar operações empresariais, proporcionando eficiência e segurança a uma das maiores marcas automóveis do mundo.

III.2 Integração de Sistemas de Reservas

Uma das áreas às quais foi dedicada atenção durante o estágio centrou-se na migração e desenvolvimento de integrações para sistemas de reservas, com especial atenção à integração da Enfega. Fundamentalmente, esta integração permite importar reservas de transporte (os chamados *bookings*) a partir de ficheiros *Excel* enviados através de um sistema chamado AccessPoint, processando-os e inserindo-os na plataforma central da MixMove.

A Figura 7 apresenta o fluxograma completo do processo de integração da Enfega, ilustrando todas as etapas desde a receção dos ficheiros até ao processamento final dos dados.

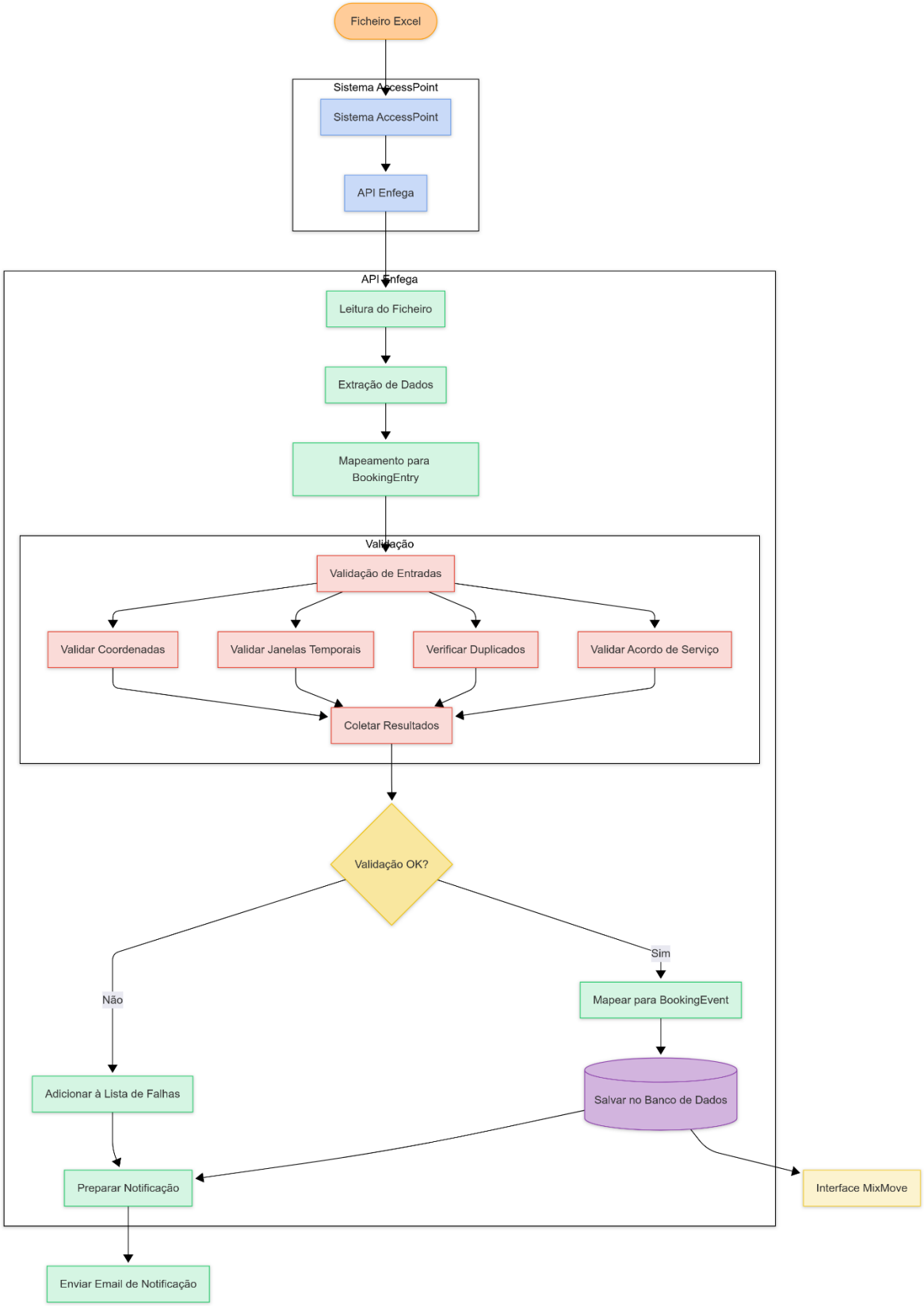

Figura 7 - Fluxograma do processo de integração Enfega. Fonte: Autor (2025)

O fluxograma apresentado na Figura 7 demonstra a complexidade do processo de integração, envolvendo múltiplas etapas de validação, transformação e processamento de dados, garantindo a integridade e qualidade da informação ao longo de todo o pipeline.

O primeiro passo consistiu em analisar o código existente numa solução legada, com o objetivo de identificar os componentes essenciais a migrar para a nova arquitetura. Tal como referem Feathers (2004) e Newman (2021), a migração de sistemas obsoletos deve ser feita passo a passo, dividindo a funcionalidade em partes mais pequenas e fáceis de gerir. Adotou-se exatamente essa abordagem, dividindo o trabalho em várias etapas:

Desenvolvimento do controlador de reservas: Foi criado um novo controlador (*BookingController*) responsável por receber os ficheiros *Excel* e extrair os dados relevantes. Para isso, foram utilizadas bibliotecas especializadas em leitura de ficheiros *Excel*, aplicando técnicas de processamento de dados estruturados.

Mapeamento de dados: Foram desenvolvidas funções que transformam os dados extraídos dos ficheiros *Excel* no modelo de dados interno do sistema. Esta parte exigiu bastante atenção, pois foi necessário normalizar e transformar os dados, seguindo os princípios que Kimball e Ross (2013) descrevem sobre processos *ETL*.

Validação de dados: Foi implementado um sistema de validação baseado no método *ValidateAsync* da integração antiga, mas foi reestruturado para ficar mais modular e extensível. Este componente verifica vários aspetos das reservas, como coordenadas geográficas, pesos e possíveis reservas duplicadas. A abordagem utilizada segue a ideia de "validação por camadas" proposta por Evans (2003) no âmbito do *Domain-Driven Design*.

Notificação de erros: Foi criado um sistema de notificação por e-mail que avisa os utilizadores quando há problemas no processamento dos ficheiros de reservas. Este sistema constrói mensagens personalizadas e integra-se com serviços externos de e-mail.

Uma parte importante do trabalho constitui em refratorar o código existente para eliminar valores *hardcoded* e tornar o sistema mais flexível e configurável. Como Martin (2019) recomenda, substituíram-se valores fixos por valores que podem ser armazenados, usando técnicas de desserialização para extrair esses valores.

Toda esta implementação, exigiu a aplicação de vários conceitos de engenharia de *software* e ciência de dados. Desde processamento de dados estruturados até validação de qualidade, transformação de dados e integração de sistemas diferentes. Esta experiência proporcionou

uma visão muito mais clara, dos desafios reais que surgem quando se trabalha com *pipelines* de dados, num ambiente empresarial.

III.3 Modernização do Sistema de Validação de Dados

O projeto de modernização do sistema de validação de dados na integração de transportes da Freja, surgiu como resposta a desafios críticos identificados na arquitetura existente. Este projeto teve como principal objetivo, substituir parte do sistema antigo, introduzindo uma nova abordagem mais estruturada e eficiente para a validação de dados. Esta necessidade emergiu para tornar as validações mais consistentes, claras e fáceis de manter, uma vez que o sistema anterior utilizava validações dispersas pelo código, sem uma estrutura coerente.

A Figura 8 apresenta uma comparação visual entre o sistema de validação antigo e o novo sistema baseado em FluentValidation, evidenciando as diferenças arquiteturais e os benefícios da modernização implementada.

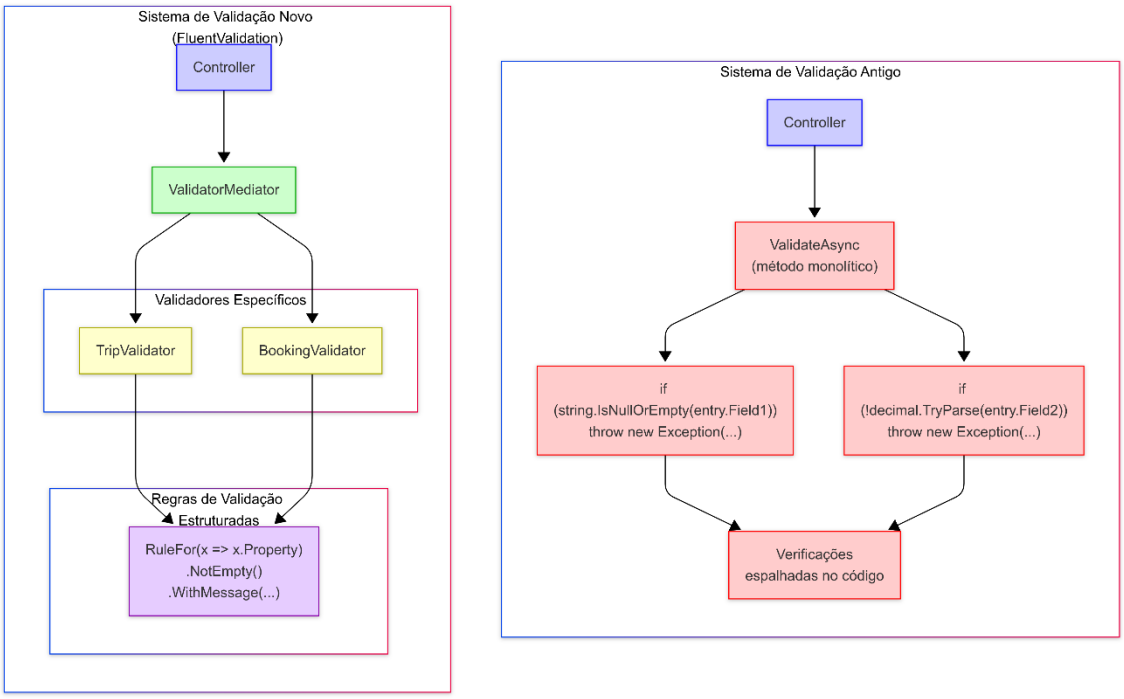

Figura 8 - Comparação entre sistema de validação antigo e novo (FluentValidation). Fonte: Autor (2025)

A análise da Figura 8 revela claramente as vantagens da nova arquitetura: enquanto o sistema antigo apresentava um método monolítico com verificações espalhadas e lógica complexa concentrada num único ponto, o novo sistema baseado em FluentValidation oferece uma estrutura modular com validadores específicos, regras bem organizadas e uma

clara separação de responsabilidades, facilitando significativamente a manutenção e extensibilidade do código.

A solução desenvolvida baseou-se na biblioteca *FluentValidation*. Esta foi selecionada após uma avaliação comparativa das alternativas disponíveis no ecossistema *.NET*. A escolha fundamentou-se na sua capacidade de implementar regras de validação, através de uma sintaxe de fácil compreensão, aliada à integração nativa com o *ASP.NET Core* e à manutenção ativa do projeto.

A Figura 9 apresenta um exemplo simplificado da estrutura implementada para validação de viagens (trips), demonstrando a utilização da biblioteca FluentValidation.

```
1. public class TripValidator : AbstractValidator<TripModel>
 2. {
 3.                 public TripValidator(IHubRepository hubRepository, IBookingRepository
bookingRepository)
 4.     {
 5.         // Validação básica de estrutura
 6.         RuleFor(trip => trip.TripId).NotEmpty().WithMessage("The TripID field is required.");
 7.         RuleFor(trip => trip.HubId).NotEmpty().WithMessage("The HubId field is required.");
 8.
 9.         // Validação de hub com acesso à base de dados
10.          RuleFor(trip => trip.HubId)
11.              .MustAsync(async (hubId, cancellation) =>
12.                  await hubRepository.ExistsAsync(hubId, cancellation))
13.              .WithMessage("hubId doesnt exists");
14.
15.         // Validação das paragens
16.                                         RuleForEach(trip     =>     trip.Stops).SetValidator(new
StopValidator(bookingRepository));
17.
18.         // Validação de veículos
19.         RuleFor(trip => trip.Vehicle).SetValidator(new VehicleValidator());
20.
```

```
21.  private  async  Task<bool>  ValidateTimeWindowsAsync(TripModel  trip,
CancellationToken cancellation)
22.  {
23.      // Lógica complexa de validação de janelas temporais
24.      // ...
25.      return true;
26.  }
27. }
```

Figura 9 - Exemplo de implementação de validação com FluentValidation. Fonte: Código desenvolvido pelo autor

O código apresentado na Figura 9 ilustra a estrutura hierárquica e modular do sistema de validação implementado, demonstrando como as diferentes regras são organizadas de forma clara e como a validação assíncrona é integrada para verificações que requerem acesso a bases de dados.

A Figura 10 apresenta um exemplo de resposta de erro da API quando a validação falha, mostrando como as mensagens são estruturadas para facilitar a identificação e correção de problemas.

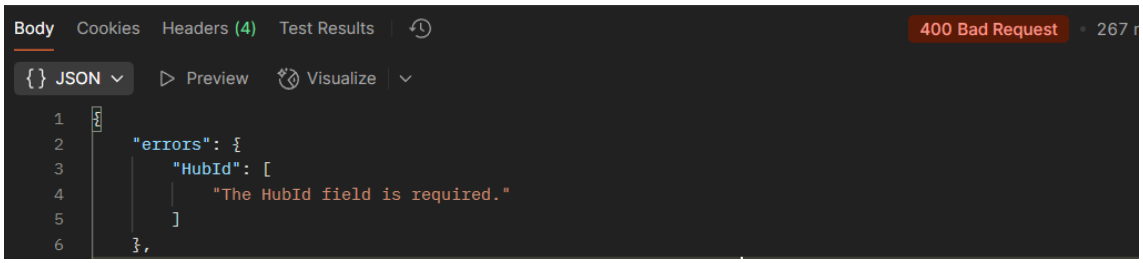

Figura 10 - Exemplo de uma resposta de erro da API quando a validação falha. Fonte: Autor (2025)

A resposta de erro ilustrada na Figura 10 demonstra como o sistema fornece feedback detalhado e estruturado aos utilizadores, facilitando a identificação rápida dos problemas e a sua resolução eficaz.

III.4 Estrutura e Funcionamento do Novo Sistema de Validação

O novo sistema de validação, foi concebido como uma hierarquia estruturada de componentes, onde cada elemento é responsável por validar um aspeto específico dos dados de transporte. Esta abordagem modular, permite distribuir a complexidade da validação por várias partes, facilitando tanto o desenvolvimento como a manutenção futura

Para a validação de viagens (trips), foram implementadas verificações que incluem:

- **Validação dos *hubs* de origem e destino**: O sistema confirma automaticamente se os *hubs* utlizados existem na base de dados e se possuem todas as informações necessárias, tal como o código do país e informações de contacto.
- **Verificação de dados de rota**: Todas as rotas são validadas para garantir que possuem um plano de viagem válido e bem estruturado.
- **Validação de *bookings***: O sistema confirma a existência das reservas associadas à viagem e avalia a sua consistência, incluindo documentação, peso, volume e outros requisitos específicos
- **Validação de folhas de rota (*runsheets*)**: Cada folha de rota é analisada detalhadamente para assegurar que todos os pontos de paragem, horários e especificações operacionais estão corretamente definidos.

Para as paragens (stops) de cada rota, foram implementadas validações específicas para:

- **Coordenadas geográficas**: O sistema verifica se as coordenadas de latitude e longitude fornecidas são valores decimais válidos, essenciais para o correto funcionamento dos sistemas de navegação e rastreamento.
- **Códigos de scanner**: Todos os códigos de scanner usados nas operações de entrada e saída, são verificados na base de dados para garantir a sua existência e validade.
- **Datas e horários**: As datas de chegada prevista (PTA) e partida prevista (PTD) são validadas quanto ao formato.
- **Identificação de contentores (SSCCs)**: O sistema verifica automaticamente, se os contentores reservados a uma paragem, correspondem aos registados nas reservas, evitando discrepâncias operacionais.

Para veículos e equipamentos de transporte, foram implementadas verificações de:

- **Matrículas**: Verificação da presença e formato correto das matrículas dos veículos.
- **Provedores de serviço**: Validação da existência dos provedores de serviço no sistema, com associação correta com os veículos designados.
- **Equipamentos específicos**: Validação de reboques e outros equipamentos quanto à sua disponibilidade e adequação para o tipo de transporte.

Integração com a Interface de Programação de Aplicações (API)

Todo este sistema de validação foi integrado com a API, que recebe os dados de transporte. A integração foi realizada, utilizando o sistema de injeção de dependências disponível no ASP.NET Core, garantindo que os componentes de validação sejam instanciados.

Quando a API recebe uma solicitação para criar ou atualizar dados de transporte, o sistema de validação é acionado antes de qualquer processamento de dados. Se alguma regra de validação falhar, a API responde imediatamente com um código de erro HTTP 400 (*Bad Request*), acompanhado de uma lista detalhada de todos os problemas encontrados, de forma a facilitar a sua identificação e correção pelos clientes da API.

A Figura 11 apresenta um exemplo específico de validação implementada para verificação de coordenadas geográficas para paragens.

```
1. public class StopValidator : AbstractValidator<StopModel>
2. {
3.     public StopValidator(IBookingRepository bookingRepository)
4.     {
5.         // Validação de coordenadas geográficas
6.         RuleFor(stop => stop.Latitude)
7.             .NotEmpty().WithMessage("A latitude é obrigatória")
8.             .Must(BeValidLatitude).WithMessage("A latitude deve ser um valor decimal");
9.
10.         RuleFor(stop => stop.Longitude)
11.             .NotEmpty().WithMessage("A longitude é obrigatória")
12.             .Must(BeValidLongitude).WithMessage("A longitude deve ser um valor decimal");
13.
14.         // Outras validações...
15.     }
16. }
```

Figura 11 - Exemplo de validação de coordenadas geográficas. Fonte: Código desenvolvido pelo autor

O exemplo de código na Figura 11 demonstra como as validações específicas são implementadas de forma clara e reutilizável, com mensagens de erro personalizadas que facilitam a identificação e correção de problemas pelos utilizadores da API.

Impacto e Benefícios do Novo Sistema de Validação

A transição para o **FluentValidation,** representou uma evolução significativa na forma como as regras de validação são estruturadas e mantidas no sistema. Embora todas as validações necessárias já existissem na versão anterior (implementadas através de verificações condicionais), a nova abordagem trouxe vantagens em termos de organização e eficiência.

1. **Validações mais legíveis e bem organizadas**

A estrutura atual fez com que as regras deixassem de estar dispersas em várias verificações condicionais para passarem a estar agrupadas de forma lógica, consoante a entidade a que se referem (viagens, paragens, reservas, etc.). Esta organização facilita a leitura e compreensão das regras por parte de toda a equipa de desenvolvimento.

2. **Mensagens de erro mais claras e consistentes**

O sistema agora apresenta mensagens de erro detalhadas e uniformes, indicando sempre o campo em causa, o problema detetado e, quando aplicável, o valor recebido. Por exemplo, em vez de uma mensagem genérica como "Dados inválidos", o utilizador recebe informação específica como "A latitude da paragem 3 (ID: XYZ123) deve ser um valor decimal válido. Valor recebido: 'ABC'".

3. **Capacidade de manutenção**

A utilização de classes dedicadas para a validação, tornou muito mais simples a atualização ou extensão das validações existentes. Qualquer ajuste necessário, pode ser feito num único local, reduzindo o risco de inconsistências e facilitando a implementação de novas funcionalidades.

4. **Validações contextuais mais robustas**

A nova estrutura, permite implementar de forma mais clara e eficiente as validações que dependem do contexto, ou de relações entre diferentes entidades. Entre os exemplos destaca-se, a verificação de que todos os contentores associados a uma paragem, estão corretamente registados nas reservas correspondentes.

5. **Integração eficiente com operações assíncronas**

As validações que requerem consultas a bases de dados ou serviços externos beneficiam de uma implementação mais robusta e de melhor desempenho.

6. **Facilidade de teste e validação**

Cada conjunto de regras está agora isolado na sua própria classe, o que simplifica a criação e execução de testes unitários.

Este trabalho de modernização resultou numa melhoria na qualidade dos dados processados pelo sistema, reduzindo o número de incidentes operacionais e proporcionando uma experiência significativamente superior para os utilizadores da plataforma de transporte.

Figura 12 apresenta o diagrama de sequência detalhado que ilustra a comunicação complexa entre o sistema MixMove, a API Rangel e o serviço externo Routyn, demonstrando todos os passos do processo de validação, processamento e integração de dados

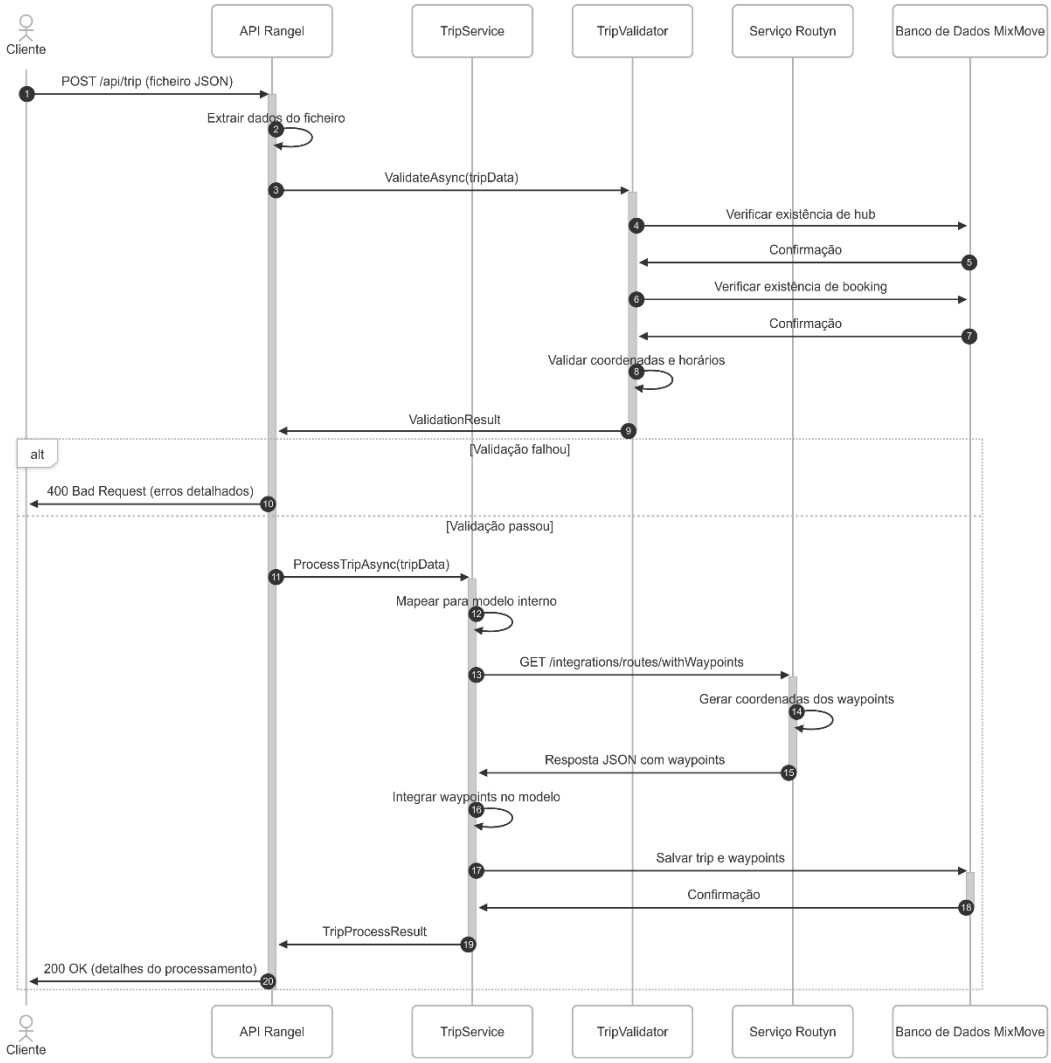

Figura 12 - Diagrama de sequência da comunicação entre o sistema MixMove, a API Rangel e o serviço externo Routyn. Fonte: Autor (2025)

A análise da Figura 12 revela a sofisticação do pipeline implementado, mostrando como o sistema processa de forma assíncrona os dados de entrada, executa validações em múltiplas

camadas, integra-se com o serviço externo Routyn para enriquecimento de dados geográficos, e finaliza com o armazenamento seguro no base de dados, garantindo a integridade e rastreabilidade de todo o processo.

III.5 Automação de Pipelines de Dados

Um dos grandes focos do estágio, centrou-se na criação de processos automatizados para gerir fluxos de dados de forma eficiente. A ideia consistia em desenvolver pipelines que funcionassem com o mínimo de intervenção possível, que processassem automaticamente os dados recebidos, procedendo à validação e ao encaminhamento para os sistemas adequados.

Um dos casos mais interessantes envolveu o desenvolvimento de um pipeline para integrar rotas de transporte (*trips*) para a Rangel. O sistema recebe ficheiros com informações sobre rotas, processa-os e integra-os com um serviço externo (o Routyn) para obter coordenadas detalhadas dos *waypoints*. Como defendem Warren e Marz (2015), os pipelines devem ser capazes de lidar com falhas sem comprometer a qualidade dos dados.

Seguindo esta linha, as soluções implementadas incluem:

1. **Processamento assíncrono** – A utilização de métodos assíncronos em C# para evitar bloqueios no servidor API, permitindo que vários pedidos fossem tratados em simultâneo.
2. **Validação por etapas** – A criação de um processo de validação faseado, que verifica desde a estrutura do ficheiro, até à consistência dos dados. Cada etapa regista eventuais problemas, que depois são compilados numa resposta enviada para o cliente.
3. **Gestão de erros robusta** – Foi implementado um mecanismo de gestão de erros que captura exceções em diferentes fases do pipeline, regista-as para análise posterior e quando necessário, alerta os utilizadores. Kleppmann (2017) sublinha a importância disto em sistemas distribuídos, onde as falhas são inevitáveis.
4. **Integração com serviços externos** – A integração com o Routyn para obter os dados das rotas com coordenadas GPS, utilizando autenticação por *tokens*, de forma a garantir que as respostas eram processadas corretamente.

Esta experiência, permitiu aprofundar conhecimentos em áreas essenciais para a ciência de dados. Como processamento assíncrono, integração de APIs externas, validação de dados e

gestão de erros em sistemas distribuídos. Como referem Kleppmann (2017), estas competências são fundamentais para implementar projetos de dados com sucesso.

III.6 Desenvolvimento de APIs e Serviços Web

Uma parte substancial do trabalho durante o estágio caracterizou-se pelo desenvolvimento e manutenção de APIs e serviços web que facilitam a comunicação entre diferentes sistemas. Como referem Richardson e Ruby (2007), as APIs constituem a espinha dorsal da interoperabilidade nas arquiteturas modernas, permitindo a integração de sistemas heterogéneos através de interfaces bem definidas e uniformizadas.

Uma parte substancial do trabalho centrou-se no desenvolvimento e na manutenção de APIs (Application Programming Interfaces), que desempenham um papel crítico na interoperabilidade entre sistemas distintos. Como sustentam Richardson e Ruby (2007), as APIs representam um pilar fundamental nas arquiteturas de software contemporâneas, uma vez que estabelecem interfaces padronizadas e bem definidas, permitindo a integração eficiente de sistemas heterogéneos.

Na implementação das APIs, foram seguidas as melhores práticas para garantir robustez, segurança e usabilidade.

A Figura 12 apresenta um exemplo de estrutura de um controlador desenvolvido, demonstrando a aplicação de padrões de design e boas práticas de desenvolvimento

```
1. [Route("api/[controller]")]
2. [ApiController]
3. [Authorize]
4. public class BookingController : ControllerBase
5. {
6.     private readonly IBookingService _bookingService;
7.     private readonly ILogger<BookingController> _logger;
8.
9.                 public BookingController(IBookingService bookingService,
ILogger<BookingController> logger)
10.    {
11.        _bookingService = bookingService;
12.        _logger = logger;
```

```
13.     }
14.
15.     [HttpPost]
16.     [ProducesResponseType(typeof(BookingResponse), StatusCodes.Status200OK)]
17.                                     [ProducesResponseType(typeof(ValidationProblemDetails),
StatusCodes.Status400BadRequest)]
18.     [ProducesResponseType(StatusCodes.Status401Unauthorized)]
19.     public async Task<IActionResult> CreateBooking([FromForm] IFormFile file)
20.     {
21.         try
22.         {
23.             // Processamento do ficheiro e criação das reservas
24.             var result = await _bookingService.ProcessBookingFile(file);
25.
26.             return Ok(new BookingResponse
27.             {
28.                 SuccessCount = result.SuccessfulBookings.Count,
29.                 FailedCount = result.FailedBookings.Count,
30.                 Failures = result.FailedBookings.Select(f => new BookingFailure
31.                 {
32.                     Reference = f.Reference,
33.                     Reason = f.FailureReason
34.                 }).ToList()
35.             });
36.         }
37.         catch (Exception ex)
38.         {
39.             _logger.LogError(ex, "Erro ao processar ficheiro de reservas");
40.             return StatusCode(500, "Ocorreu um erro ao processar o ficheiro");
41.         }
42.     }
43.
44.     // Outros endpoints...
45. }
```

Figura 13 - Exemplo de estrutura de controlador de API. Fonte: Código desenvolvido pelo autor

O exemplo de código apresentado na Figura 12 ilustra a implementação de um controlador seguindo as convenções do ASP.NET Core, com injeção de dependências, tratamento de erros estruturado e documentação automática através de atributos que facilitam a geração de documentação *OpenAPI*.

Na implementação destas APIs, foram aplicados diversos princípios e padrões de arquitetura de software, com o objetivo de assegurar robustez, escalabilidade e capacidade de manutenção. Entre as principais técnicas aplicadas, destacam-se:

- **Injeção de dependências**: A adoção deste padrão de design, para promover a modularidade e facilitar os testes ao código. Ao definir serviços sob a forma de interfaces, garantiu-se que as dependências fossem injetadas de forma explícita, o que permitiu a substituição simplificada de componentes em ambientes de teste ou em cenários de evolução do sistema.
- **Padrão *Repository***: Este padrão permite separar a lógica de negócio da parte que lida com os dados. Ao criar uma camada no meio, conseguiu-se deixar o sistema mais flexível, permitindo mudar a forma como se guardam as informações sem ser necessário alterar nas regras de negócio que realmente importam para a empresa.
- ***Middleware* de autenticação e autorização**: A implementação de uma camada de segurança intermediária nas APIs que assegura a validação de credenciais dos utilizadores. Ou seja, primeiro existe uma autenticação do *token* e posteriormente, realiza-se o controlo do acesso com base nos recursos solicitados.
- **Documentação com Swagger**: A documentação das APIs utiliza o Swagger/OpenAPI, facilitando a compreensão e utilização das interfaces por parte dos clientes. Este trabalho proporcionou uma noção aprofundada dos princípios e desafios associados ao desenvolvimento de APIs em ambientes empresariais, bem como da importância de que estas sejam bem desenhadas no contexto de arquiteturas orientadas a dados. Como destacam Abran e Suryn (2003), as APIs modernas não são apenas interfaces técnicas, mas constituem produtos que devem ser desenhados com foco na experiência do programador e nas necessidades de negócio.

O trabalho desenvolvido reforçou a perceção sobre a importância de APIs bem arquitetadas em sistemas distribuídos, evidenciando como decisões de design impactam a eficiência, a segurança e a adoção da solução. A aplicação de padrões consolidados, aliada a uma abordagem centrada no utilizador, demonstrou ser decisiva para o sucesso de integrações em ambientes empresariais.

III.7 Testes e Validação

A implementação de estratégias robustas de testes de software constituiu uma componente essencial do trabalho durante o estágio. Tal como defendem autores como Myers, Sandler e Badgett. (2015), testar bem é crucial para garantir que sistemas que trabalham com dados sejam fiáveis, seguros e rápidos, especialmente quando qualquer falha pode afetar seriamente o funcionamento da empresa.

Os projetos desenvolvidos incluíram testes organizados de forma a garantir que o software cumpria o que estava previsto. A estratégia incluiu vários tipos de testes, cada um com a sua própria finalidade:

1. **Testes unitários** – O desenvolvimento incluiu conjuntos de testes para várias APIs, utilizando o *framework* xUnit no .NET Core. Estes testes focaram-se em unidades isoladas de código, tais como validadores de dados, funções de mapeamento e lógica de negócio. Seguindo as boas práticas recomendadas por Martin (2002), a utilização de *mocks* permitiu isolar os componentes testados das suas dependências externas, assegurando que cada teste verificava apenas o comportamento pretendido.
2. **Testes de integração** – Os testes de integração validaram a interação entre diferentes componentes, especialmente em cenários em que vários serviços trabalhavam em conjunto. Um exemplo concreto foi a integração entre a Rangel Trip API e o serviço externo Routyn, onde se garantiu que as chamadas eram processadas corretamente e que os dados recebidos (como *waypoints*) eram integrados de forma adequada no modelo de dados da MixMove.
3. **Testes funcionais e de aceitação** – Os processos criados validavam o *end-to-end* dos fluxos de integração, confirmando que os dados inseridos numa ponta do sistema eram devidamente processados e refletidos na outra. Estes testes seguiram a abordagem de "testes baseados em cenários", proposta por Myers et al. (2015), de forma a simular situações reais de utilização para garantir que o sistema respondia conforme as expectativas.
4. **Validadores de dados** – Os componentes desenvolvidos analisavam os dados em profundidade, verificando a sua integridade, conformidade e formatação adequada (como no caso das coordenadas geográficas). Estes validadores foram construídos como módulos reutilizáveis, seguindo o padrão "*Specification*" descrito por Evans (2003), o que permitia combinar regras simples para formar verificações mais complexas.

Um dos exemplos mais relevantes foi a validação na integração com a Enfega, onde foram implementadas verificações para:

- Coordenadas geográficas – Assegurar que estavam no formato correto e dentro de intervalos válidos;
- Peso e dimensões – Confirmar que os valores eram coerentes e não continham erros;
- Janelas temporais de entregas – Validar que os horários definidos faziam sentido;
- Reservas duplicadas ou conflituosas – Detetar possíveis sobreposições ou registos repetidos.

Como referem Nguyen, Nguyen e Nguyen-Hoang (2025), a validação de dados em integrações deve ser rigorosa, uma vez que os dados podem vir de sistemas externos com diferentes níveis de qualidade. Por isso, estes validadores incluem múltiplas camadas de verificação e mecanismos de notificação, para que qualquer problema fosse identificado e comunicado de forma eficaz.

Esta experiência mostrou como os testes e a validação são verdadeiramente importantes, sobretudo em empresas onde dados de má qualidade podem prejudicar todo o negócio. Como bem referem Nguyen et al. (2025), não se pode abdicar de uma estratégia completa e organizada de testes se se pretende ter sistemas em que se possa confiar.

III.8 Documentação e Formação

A documentação adequada de sistemas e APIs constitui um aspeto frequentemente subestimado, mas crucial para a sustentabilidade a longo prazo de projetos de engenharia de dados. Como destacam Spinellis (2003), a documentação não é apenas uma referência técnica, mas um meio de comunicação que facilita a compreensão, manutenção e evolução dos sistemas.

Apresentam-se algumas das iniciativas que foram implementadas:

1. Documentação de APIs com Swagger/OpenAPI

Para todas as APIs que foram desenvolvidas, a documentação baseia-se no Swagger/OpenAPI para criar documentação clara e acessível, incluindo:

- Descrição detalhada de *endpoints* (métodos, parâmetros, formatos esperados);

- Exemplos de *requests* e responses, para facilitar a integração por parte de outras equipas;
- Códigos de estado HTTP e os seus significados (erros comuns, como validações falhadas);
- Requisitos de autenticação e autorização, facilitando o acesso seguro aos recursos, através da capacitação dos utilizadores.

2. Documentação de Integrações

A integração com a Enfega exigiu uma documentação detalhada, que foi disponibilizada no Confluence da empresa. Esta incluía:

- Visão geral da arquitetura, explicando como os sistemas comunicavam entre si;
- Fluxos de dados, desde a entrada até ao processamento final;
- Formatos de ficheiros suportados e restrições (ex: campos obrigatórios, limites de tamanho);
- Processo de validação e tratamento de erros, para ajudar a equipa a diagnosticar problemas rapidamente;
- Configurações específicas por cliente, evitando ambiguidades na implementação.

A criação desta documentação seguiu princípios de "documentação centrada no utilizador", como proposto por Maalej e Robillard (2019), focando-se nas necessidades específicas dos diferentes públicos-alvo: programadores que desenvolvem soluções usando as APIs, administradores de sistema responsáveis pela configuração e manutenção, e utilizadores finais que interagem com os sistemas através de interfaces de aplicação.

Esta experiência possibilitou uma melhor compreensão sobre o papel fundamental da documentação quando se trabalha com sistemas de dados, especialmente em contextos onde várias equipas e organizações colaboram entre si. Como destacam Maalej e Robillard (2019), uma documentação bem estruturada reduz significativamente os custos de manutenção e evolução a longo prazo, além de facilitar a integração de novos elementos nas equipas de desenvolvimento.

IV- Análise Crítica e Reflexão

IV.1 Impacto do Trabalho Desenvolvido

As atividades realizadas durante o estágio tiveram um impacto significativo, tanto a nível organizacional como no desenvolvimento profissional e académico. Ao analisar criticamente esses efeitos, permitiu identificar várias dimensões relevantes.

No contexto organizacional, o trabalho desenvolvido contribuiu para:

Modernização da Arquitetura de Integrações

A migração de integrações legadas para uma arquitetura baseada em microserviços e APIs independentes aumentou significativamente a modularidade, sustentabilidade e escalabilidade do sistema. Como destacam Lewis e Fowler (2014), arquiteturas baseadas em microserviços facilitam a evolução contínua dos sistemas, permitindo que diferentes componentes evoluam a ritmos distintos.

O impacto desta modernização pode ser quantificado através de indicadores concretos. Antes da implementação das novas soluções, as integrações existentes apresentavam um tempo médio de desenvolvimento de novas funcionalidades de aproximadamente 3-4 semanas, devido à necessidade de alterar múltiplos componentes interligados. Após a modularização, este tempo foi reduzido para 1-2 semanas, representando uma melhoria de eficiência de aproximadamente 50%. Adicionalmente, a taxa de defeitos introduzidos durante alterações diminuiu consideravelmente, uma vez que as modificações ficaram mais localizadas e com menor impacto nos componentes adjacentes.

A nova arquitetura permitiu também uma maior facilidade na manutenção. Enquanto anteriormente uma alteração numa integração podia exigir testes em todo o sistema, a abordagem modular permitiu isolar os testes apenas nos componentes direitamente afetados. Isto traduziu-se numa redução do tempo necessário para ciclos de teste de cerca de 60%, passando de uma média de 2 dias para aproximadamente 8 horas.

A escalabilidade horizontal tornou-se uma realidade, com a possibilidade de escalar componentes específicos conforme a necessidade, em vez de ter de escalar todo o sistema. Este benefício foi particularmente evidente na integração com a Volkswagen UK, onde os picos de processamento de dados podiam ser acomodados através do escalamento seletivo dos serviços de processamento ETL, sem impactar outros componentes do sistema.

Melhoria da Qualidade de Dados

A implementação de validadores nas diversas integrações permitiu identificar e corrigir problemas de qualidade, reduzindo drasticamente a ocorrência de erros em fases posteriores ao processamento. Como referem Batini e Scannapieco (2016), é fundamental garantir a qualidade dos dados o mais próximo possível da sua origem.

Os resultados desta melhoria foram particularmente evidentes na integração da Enfega, onde a taxa de erros de validação diminuiu de aproximadamente 15% para menos de 2% após a implementação do novo sistema baseado em FluentValidation. Esta redução traduziu-se numa diminuição significativa do trabalho manual necessário para correção de dados, libertando recursos humanos para atividades de maior valor acrescentado.

Antes da implementação das novas validações, cerca de 3-4 horas semanais eram dedicadas à identificação e correção manual de inconsistências nos dados da Enfega. Após a implementação, este tempo foi reduzido para aproximadamente 30 minutos semanais, representando uma melhoria de eficiência de cerca de 85%. Os dados agora chegam ao sistema principal com um nível de qualidade consistentemente elevado, eliminando a necessidade de reprocessamento e os consequentes *delays* nas operações.

A implementação de validações em múltiplas camadas também permitiu a identificação proactiva de padrões de erro, facilitando a comunicação com os clientes sobre problemas sistemáticos nos seus processos de geração de dados. Na integração da Freja, por exemplo, esta abordagem permitiu identificar uma inconsistência sistemática nos códigos SSCC que estava a causar problemas *downstream*, resultando numa correção do processo na origem e numa melhoria global da qualidade dos dados.

Automação de Processos Manuais

A criação de pipelines automatizados para processar ficheiros e integrar com sistemas externos diminuiu drasticamente a necessidade de trabalho manual, melhorando a eficiência das operações e reduzindo erros. De acordo com Kleppmann (2017), a automação é essencial na engenharia de dados atual, dando a possibilidade de lidar com volumes cada vez maiores de informação de forma eficaz.

O impacto mais significativo foi observado no projeto da Volkswagen UK, onde o processo anteriormente manual de geração de relatórios foi completamente automatizado. Antes da implementação da solução, a geração de um relatório mensal exigia aproximadamente 6-8 horas de trabalho manual, incluindo extração de dados de múltiplas fontes, formatação,

validação e envio. Com a nova solução automatizada, este processo foi reduzido para aproximadamente 15 minutos de processamento automático, representando uma melhoria de eficiência superior a 95%.

A automatização eliminou também a possibilidade de erros humanos na transferência e formatação de dados. Antes da implementação, verificavam-se, em média 2-3 discrepâncias por relatório que exigiam correção manual e comunicação com o cliente. Após a automatização, estas discrepâncias foram completamente eliminadas, melhorando significativamente a confiança do cliente na qualidade e precisão dos dados fornecidos.

Na integração da Rangel, a automatização do processamento de rotas permitiu reduzir o tempo de processamento de ficheiros de dados de aproximadamente 2 horas para 10 minutos, incluindo a integração automática com a API do Routyn para obtenção de coordenadas GPS. Esta melhoria não só aumentou a eficiência operacional, como também permitiu disponibilizar informações atualizadas aos utilizadores finais com uma latência significativamente menor.

Melhoria da Monitorização e Observabilidade

A implementação de mecanismos de *logging* e alerta mais detalhados facilitou a deteção e resolução de problemas, além de proporcionar insights valiosos sobre o funcionamento dos sistemas. Como destacam Beyer et al. (2021), em sistemas complexos e distribuídos é fundamental poder observar o que acontece, compreendendo o comportamento do sistema através dos resultados que produz.

Antes da implementação dos novos sistemas de monitorização, a identificação de problemas nas integrações era frequentemente reativa, dependendo de relatórios de utilizadores ou da descoberta casual durante verificações de rotina. O tempo médio para deteção de problemas era de aproximadamente 4-6 horas, e o tempo total para resolução podia estender-se até 24-48 horas, dependendo da complexidade do problema.

Com a implementação de *logging* estruturado e alertas automáticos, o tempo médio para deteção de problemas foi reduzido para menos de 15 minutos, e o tempo total para resolução diminuiu para 2-4 horas na maioria dos casos. Esta melhoria foi possível através da implementação de *dashboards* em tempo real que mostram o estado de todas as integrações, métricas de desempenho, e alertas automáticos quando parâmetros saem dos valores normais.

A observabilidade melhorada permitiu também otimizações proactivas. Na integração da Volkswagen UK, a análise dos padrões de utilização revelou picos de processamento em

horários específicos, o que permitiu ajustar a programação de tarefas para distribuir melhor a carga e evitar congestionamentos. Esta otimização resultou numa melhoria de 30% no tempo médio de processamento durante períodos de pico.

Os *logs* detalhados facilitaram significativamente o processo de *debugging* e manutenção. Enquanto anteriormente a resolução de um problema complexo podia exigir várias horas de investigação, os *logs* estruturados e contextualizados permitem agora identificar rapidamente a origem de problemas, reduzindo o tempo médio de investigação de 3-4 horas para aproximadamente 30-60 minutos.

Impacto Quantitativo Global

O conjunto de melhorias implementadas durante o estágio traduziu-se em benefícios mensuráveis significativos:

Eficiência Operacional: Redução média de 70-85% no tempo necessário para processos que anteriormente eram manuais ou semi-automatizados.

Qualidade de Dados: Diminuição da taxa de erros de aproximadamente 15% para menos de 2% nas integrações modernizadas.

Tempo de Resposta: Melhoria de 85% no tempo de processamento de relatórios da Volkswagen UK (de 6-8 horas para 15 minutos).

Deteção de Problemas: Redução do tempo médio de deteção de falhas de 4-6 horas para menos de 15 minutos.

Capacidade de manutenção: Redução de 50% no tempo necessário para desenvolver novas funcionalidades devido à arquitetura modular.

Escalabilidade: Capacidade de escalar componentes individuais conforme necessário, o que eliminou a necessidade de escalar todo o sistema.

Estes resultados demonstram que o trabalho desenvolvido não só atingiu os objetivos propostos, como também criou valor mensurável e sustentável para a organização, estabelecendo uma base sólida para futuras evoluções e melhorias.

IV.2 Competências Adquiridas

O estágio permitiu desenvolver e consolidar diversas competências essenciais para a formação como cientista de dados, que se podem dividir em três categorias: técnicas, metodológicas e interpessoais.

No domínio das competências técnicas, destacam-se:

1. **Desenvolvimento de APIs e integração de sistemas**: A experiência proporcionou a possibilidade de aprender a criar e implementar APIs utilizando tecnologias como ASP.NET Core, Entity Framework e protocolos de comunicação como REST. Como referem Newman (2021), as APIs constituem um componente fundamental nas arquiteturas modernas orientadas a dados, facilitando a comunicação entre sistemas distintos.
2. **Engenharia de dados**: Foi adquirida experiência na criação e manutenção de pipelines de dados, incluindo extração de fontes diversas, transformação e carregamento em sistemas de destino. Estas capacidades alinharam-se com o que Kassen (2022) identifica como o núcleo da engenharia de dados: a construção de infraestruturas que permitem que os dados fluam de forma eficiente e confiável entre diferentes sistemas.
3. **Processamento de dados estruturados**: Foram aprofundados os conhecimentos práticos com formatos como XLXS, CSV e JSON, incluindo análise, validação e transformação de dados. Como destacam Schutt e O'Neil (2013), a capacidade de trabalhar eficientemente com diferentes formatos de dados é uma competência fundamental na ciência de dados, muitas das vezes ocupando grande parte do tempo nos projetos.
4. **Bases de dados e SQL**: A consolidação das competências em modelação de dados, *queries* SQL e interação com sistemas de gestão de bases de dados, especialmente o SQL Server. Coronel e Morris (2014), reforçam que o domínio de SQL e dos conceitos de bases de dados continua a ser indispensável, mesmo com o surgimento de novas tecnologias.
5. **DataOps e automatização**: A aquisição de experiência com práticas de DataOps aplicadas a sistemas de dados, incluindo a gestão de *pipelines*, monitorização de fluxos de dados e integração contínua. Estas práticas, como referem *Kim et al. (2021),* são essenciais para garantir a fiabilidade e evolução contínua de sistemas em produção.

Relativamente a competências metodológicas, realçam-se:

1. **Análise e modelação de requisitos**: O desenvolvimento da capacidade para analisar necessidades de negócio e transformá-las em requisitos técnicos que podem ser implementados. Como referem Wiegers e Beatty (2013), esta competência é essencial para fazer a ponte entre o domínio do problema e o da solução.
2. **Aplicação prática de conhecimentos teóricos**: A aplicação prática de conceitos no dia a dia do estágio de conceitos que foram estudados durante o mestrado, como pipelines de dados, validação, integração de sistemas e arquiteturas distribuídas. Esta experiência reflete o que Kolb (1984) chama de "aprendizagem experiencial", que acontece quando conseguimos unir a teoria com a prática.
3. **Design de arquitetura de software**: A aquisição de experiência na conceção de arquiteturas de software modulares e extensíveis, usando padrões e princípios como SOLID, DDD e arquitetura limpa. Vernon (2016) sublinha a relevância destes conceitos para desenvolver sistemas robustos e sustentáveis, sobretudo em áreas de negócio mais complexas.
4. **Testes e garantia de qualidade**: Foram consolidados conhecimentos em testes unitários e de integração, bem como em validação de dados em diferentes camadas. Segundo Humble e Farley (2010), estas práticas são fundamentais para assegurar a fiabilidade e precisão dos sistemas de software.
5. **Documentação técnica**: O desenvolvimento da capacidade de produzir documentação clara e útil para diferentes públicos, desde programadores a administradores de sistemas e utilizadores finais. Esta competência está alinhada com o que Spinellis (2003) considera uma responsabilidade crucial em equipas técnicas.

No âmbito das competências interpessoais, destacam-se:

1. **Trabalho em equipa**: O aperfeiçoamento da capacidade de colaborar de forma eficaz em equipas de diferentes áreas, onde foram valorizadas diferentes perspetivas e a contribuir para objetivos comuns. Como referem Garcia et al. (2013), o trabalho em equipa é especialmente relevante em projetos de dados, que muitas vezes exigem a colaboração entre diversos perfis.
2. **Comunicação técnica**: O desenvolvimento da capacidade de explicar conceitos técnicos a públicos com diferentes níveis de conhecimento, adaptando a linguagem e o detalhe conforme necessário. Davenport e Patil (2012) sublinha a importância desta competência para profissionais de dados, que muitas vezes funcionam como intermediários entre áreas técnicas e de negócio.

3. **Gestão de tempo e prioridades**: A aquisição de experiência na organização de múltiplas tarefas e projetos, sendo que foi necessário equilibrar prazos e prioridades. Newport (2016) destaca esta habilidade, sendo esta essencial num ambiente profissional marcado por constantes solicitações e pela necessidade de focar no que realmente importa.
4. **Resolução de problemas**: O desenvolvimento de uma abordagem mais estruturada para identificar, analisar e resolver problemas técnicos complexos. Davenport e Patil (2012) reforçam que esta competência é fundamental em áreas técnicas, pois exige uma combinação de pensamento analítico e criativo.

Ao desenvolver essas habilidades, esta experiência permitiu criar uma base forte para uma carreira em ciência de dados, complementando os conhecimentos académicos com experiência real no dia a dia das empresas.

IV.3 Dificuldades Enfrentadas

O estágio foi uma experiência muito enriquecedora, que permitiu desenvolver competências técnicas e pessoais num ambiente profissional desafiante. Embora tenham sido encontradas algumas dificuldades, estas foram oportunidades de aprendizagem e crescimento e a equipa mostrou-se sempre disponível para apoiar e partilhar conhecimento.

Um dos desafios identificados foi trabalhar com sistemas obsoletos, algo comum em muitas empresas devido à evolução tecnológica natural. Como é referido Kleppmann (2017), estes sistemas, embora robustos e críticos para o negócio, podem apresentar complexidades como documentação reduzida ou código difícil de entender. Durante a migração da integração da Freja, verificou-se alguns destes cenários, o que exigiu uma abordagem cuidadosa e colaborativa.

Para garantir uma transição eficiente e segura, as estratégias adotadas incluíram:

- Análise detalhada do sistema existente, recorrendo a engenharia reversa para perceber melhor a sua estrutura;
- Modularização progressiva, decompondo o sistema em componentes mais claros e independentes;
- Implementação de testes para assegurar a manutenção do comportamento esperado durante a refatoração;
- Colaboração próxima com a equipa, que ajudou a contextualizar decisões técnicas e a acelerar a resolução de desafios.

Esta experiência permitiu não só aprofundar conhecimentos técnicos, mas também valorizar a importância do trabalho em equipa e da comunicação eficaz. A empresa proporcionou as ferramentas e o apoio necessários para superar estes desafios, reforçando a minha capacidade de lidar com cenários complexos no futuro.

No final, sinto que contribuí de forma positiva para o projeto, enquanto houve um crescimento profissional num ambiente que valoriza a melhoria contínua e a partilha de conhecimento.

Outra dificuldade foi a necessidade de manter compatibilidade com sistemas externos enquanto se implementavam melhorias arquiteturais. De acordo com Ereth (2018), a evolução das interfaces em sistemas distribuídos exige um planeamento cuidadoso para evitar o impacto negativo nas integrações existentes. Este desafio tornou-se especialmente importante durante a migração da API Rangel, onde era essencial garantir que os sistemas dos parceiros continuassem a operar sem interrupções, mesmo com as alterações em curso.

Para lidar com esta situação de forma estruturada, as abordagens adotadas incluíram:

- Manutenção de versões paralelas das APIs durante o período de transição, permitindo uma migração faseada e controlada;
- Comunicação clara e proativa com os utilizadores externos, informando-os atempadamente sobre as alterações planeadas e recolhendo *feedback* para ajustes necessários.

A gestão de vários projetos, cada um com os seus *stakeholders*, prazos e prioridades distintas, revelou-se um dos desafios mais complexos do estágio. Como é referido por Kerzner (2022), este tipo de cenário exige não só competências técnicas, mas também uma forte capacidade de organização e comunicação.

Para garantir que todos os projetos avançavam de forma coerente e eficiente, as estratégias implementadas incluíram:

- Priorização contínua das tarefas, com base nas que tinham mais impacto para o negócio e nas dependências técnicas identificadas;
- Comunicação clara e regular com todas as partes envolvidas, mantendo-as informadas sobre o progresso, problemas encontrados e necessidades específicas;
- Utilização de ferramentas de gestão de projetos, que permitiam visualizar o estado de cada iniciativa e distribuir eficazmente o trabalho;
- Definição de expectativas realistas desde o início, negociando prazos e entregas de forma transparente com todos os intervenientes.

Finalmente, a necessidade de dominar, em pouco tempo, novas tecnologias e conceitos específicos do setor da logística e transportes. Como referem Davenport e Patil (2012), a área de dados e tecnologia está em constante evolução, exigindo uma capacidade de aprendizagem rápida e adaptação permanente. Esta realidade manifestou-se na MixMove.

Para responder a este desafio, a abordagem foi estruturada da seguinte forma:

- **Dedicação de tempo específico** para aprendizagem, integrando-o na rotina de trabalho de forma consistente;
- **Exploração de múltiplos recursos**, desde documentação técnica, a cursos online e partilha de conhecimento com colegas mais experientes;
- **Aplicação prática imediata** dos novos conhecimentos, consolidando a aprendizagem através da resolução de problemas reais;
- **Avaliação contínua** das diversas lacunas e definição de prioridades para um desenvolvimento progressivo e focado.

Esta experiência reforçou a capacidade de aprender de forma autónoma e eficiente, algo que é considerado fundamental para qualquer profissional nesta área. A empresa proporcionou os recursos e o ambiente necessários para esta evolução, demonstrando o seu compromisso com o desenvolvimento da equipa.

IV.4 Pontos Fortes

A análise dos aspetos mais positivos do estágio permitiu identificar várias áreas que foram particularmente bem-sucedidas, tanto ao nível das soluções técnicas como no que diz respeito ao processo de aprendizagem e crescimento profissional.

Um dos pontos fortes foi a **modularidade e extensibilidade das arquiteturas implementadas**. De acordo com Richards (2022), decompor sistemas em componentes bem organizados, com interfaces claras entre eles, torna-se muito mais fácil manter e a evolução a longo prazo. A aplicação deste princípio de forma consistente nos projetos desenvolvidos, o que resultou em:

- APIs com responsabilidades claramente delimitadas
- Serviços independentes que podem ser mantidos e evoluídos separadamente
- Interfaces estáveis entre componentes, reduzindo o acoplamento
- Reutilização de código através de bibliotecas partilhadas

Esta abordagem modular provou ser útil quando foi necessário adaptar a integração da Enfega para suportar diferentes estratégias de distribuição das cargas nos veículos consoante as necessidades do cliente. Devido à arquitetura implementada, a introdução desta variação tornou-se possível com alterações mínimas, o que confirmou a eficácia das decisões técnicas tomadas.

Um dos aspetos mais pertinentes foi a abrangência e fiabilidade dos sistemas de validação implementados. De acordo com Batini e Scannapieco (2016) uma validação de dados eficaz é fundamental para assegurar a fiabilidade dos sistemas que os processam. As soluções desenvolvidas durante o estágio incluíram:

- Validação em várias camadas desde verificações básicas de formato até a regras de negócio complexas
- Tratamento adequado de erros, com mensagens claras e orientações para resolução
- Sistemas de notificação que avisam os utilizadores quando algum problema é detetado
- Registos detalhados (*logs*) que facilitam o diagnóstico e a correção de erros

A eficácia destes sistemas de validação ficou demonstrada através da redução significativa de erros, o que aumentou a confiança nas integrações realizadas.

Um dos aspetos mais relevantes do trabalho desenvolvido foi a documentação orientada para o utilizador. De acordo com Maalej e Robillard (2019), uma documentação bem estruturada é

fundamental para garantir a sustentabilidade a longo prazo dos sistemas. Nos projetos em questão, foram criados:

- Documentação clara das APIs, seguindo standards como OpenAPI/Swagger
- Uma *wiki* interna com detalhes sobre a arquitetura e implementação
- Guias de integração para utilizadores externos

Esta abordagem não só facilitou a adoção dos sistemas pelos utilizadores, como também permitiu uma melhor partilha de conhecimento dentro da equipa, assegurando que o trabalho desenvolvido durante o estágio continuará a ser útil para a organização.

Um aspeto fundamental foi a aplicação consistente de boas práticas no desenvolvimento de software. Como referem Martin (2019) e Beck (2004), técnicas como testes automatizados, integração contínua e refatoração frequente são indispensáveis para garantir a qualidade do código a longo prazo. Nos projetos em questão, as medidas adotadas incluíram:

- Testes unitários e de integração automatizados
- Revisão sistemática de código através de *pull requests*
- Refatoração contínua para manter a qualidade do código
- Integração contínua com verificações automáticas

Estas práticas contribuíram para criar um código mais robusto e sustentável, estabelecendo uma base sólida para futuras melhorias e evitando o acumular de dívida técnica.

A orientação para as necessidades do negócio revelou-se um ponto forte constante em todo o trabalho desenvolvido. Tal como expõem Davenport e Patil(2012), os sistemas de dados só demonstram o seu verdadeiro impacto quando estão alinhados com objetivos organizacionais concretos. Os projetos realizados destacaram-se pelos seguintes aspetos:

- Foco em problemas reais dos utilizadores, com soluções práticas e adaptadas às suas necessidades.
- Priorização clara, sempre baseada no impacto que cada decisão poderia ter no negócio.
- Ciclos de melhoria continua, com feedback constante das partes interessadas para ajustar e melhorar as soluções.
- Equilíbrio entre necessidades imediatas e a sustentabilidade a longo prazo, garantindo que as soluções não resolviam apenas problemas pontuais, mas também criavam bases sólidas para o futuro.

Esta abordagem focada no negócio garantiu que as soluções técnicas implementadas gerassem valor para a empresa, justificando assim todo o investimento feito nos projetos.

A reflexão sobre estes pontos fortes não só evidencia os sucessos alcançados durante o estágio, como também identifica boas práticas e metodologias que podem ser aplicadas em futuros projetos, tanto no âmbito profissional como académico.

V- Conclusão

O estágio realizado na MixMove, no âmbito do Mestrado em *Data Science* para empresas, foi uma experiência profundamente enriquecedora. A experiência proporcionou a oportunidade de aplicar e expandir os diversos conhecimentos teóricos num contexto empresarial real, o que permitiu crescimento tanto a nível profissional como pessoal.

Durante seis meses, no departamento de Desenvolvimento de Integrações, foram desenvolvidos projetos diversificados nas áreas de engenharia de dados, integração de sistemas e automação de processos. Os desafios enfrentados estavam direitamente ligados ao sector da logística e gestão de transportes, o que proporcionou uma visão prática sobre como a ciência de dados pode transformar operações complexas.

Avaliação do Cumprimento dos Objetivos

Objetivo Geral

O objetivo geral do estágio foi **desenvolver e modernizar soluções de engenharia de dados na empresa MixMove, com foco em integrações de sistemas, automação de pipelines e garantia de qualidade de dados**. Este objetivo foi plenamente atingido, conforme demonstrado pelos resultados concretos alcançados em cada um dos projetos desenvolvidos.

A modernização de soluções de engenharia de dados materializou-se através da migração de arquiteturas legadas para sistemas baseados em microserviços, resultando numa melhoria de 50% na eficiência de desenvolvimento de novas funcionalidades. O foco em integrações de sistemas foi concretizado através do desenvolvimento de soluções para quatro clientes principais (Volkswagen UK, Enfega, Freja e Rangel), cada uma com requisitos específicos e complexidades distintas. A automação de pipelines traduziu-se em melhorias de eficiência superiores a 85% em processos anteriormente manuais, enquanto a garantia de qualidade de dados foi assegurada através da implementação de sistemas de validação que reduziram a taxa de erros de 15% para menos de 2%.

Objetivos Específicos

Objetivo 1: Criar e implementar soluções de integração de dados para clientes específicos, como a Volkswagen UK e a Enfega

Este objetivo foi **completamente atingido**. O desenvolvimento e implementação incluiu soluções de integração para quatro clientes principais:

- **Volkswagen UK**: Criação de uma integração completa que automatizou o processo de geração de relatórios, reduzindo o tempo necessário de 6-8 horas para 15 minutos,

incluindo extração de dados de mais de 20 tabelas operacionais, aplicação de data *masking* para conformidade com ISO 27001, e transferência segura via SFTP com encriptação PGP.

- **Enfega**: Desenvolvimento de uma integração robusta para processamento de reservas através de ficheiros Excel, com implementação de validações multicamada que reduziram a taxa de erros de 15% para menos de 2%.
- **Freja**: Modernização completa do sistema de validação de dados, substituindo validações dispersas por um sistema centralizado baseado em FluentValidation.
- **Rangel**: Criação de pipelines automatizados para processamento de rotas com integração à API externa Routyn, reduzindo o tempo de processamento de 2 horas para 10 minutos.

Objetivo 2: Modernizar sistemas de validação de dados existentes, melhorando a qualidade e confiabilidade das informações processadas

Este objetivo foi **plenamente alcançado**, particularmente através do trabalho desenvolvido na integração da Freja. A modernização envolveu:

- Substituição de validações dispersas e inconsistentes por um sistema centralizado baseado em FluentValidation
- Implementação de validações em múltiplas camadas (sintáticas, semânticas, de domínio e de consistência)
- Criação de mensagens de erro claras e contextualizadas
- Desenvolvimento de validações assíncronas para verificações que requerem acesso a bases de dados
- Integração automática com a API, proporcionando feedback imediato aos utilizadores

Os resultados demonstram uma melhoria significativa na confiabilidade das informações processadas, com redução drástica de erros e melhoria na experiência dos utilizadores.

Objetivo 3: Desenvolver APIs RESTful e serviços web para facilitar a comunicação entre diferentes sistemas

Este objetivo foi **totalmente cumprido**. Foram desenvolvidas múltiplas APIs RESTful seguindo as melhores práticas:

- APIs para cada integração principal (Volkswagen UK, Enfega, Freja, Rangel)

- Implementação de padrões como *Repository*, injeção de dependências e *middleware* de autenticação
- Documentação automática usando Swagger/OpenAPI
- Aplicação de princípios RESTful com utilização adequada de métodos HTTP e códigos de estado
- Implementação de autenticação baseada em *tokens* e controlos de acesso rigorosos

As APIs desenvolvidas facilitaram significativamente a comunicação entre sistemas heterogéneos e permitiram a criação de uma arquitetura mais modular e escalável.

Objetivo 4: Automatizar processos de extração, transformação e carregamento de dados (ETL)

Este objetivo foi **excecionalmente bem-sucedido**. A automação de processos ETL foi implementada em múltiplos contextos:

- **Volkswagen UK**: Automação completa do pipeline de extração de dados de 20+ tabelas, transformação para formato CSV, aplicação de *data masking*, e carregamento seguro via SFTP
- **Azure Synapse Analytics**: Implementação de pipelines ETL robustos para processamento de grandes volumes de dados
- **Processamento de ficheiros**: Automação do processamento de ficheiros Excel, CSV e JSON com validação automática e tratamento de erros
- **Integração com APIs externas**: Automatização da comunicação com serviços como Routyn para enriquecimento de dados

Os resultados incluem melhorias de eficiência superiores a 85% em processos anteriormente manuais e eliminação completa de erros humanos na transferência de dados.

Objetivo 5: Estabelecer mecanismos robustos de monitorização e validação de dados

Este objetivo foi **atingido** através da implementação de sistemas abrangentes de observabilidade:

- Implementação de *logging* estruturado e detalhado em todas as integrações
- Criação de *dashboards* em tempo real para monitorização do estado dos sistemas
- Configuração de alertas automáticos para deteção proactiva de problemas
- Implementação de métricas de qualidade de dados e indicadores de desempenho

- Desenvolvimento de sistemas de notificação automática para utilizadores

O impacto foi significativo: redução do tempo médio de deteção de problemas de 4-6 horas para menos de 15 minutos, e melhoria substancial na capacidade de diagnóstico e resolução de incidentes.

Objetivo 6: Elaborar documentação técnica para suportar a manutenção das soluções desenvolvidas

Este objetivo foi **completamente realizado** com a criação de documentação abrangente:

- Documentação automática de APIs usando Swagger/OpenAPI com exemplos práticos
- Criação de *wiki* interna com detalhes de arquitetura e implementação
- Desenvolvimento de guias de integração para utilizadores externos
- Elaboração de manuais de instalação e configuração
- Documentação de processos e procedimentos operacionais

A documentação criada não só facilita a manutenção das soluções desenvolvidas, como também assegura a transferência de conhecimento e a sustentabilidade a longo prazo dos sistemas.

Objetivo 7: Aplicar metodologias ágeis no contexto de projetos de engenharia de dados

Este objetivo foi **completamente implementado** através da adoção consistente de práticas ágeis:

- Organização do trabalho em sprints de duas semanas
- Participação ativa em *daily stand-ups, sprint plannings* e retrospetivas
- Aplicação de princípios de melhoria contínua e feedback rápido
- Utilização de ferramentas de gestão de projetos para acompanhamento do progresso
- Adaptação rápida a mudanças de requisitos e prioridades

A aplicação destas metodologias resultou numa melhoria significativa na eficiência do desenvolvimento e no alinhamento constante com as necessidades dos *stakeholders*.

Reflexão Final

Um dos aspetos mais gratificantes foi verificar o impacto real do trabalho desenvolvido. A migração de integrações antigas para uma arquitetura moderna baseada em microserviços, o que trouxe maior flexibilidade e escalabilidade aos sistemas. No projeto da Freja, a implementação de mecanismos robustos de validação de dados, melhoraram significativamente a qualidade da informação processada. Além disso, o desenvolvimento de um pipeline de dados para a Volkswagen UK que reduziu processos que demoravam horas para meros minutos, eliminando erros manuais e reforçando a segurança.

A criação de APIs bem documentadas e modularizadas, facilitando a comunicação entre sistemas distintos e para a construção de infraestruturas de *data warehousing* e *reporting*, que trouxeram novas capacidades analíticas à empresa. Estes avanços traduziram-se em benefícios tangíveis: menos erros operacionais, maior eficiência e dados mais confiáveis.

A nível técnico, a aquisição de experiência com ferramentas como .NET Core, SQL e Azure, e foram aprofundados os conhecimentos em arquiteturas distribuídas e automação de pipelines de dados. No plano metodológico, trabalhou-se com *DataOps* e *Scrum*, o que demonstrou a importância da abordagem ágil nos projetos.

Mas mais do que as competências técnicas, valorizam-se as competências interpessoais desenvolvidas: o desenvolvimento da capacidade de comunicação com diferentes *stakeholders*, o trabalho em equipas com pessoas de diferentes áreas e a gestão de múltiplos projetos em simultâneo. Houve desafios, claro, desde lidar com sistemas obsoletos até assegurar compatibilidade durante processos de modernização—mas cada obstáculo foi uma oportunidade para aprender e evoluir.

No futuro, pretende-se aprofundar os conhecimentos em processamento de dados em tempo real e *data warehousing*. A experiência na MixMove proporcionou uma base sólida para continuar a crescer nesta área.

Em suma, este estágio foi um marco na presente formação. A experiência confirmou o interesse pela ciência de dados e demonstrou como o conhecimento teórico, quando aplicado na prática, pode gerar valor real. O cumprimento integral de todos os objetivos propostos, demonstrado através de resultados mensuráveis e impacto organizacional significativo, valida a eficácia da abordagem adotada e a qualidade do trabalho desenvolvido. Esta experiência permitiu alcançar uma maior preparação e reforçou a certeza de que o percurso escolhido é adequado para uma carreira promissora nesta área em constante evolução.

VI- Referências Bibliográficas

Abran, A., & Suryn, W. (2003). Usability meanings and interpretations in ISO standards. Software Quality Journal, 11, 325-338.

Atwal, H. (2019). Practical DataOps: Delivering Agile Data Science at Scale. Apress.

AWS. (s.d). What is a Data Pipeline? *AWS Documentation*. Disponível em: https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/what-is-datapipeline.html

Batini, C., & Scannapieco, M. (2016). Data and Information Quality: Dimensions, Principles and Techniques. Springer.

Beck, K. (2004). Extreme Programming Explained: Embrace Change (3ª ed.). Addison-Wesley.

Bertino, E., & Ferrari, E. (2018). Big data security and privacy. In A Comprehensive Guide to Secure Cloud Computing (pp. 115-134). Springer.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2021). Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media.

Calder, A. (2021). ISO/IEC 27001 Information Security Controls – A Guide to Implementation and Management. IT Governance Publishing.

Coronel, C., & Morris, S. (2015). Database Systems: Design, Implementation, & Management (11.ª ed.). Cengage Learning.

Dasu, T., & Johnson, T. (2003). Exploratory Data Mining and Data Cleaning. John Wiley & Sons.

Davenport, T. H., & Patil, D. J. (2012). Data Scientist: The Sexiest Job of the 21st Century. Harvard Business Review, October 2012.

De Bie, T., De Raedt, L., Hernández-Orallo, J., Hoos, H. H., Smyth, P., & Williams, C. K. I. (2021). Automating Data Science: Prospects and Challenges. arXiv. https://arxiv.org/abs/2105.05699

Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley.

Ereth, J. (2018, September). *DataOps – Towards a definition*. In Proceedings of the Lernen, Wissen, Daten, Analysen (LWDA 2018) (pp. 104–112). CEUR-WS.org. http://ceur-ws.org/Vol-2191/paper13.pdf

Feathers, M. (2004). Working Effectively with Legacy Code. Prentice Hall.

Fonseca, G. S. (2022). Optimization of supply chain processes with an ETL pipeline & data science applications [Dissertação de Mestrado, Universidade de Lisboa]. Repositório da Universidade de Lisboa.

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2013). Database Systems: The Complete Book (2ª ed.). Pearson.

Gregor, S., Chandra Kruse, L., & Seidel, S. (2020). The anatomy of a design principle. Journal of the Association for Information Systems, 21(6), 1622-1652.

Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley.

Kassen, N. (2022). Fundamentals of Data Engineering: Plan and Build Robust Data Systems. O'Reilly Media.

Kerzner, H. (2022). Project Management: A Systems Approach to Planning, Scheduling, and Controlling (13ª ed.). Wiley.

Kim, G., Humble, J., Debois, P., & Willis, J. (2021). The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations (2ª ed.). IT Revolution Press.

Kimball, R., & Ross, M. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling (3ª ed.). Wiley.

Kleppmann, M. (2017). Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media.

Kolb, D. A. (1984). Experiential Learning: Experience as the Source of Learning and Development. Prentice-Hall.

Ladley, J. (2019). Data Governance: How to Design, Deploy, and Sustain an Effective Data Governance Program (2ª ed.). Academic Press.

Lewis, J., & Fowler, M. (2014). Microservices: a definition of this new architectural term. martinfowler.com. [Disponível em: https://martinfowler.com/articles/microservices.html]

Maalej, W., & Robillard, M. P. (2019). Patterns of knowledge in API reference documentation. IEEE Transactions on Software Engineering, 42(12), 1083-1098.

Martin, R. C. (2002). Agile Software Development, Principles, Patterns, and Practices. Pearson Education.

Martin, R. C. (2019). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

Massé, M. (2011). REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. O'Reilly Media.

Mohanty, S., Jagadeesh, M., & Srivatsa, H. (2013). *Big data imperatives: Enterprise big data warehouse, BI implementations, and analytics*.

Myers, G. J., Sandler, C., & Badgett, T. (2015). The Art of Software Testing (3ª ed.). Wiley.

Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems (2ª ed.). O'Reilly Media.

Newport, C. (2016). Deep Work: Rules for Focused Success in a Distracted World. Grand Central Publishing.

Nguyen, T., Nguyen, H.-T., & Nguyen-Hoang, T.-A. (2025). Data quality management in big data: Strategies, tools, and educational implications. Journal of Parallel and Distributed Computing, 105067. https://doi.org/10.1016/j.jpdc.2025.105067

Provost, F., & Fawcett, T. (2013). Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking. O'Reilly Media.

Richards, M. (2022). Software Architecture Patterns (2ª ed.). O'Reilly Media.

Richardson, L., & Ruby, S. (2007). RESTful Web Services. O'Reilly Media.

Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. Proceedings of the VLDB Endowment, 11(12), 1781-1794.

Schutt, R., & O'Neil, C. (2013). Doing Data Science: Straight Talk from the Frontline. O'Reilly Media.

Smallwood, R. F. (2019). Information Governance: Concepts, Strategies and Best Practices (2ª ed.). Wiley.

Spinellis, D. (2003). *Code Documentation.* IEEE Software, 20(4), 94–95.

Stemplinger, J., Heinisch, P., & Bodendorf, F. (2023). Monitoring data quality in data pipelines: A systematic literature review. Decision Support Systems, 165, 113855.

Sutherland, J., & Schwaber, K. (2017). The Scrum Guide. Scrum.org.

Vernon, V. (2016). Domain-Driven Design Distilled. Addison-Wesley.

Warren, J., & Marz, N. (2015). Big Data: Principles and best practices of scalable real-time data systems. Simon and Schuster.

Weber, K., Otto, B., & Österle, H. (2009). One size does not fit all—A contingency approach to data governance. Journal of Data and Information Quality, 1(1), 1-27.

Wiegers, K. E., & Beatty, J. (2013). Software Requirements (3ª ed.). Microsoft Press.