INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Motor de Regras de Negócio

ANA SOFIA MOREIRA ANES DE SOUSA NEVES
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Motor de Regras de Negócio

Deteção de falhas ou anomalias

Ana Sofia Moreira Anes de Sousa Neves

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Paulo Oliveira

Supervisor: Rui Nunes

Porto, Outubro de 2022

“Persistence is the path to success.”

Charles Chaplin

Resumo

Numa organização, a informação é um recurso essencial para a progressão. Contudo, com a quantidade de dados que são gerados diariamente, há uma dificuldade em verificar possíveis anomalias ou falhas nos dados. Tendo em conta que a informação gerada numa organização é uma peça chave na tomada de decisão, torna-se fundamental realizar uma validação de que a informação apresentada não exibe anomalias.

Surgiu, então, a necessidade de desenvolver um motor de regras de negócio que verifique e garanta o cumprimento das regras parametrizadas no sistema. Este tipo de mecanismos detetam falhas, incoerências ou erros que não são detetáveis com uma análise agregada dos dados.

Este projeto apresenta uma solução que permite detetar e detalhar falhas encontradas no sistema auxiliando equipas de analistas de negócio e administradores de sistemas. Fornece, assim as informações necessárias para definir um plano de ação de correção para a anomalia detetada.

No presente documento foi descrito todo o estudo realizado e comparações de análises realizadas a temas fulcrais na resolução do problema apresentado pela organização. Por fim, será apresentada a implementação do projeto e análise de resultados obtidos com o intuito de verificar que o projeto corresponde ao objetivo estipulado.

Palavras-chave: Regras de Negócio, Motor de Regras, Deteção de Anomalias, Qualidade de Dados, Retalho de Moda

Abstract

Inside a corporation, the data is an essential resource to progression, especially with the amount of data that are generated per day nowadays, so there is a raising difficulty on verifying potential anomalies or missing parts in the collected data. Considering that the data generated inside a corporation is key to the decision-making process, it becomes essential to validate data without any anomalies.

Therefore, the necessity to developing a search engine of business rules which checks, with a guaranteed level, the fulfillment of the parameterized rules existing in the system. This type of mechanisms detects anomalies, inconsistences or errors that aren't detectable with an aggregated analysis of the data.

This project aims to provide a solution that allows the detection, with details, of missing parts in the system while assisting the business analysis teams and system administrators. Therefore, supplying necessary info to define a correction plan against the detected anomaly.

It is described, in the present document, the study elaborated as well as the analyzed comparisons made to essential themes to the resolution of the problem presented by the corporation. Lastly, the presentation of the project and the results analysis of the acquired data will be presented to verify that the project matches the agreed goals.

Keywords: Business Rules, Rule Engine, Anomaly Detection, Data Quality, Fashion Retail

Agradecimentos

Começo por agradecer ao ISEP, em especial ao departamento de informática e aos professores que se inserem neste departamento por toda a ajuda e conhecimentos partilhados com os alunos. De referir os professores Paulo Oliveira, Isabel Azevedo, Goreti Marreiros e Nuno Silva que geraram um grande impacto no meu percurso de mestrado.

Agradeço ao meu orientador, Professor Doutor Paulo Oliveira, que sempre se disponibilizou para ajudar no que fosse necessário e contribuiu com sugestões e críticas construtivas para um melhor resultado deste projeto.

Agradeço à empresa em que se encontra inserido este projeto pela oportunidade de realizar um projeto desafiante. Como o projeto não está diretamente relacionado com as minhas funções, permitiu-me adquirir novos conhecimentos.

Agradeço ao meu supervisor Dr. Rui Nunes pelo desafio proposto e confiança deposita na realização do projeto. A toda a equipa de BI igualmente, em especial ao Jorge Abreu e Fábio Baptista.

Agradeço há minha família pelo apoio na luta pelos meus objetivos, em especial aos meus pais e aos meus avós.

Por fim, agradeço aos meus amigos em especial: Márcia Pinto, Ana Nunes, Nelson Palmas, Pedro Sousa e Claúdia Silveira por toda a compreensão e motivação neste percurso.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

ADF	*Azure Data Factory*
AHP	*Analytic Hierarchy Process*
BD	base de dados
BI	*Business Intelligence*
BPM	*Business Process Management*
BPMN	*Business Process Modeling Notation*
BRM	*Business Rule Management*
BRMS	*Business Rule Management Systems*
DAMA	*International Data Management Association*
ELT	*Extract, Load and Transform*
ETL	*Extract, Transform and Load*
ISEP	Instituto Superior de Engenharia do Porto
IT	*Information Technology*
NCD	*New Concept Development*
PVP	Preço de Venda ao Publico
QFD	*Quality Function Deployment*
TOPSIS	*Technique for Order Preference by Similarity to Ideal Solution*
UI	*User Interface*

1 Introdução

No presente capítulo foi realizada uma apresentação breve do projeto, onde o leitor pode compreender o enquadramento, interesse científico, métodos utilizados, assim como o contexto e problema do projeto desenvolvido.

Por fim, foi apresentada a estrutura do presente documento, onde é disponibilizada a forma como o documento está organizado e o assunto abordado, sucintamente, em cada capítulo.

1.1 Enquadramento

As organizações apresentam cada vez mais uma necessidade de melhoria dos processos corporativos com os avanços tecnológicos constantes. A capacidade de entender, melhorar e gerir os processos existentes na organização é uma mais-valia para uma diminuição de custos de desenvolvimento e manutenção de sistemas informáticos [1], [2].

Com o intuito de melhorar e gerir os processos existentes na organização foram criados processos de extração, tratamento e carregamento de dados (ETL) pela equipa de *Business Intelligence* (BI), que validam regras de negócio. Como foram detetadas oportunidades de melhoria, surgiu a necessidade do presente projeto, um motor de validação de regras de negócio.

O conceito de motor de validação de regras de negócio sempre foi um conceito bastante abordado e estudado [3], no entanto a sua relevância é cada vez mais notória nos dias de hoje. Nas empresas cada vez mais se lida com processos automatizados e com grandes volumes de

dados em que é impossível realizar validações de regras de negócio de forma manual. Alem disso, qualquer processo que suporte uma tomada de decisão rápida e informada pelas equipas é uma mais-valia.

A verificação de regras de negócios auxilia os administradores de processos, garantindo o cumprimento das mesmas. Este tipo de mecanismos consegue detetar falhas, incoerências ou erros que não seria possível detetar através de uma análise agregada de dados, devido à quantidade de dados que é manipulada constantemente numa empresa.

1.2 Problema

Cada empresa apresenta um conjunto de regras de negócio distintas e uma arquitetura de sistema de informação única. Com o intuito de monitorizar se as mesmas são cumpridas, tendo em conta as especificidades da arquitetura do sistema existente, surge a necessidade de desenvolver um motor de verificação de regras de negócio.

A organização onde é inserido o presente projeto detetou que apesar de já serem realizadas validações de dados e disponibilizadas análises com anomalias/falhas detetadas, havia pontos de melhoria fundamentais para esta ferramenta de análise conseguir dar melhor resposta às necessidades dos utilizadores.

Os principais problemas detetados no processo utilizado foram a adição ou alteração de regras. Para cada nova regra a ser acrescentada era necessário acrescentar um novo processo de ETL, este processo era técnico e demorado, e no caso de ser necessária alguma alteração era necessário rever este processo.

Foi, também, referido como problema o facto de só ser possível identificar anomalias no caso de serem verificadas as análises disponíveis na plataforma Power BI, não havia possibilidade de, mal o alarme fosse detetado, notificar que o mesmo ocorreu. O que faz com que o tempo de resposta no caso de anomalia seja mais lento.

A solução apresentada anteriormente permitia a validação de regras de negócio, mas a adição de novas regras não era intuitiva e não era possível criar alarmísticas para notificação de problemas em tempo útil para uma resolução rápida. Surgiu, assim, a necessidade de dotar a empresa de um motor de regras de negócio.

Este processo permite melhorar as operações de monitorização de dados, consequentemente, auxiliar, de forma eficaz e rápida, na resolução de anomalias/violações que possam vir a surgir.

1.3 Objetivos

O objetivo principal do presente projeto consiste em desenvolver um motor de verificação de regras de negócio que gere alarmes e disponibilize dados capazes de representar anomalias/violações, para análise, permitindo resoluções rápidas e eficazes.

Estas verificações têm como foco principal, a melhoria dos processos de validação realizados por equipas de analistas de dados, assim como, melhorar as validações de processos de integração de dados nos sistemas de *Business Intelligence* (BI).

A solução final deve aceder à informação gerada pelas ferramentas de BI, realizar a verificação de regras de negócio através do motor de verificação e, por fim, apresentar anomalias detetadas através de análises e/ou através de alarmísticas. Os utilizadores finais deverão ter acesso aos resultados obtidos pelas análises e às alarmísticas geradas, para uma resolução o mais rápida e eficaz possível.

1.4 Abordagem

Esta tese de Mestrado tem, como foco principal, a criação do motor de validação de regras de negócio, mas incluiu também, a criação de novas análises assim como a adaptação das análises já existentes e criação de alarmísticas para anomalias detetadas que necessitam de uma ação mais imediata na sua resolução.

Numa primeira fase foi necessário explorar e analisar tecnologias, *workflows*, regras e dados. Nesta fase foi fundamental entender as necessidades/especificidades das regras de negócio, definir os requisitos do trabalho, o potencial fluxo do processo, cenários de teste, definição de regras e formas possíveis de apresentar os resultados obtidos.

Numa segunda fase foi definido o design e implementado o motor de verificação de regras de negócio, tendo em conta que este motor deve ser configurável para mapear novas regras. Foram também criadas análises e alarmísticas de anomalias detetadas.

Por fim, foram realizados testes ao motor de verificação de regras desenvolvido, assim como, uma avaliação e análise dos resultados obtidos.

Mais concretamente, para a resolução do problema indicado, criação do motor de regras de negócio, a criação de novas análises e alarmísticas, foram desenvolvidas da seguinte forma:

1. Estudar o conceito de regra de negócio, motor de regras de negócio e alarmísticas, assim como, o estudo de arquiteturas de motores de regras de negócio;
2. Reunir e representar regras críticas de negócio;
3. Conceber e desenvolver mecanismo genérico de configuração para mapear regras de negócio;
4. Desenvolver o processo de validação das regras parametrizadas;
5. Testar o processo implementado;
6. Avaliar e analisar os resultados obtidos.

1.5 Estrutura do Documento

Este documento foi dividido em 5 capítulos principais: Introdução, Estado da Arte, Solução Prévia, Análise de Valor, Análise, Proposta de Solução, Implementação, Avaliação e Experimentação e Conclusão.

No primeiro capítulo, Introdução, foi descrito o enquadramento e problema do projeto desenvolvido. É também efetuada uma apresentação dos objetivos do projeto e abordagem do mesmo.

No segundo capítulo, Estado da Arte, foi aprofundado o problema da empresa associado ao projeto enquanto estudados diferentes temas para solucionar o problema descrito da melhor forma.

No terceiro capítulo, Solução Prévia, foi realizada uma análise da solução disponível na organização aquando do início do projeto. Foi abordada a arquitetura da solução e os componentes da mesma.

No quarto capítulo, Análise de Valor, são consideradas diferentes técnicas que avaliam se o sistema projetado é capaz de dar resposta ao problema apresentado nesta dissertação.

No quinto capítulo, Análise, é descrito o processo de levantamento de requisitos. Neste capítulo, são abordadas as partes interessadas, os requisitos funcionais e os requisitos não funcionais do projeto.

No sexto capítulo, Proposta de Solução, foram apresentados o design projetado e o *design* final da solução. Assim, foi abordada a arquitetura da solução, as tecnologias utilizadas na implementação, a sequência dos processos, o modelo relacional do *dataset* e o diagrama de componentes do sistema.

No sétimo capítulo, Implementação, é detalhado o processo de implementação do projeto. Este capítulo foi dividido pelas diferentes implementações realizadas, são estas: o motor de regras, o processo de geração de alarmísticas e por fim, a parte analítica do projeto, onde são visualizados os resultados.

No oitavo capítulo, Experimentação e Avaliação, foi indicado como estava planeada a avaliação e experimentação do projeto realizado, tal como o nome indica. Para isso, foram identificadas grandezas e hipóteses associadas ao problema, foram apresentados indicadores de avaliação e fontes de informação utilizadas para validar estes indicadores, foi apresentada a metodologia de avaliação dos indicadores definidos, foram mostrados os testes realizados à solução implementada e, por fim, é realizada uma comparação entre a solução previamente utilizada e a solução desenvolvida.

No nono capítulo, Conclusão, foram abordados os resultados alcançados e as contribuições do projeto, assim como as limitações identificadas ao longo da execução do mesmo e possível trabalho futuro para evoluir a solução apresentada.

2 Estado da Arte

O presente capítulo descreve detalhadamente o contexto do projeto e o estudo realizado para o desenvolvimento do mesmo. Este estudo inclui a definição dos termos Regras de Negócio, Motores de Negócio e Alarmística, assim como, tecnologias e arquiteturas possíveis na resolução do problema.

2.1 Contexto

A necessidade do presente projeto surgiu em contexto empresarial dedicado à área de retalho de moda, mais concretamente inserido na equipa de *Business Inteligence* (BI). Este projeto pretende apresentar um motor de regras de negócio capaz de substituir o processo de validação de regras de negócio utilizado até então. O mesmo tem como propósito facilitar a adição de novas regras e antecipar a deteção de anomalias, para uma rápida resolução das mesmas.

Neste projeto, foram considerados exemplos de anomalias os seguintes casos:

- Talões de devolução sem talão de vendas associado;
- Talões de venda com um valor total superior a um certo montante;
- Transações realizadas fora do horário de funcionamento da loja;
- Diferença no valor de devolução de um artigo, tendo em conta o valor de venda do mesmo.

O sistema implementado pretende detetar este tipo de anomalias e permitir acrescentar novas regras

2.2 Regras de Negócio

Segundo Graham (2006), o conceito de regra de negócio foi inicialmente utilizado em meados da época de 80 por diferentes autores. As definições originais combinavam a definição de regras de negócio com restrições de bases de dados [2].

Em 1987, Ross define de forma geral uma regra de negócio, como uma política que rege o comportamento da empresa e a diferencia das demais. Mais tarde, em 1994, enfatiza que uma regra é uma declaração expressa [2].

Mais tarde, Morgan define regras de negócio como [4]:

"[…] uma declaração compacta sobre um aspeto de negócio. A regra pode ser expressa em termos que podem ser diretamente relacionados ao negócio, utilizando uma linguagem simples e inequívoca, acessível a todas as partes interessadas: administração, analistas de negócio, técnico, entre outros." (tradução livre do autor)

Regras de negócio são fundamentais na gestão empresarial de um negócio, são os meios pelos quais a organização implementa a sua estratégia competitiva, promove a politica de negócio e cumpre as obrigações legais [3]. Contudo, é necessário garantir que estas regras são definidas pelos devidos atores e de forma ponderada.

2.2.1 Definir Regras de Negócio

O processo de definição de regras é iterativo e heurístico. As Regras começam por ser declarações gerais de políticas que cabe aos funcionários traduzir em declarações especificas [5]. As regras devem ser definidas pelo negócio, apesar da equipa de IT ser responsável pela manutenção do sistema e ter conhecimento de aplicativos de negócio, não devem apresentar a responsabilidade de gerar novas regras [6].

Inicialmente, as políticas devem ser decompostas em regras de negócio atómicas, que são declarações formais e especificas de um termo, facto ou derivação. Deve ser avaliada a

estabilidade da regra, visto que a regra não deve invalidar outras regras já existentes no sistema [5].

2.2.2 Tipos de Regras de Negócio

Segundo Rowland Watkins [5], existem três tipos de regras de negócio:

- Regras de afirmação estrutural (*structural assertion*) –declaração de um facto que expressa um aspeto estrutural da empresa;
- Regras de afirmação de ação (*action assertion*) – declaração de uma restrição ou limitação de ações da empresa;
- Regras de derivação (*derivation*) – declaração de conhecimento derivada de outro entendimento do negócio.

Na Figura 1 estão representados os diferentes tipos de regras supra mencionadas.

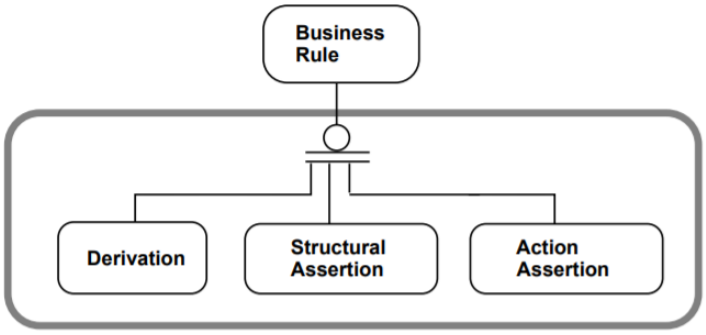


Figura 1 – Tipos de Regras de Negócio [5]

2.2.3 Exemplos de Regras de Negócio

Nem todos os tipos de regras devem ser automatizadas ou especificadas. Para o sucesso de uma abordagem de regras de negócio é fundamental avaliar o valor especifico das regras para o negócio [3]. Neste contexto em específico, retalho, podemos ter os seguintes exemplos como regras de negócio:

- Uma devolução deve apresentar um talão de troca;

- Vendas acima de um certo valor/quantidade devem ser identificadas, para serem posteriormente validadas;
- Se não houver stock na loja não é possível realizar a venda;
- No caso de haver menos de 2 artigos na loja deve ser enviado mais stock do artigo para a loja;
- Caso a percentagem de vendas integradas estiver abaixo de 20% é necessário verificar o processo de integração de vendas.

2.3 Motores de Regras de Negócio

Um mecanismo de regras de negócio compara o objeto de dados enviado com as regras de negócio carregadas no mesmo e ativa regras parametrizadas de acordo com a logica das mesmas [7]. Ronald Ross classifica mecanismos de regras de negócio em dois grandes grupos: mecanismos que lidam diretamente com bases de dados e mecanismos de inferência [6].

Maior parte dos sistemas de uma organização estão ligados a bases de dados, mais especificamente sistemas que tem como principal foco a gestão de informação. Este tipo de sistemas tem a necessidade de apresentar mecanismos capazes de lidar diretamente com a base de dados, sendo a sua aplicação fundamental com vista a garantir que estes tipos de sistemas refletem a lógica de negócio [6].

Mecanismos de inferência estão associados a sistemas de conhecimento ou de inteligência artificial. São utilizados para tirar conclusões através da análise de problemas. Este tipo de mecanismos tem a capacidade de dar resposta com base num conjunto de regras ou extrair conhecimento através de dados de outros sistemas [6].

2.3.1 Componentes do motor de regras

Pode-se destacar três componentes num motor para validação de regras típico, como é representado na Figura 2. São estes componentes a Base de Regras, Base de Factos e Mecanismo de inferência [7].

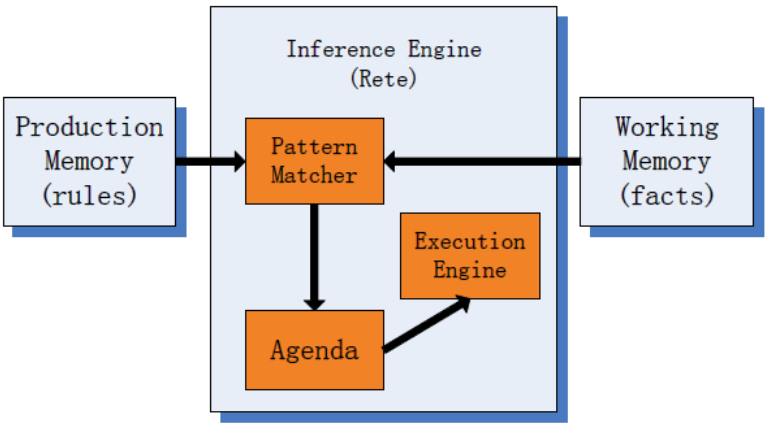


Figura 2 – Componentes típicos de um mecanismo de regras de negócio [7]

Na Base de Regras são armazenadas as regras de negócio, já a Base de Factos apresenta dados de negócio. O Mecanismo de Inferência, por sua vez, apresenta três blocos distintos: Comparador de padrões, em que são comparados os padrões que traduzem as regras com dados de negócio para verificar quais são as regras aplicadas aos dados de negócio a validar; Agenda, onde é gerida a sequência de execução das regras; e Mecanismo de execução, que executa a verificação das regras [7].

2.3.2 Atores do Sistema

Numa abordagem de regras de negócio é necessário avaliar o valor de regras especificas para o negócio, definindo com que frequência o negócio terá necessidade de alterar [3].

Um ponto importante na definição do design do motor de regras de negócio são os diferentes atores envolvidos na definição de regras de negócio, tal como os diferentes *timings* de definição das regras. O motor de regras de negócio deve ser, então, adequado ao tipo de utilização.

É possível identificar como possíveis atores de regras de negócio os seguintes [6]:

- Analistas de Negócio – profissionais responsáveis por desenvolver uma interface entre área de negócio e IT. Tipicamente, apresentam domínio na área de negócio que representam, o que os torna capazes de definir requisitos e regras de negócio. São assim uma mais-valia na projeção de parâmetros de definição dos tipos de regras.

- Operadores de Sistema – são como administradores de sistemas de produção, garantem que as aplicações são executadas. Apesar do conhecimento que apresentam dos sistemas, não são considerados uma peça fundamental na definição de regras de negócio, pois podem não apresentar conhecimento de negócio suficiente para tal.

- Consultores – Pessoa exterior à empresa contratada para uma tarefa específica. Espera-se que tenham conhecimento de negócio, mas um menor entendimento da organização comparado com analistas de negócio. Podem estar familiarizados com mecanismos de regras de negócio e compreendem processos envolvidos na definição de regras. São, assim, bons candidatos no processo de definição de regras.

- Profissionais de Conhecimento de Negócio – utilizam o *output* dos sistemas de produção para executar operações de negócio ou para tomada de decisão. Necessitam de ser convencidos da utilidade de um mecanismo de regras, visto que vai afetar o seu trabalho. Apresentam conhecimentos de regras subjacentes ao domínio de negócio em que trabalham. Obter conhecimento destes profissionais pode ser uma mais-valia na definição de regras de negócio. Deve ser evitado que estes profissionais nomeiem operadores de sistemas para definir as regras de negócio.

- Gestores de Negócio – determinam se o mecanismo de regras é necessário para uma determinada área de negócio. Não se espera que estejam envolvidos na definição de regras. Necessitam saber quem define regras de negócio e o impacto que o processo apresenta nas operações normais de negócio.

- Administração de Sistemas – Equipa que gere recursos de *software* e *hardware* na empresa. Tendo em conta a estratégia de implementação de regras, pode implicar alterações no ambiente em que é implementado o motor de regras. Embora a equipa não defina regras, pode ser necessário trabalhar com eles para que não haja nenhum problema na definição de regras.

- Programadores – necessários para a construção do motor de regras de negócio. Por vezes tendem a resolver problemas introduzindo código nos procedimentos, o que vai retirar a funcionalidade a um motor de regras de negócio, mas no caso do motor de

regras de negócio deve-se evitar a dependência de um programador para definição das regras.

2.4 Qualidade de Dados

Com a variedade aplicacional existente nas organizações, os dados cada vez mais se encontram dispersos pelos diferentes sistemas. É aqui que a qualidade de dados desempenha um papel fundamental no desempenho dos processos operacionais, na tomada de decisão e na cooperação organizacional [8].

O conceito de qualidade de dados pode ser definido através de diferentes perspetivas. Para um consumidor de dados a qualidade dos dados permite a utilização de dados adequados para consumo. Uma definição mais técnica da qualidade de dados considera que os dados apresentam uma alta qualidade, livres de defeitos e em conformidade com as especificações necessárias [9].

2.4.1 Dimensões de Qualidade de Dados

A qualidade de dados apresenta um conceito multidimensional [9] podendo assim ser medida por diferentes dimensões [10]. Estas dimensões são utilizadas para definir e medir a qualidade dos dados, separando assim os dados pelas suas características e aspetos [11].

Segundo a DAMA, as principais dimensões identificadas para garantir a qualidade de dados são: atualidade; completude; consistência; precisão; singularidade e validade [12]. A Figura 3 mostra uma representação gráfica das diferentes dimensões de qualidade de dados.

Figura 3 – Dimensões de qualidade de dados [12]

Neste projeto, para além das dimensões representadas na figura anterior, foram também abordadas as dimensões de integridade e conformidade. Estas dimensões foram necessárias na resolução do problema do projeto.

- **Atualidade**

A dimensão de temporalidade, do inglês *timeliness*, refere-se ao atraso entre a atualização da informação no mundo real comparativamente à integração no sistema [13]. Numa base de dados esta validação é realizada com as datas disponíveis nos registos de dados.

Um exemplo de validação desta dimensão pode ser a data da última atualização dos registos numa determinada tabela tendo em conta a frequência com que os valores devem ser atualizados.

- **Completude**

A dimensão de completude, do inglês *completeness*, permite determinar se um conjunto de dados descreve o mesmo de forma correspondente com o objeto real. No caso de uma base de dados relacional a completude está, tipicamente, relacionada com a existência de valores nulos. O valor nulo representa a ausência de valor, no caso de este existir no objeto real, a falta dele representa uma falha na qualidade dos dados [8].

Pode-se considerar uma falha nesta dimensão a falta do campo morada numa encomenda de loja online com entrega ao domicílio.

- **Conformidade**

A dimensão de conformidade, do inglês *conformity*, valida se os valores dos dados dos mesmos atributos são representados num determinado formato [14]. Pode-se considerar como exemplo o formato do campo de código postal, deve incluir 4 dígitos, hífen, seguido de 3 dígitos.

- **Consistência**

A dimensão de consistência, do inglês *consistency*, refere-se ao incumprimento da semântica de um conjunto de dados. A validação desta dimensão deve corresponder a um determinado intervalo de valores, deve ser considerada falha no caso de não respeitar o conjunto de valores que deve apresentar [8].

Pode-se considerar como exemplos deste tipo de dimensão os metros quadrados de uma loja que pode estar compreendida entre 30 metros quadrados e 200 metros quadrados ou a marca associada a um produto deve corresponder a um determinado grupo de marcas.

- **Integridade**

A dimensão de integridade, do inglês *integrity*, valida se uma restrição na base de dados relacional é aplicada entre dois conjuntos de dados [11]. Por exemplo, na tabela de vendas é acrescentado o identificador do produto vendido, este identificador tem de estar presente na tabela de produtos para ser possível associar o produto à venda. Para validar este caso é necessário verificar o identificador em ambas as tabelas.

- **Precisão**

A dimensão de precisão, do inglês *accuracy*, é definida como uma medida em que os dados são confiáveis e certificados. Os dados armazenados devem corresponder aos valores do mundo real. Podem ser distinguidos dois tipos de precisão: sintática e semântica. Esta dimensão, nas metodologias de qualidade de dados só é validada sintaticamente e é definida apenas a proximidade de um valor [8].

- **Unicidade**

A dimensão de unicidade, do inglês *uniqueness*, permite garantir que a informação é exclusiva no sistema, tem a intenção de validar a existência de duplicados [15]. No caso de uma base de dados relacional, esta validação deve ser utilizada em colunas chave das tabelas.

Pode-se considerar como um exemplo de validação desta dimensão a validação de códigos de produtos ou códigos de lojas.

- **Validade**

A dimensão de temporalidade, do inglês *validity*, descreve a proximidade dos valores com valores predeterminados ou provenientes de um cálculo [12]. Pode-se considerar um exemplo o cálculo do valor de vendas brutas e o valor de vendas líquidas, a diferença entre estes valores deve ser o valor das taxas aplicadas à venda.

2.5 Projetos Relacionados

Foram analisados os seguintes casos de estudo na área de mecanismos de regras:

- *Rule Engine Research and Implementation in Financial System* [16]
- *When Rule Engine meets Big Data: Design and Implementation of a Distributed Rule Engine using Spark* [17]
- *Adaptive Rule Engine for Anomaly Detection in 5G Mobile Edge Computing* [18]

Nas secções seguintes são abordados os diferentes casos de estudo.

2.5.1 Rule Engine Research and Implementation in Financial System [16]

Neste artigo é descrita a implementação de um motor de regras num sistema financeiro, designado *Financial Rule Manager* (FRM). O motor de regras de negócio interage com o aplicativo de trabalho, mas o processamento de regras de negócios é independente do componente aplicativo de trabalho.

O FRM inclui uma Base de gestão de regras (*Rule Base Manager*), um configurador de regras (*Rule Configurator*) e um processador de regras (*Rule Processor*). O aplicativo de trabalho chama a API do FRM, que por sua vez, vai consultar os conjuntos de regras para o cliente atribuído e depois de executar esses conjuntos de regras, retorna o resultado para o componente de fluxo de trabalho. Na Figura 4 está representada a arquitetura do sistema.

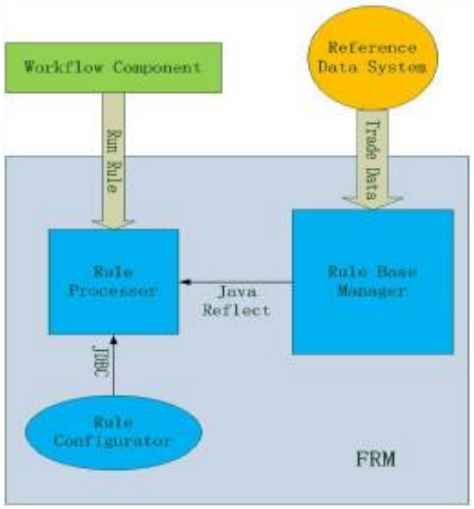


Figura 4 – Arquitetura do Financial Rule Manager [16]

A Base de Gestão de Regras contem a Base de Regras (*Rule Base*), onde as regras são guardadas em formato de texto com o formato IF e THEN ELSE; o Compilador de Regras (*Rule Compiler*), que transforma as regras em executáveis binárias através de classes Java; e, por fim, a Base de conhecimento (*Binary Rule Base*), que é onde as regras após serem compiladas e transformadas em regras binárias são guardadas numa base de dados. Na Figura 5 está representada a arquitetura da Base de Gestão de Regras.

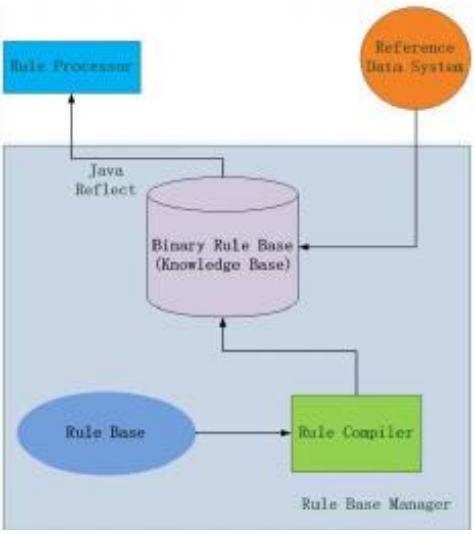


Figura 5 – Arquitetura da Base de Gestão de Regras [16]

Para configurar regras, neste projeto, é acedido ao *Reference Data System*. Neste é utilizada interface gráfica (UI) do configurador de regras que corresponde aos requisitos de um sistema financeiro.

2.5.2 When Rule Engine meets Big Data: Design and Implementation of a Distributed Rule Engine using Spark [17]

Este artigo descreve a implementação e arquitetura de um sistema de motor de regras em *Spark*, que suporta *big data*.

Este projeto utiliza como Base de trabalho várias fontes distintas, visto tratar-se de *big data*. Estas fontes são estruturadas no *DataFrame* de *Spark*, visto que é facilmente construído, apresenta uma distribuição de linhas com o mesmo *schema* e pode ser guardado e processado em forma de coluna.

Foi implementada uma linguagem de regras especifica para o projeto, que permite interpretar as regras pelo sistema e que transmite facilmente o conhecimento do utilizador.

O mecanismo de inferência é constituído por 3 fases: resolução de conflitos, onde são classificadas as regras por prioridade; comparador de padrões, onde é realizada a correspondência entre regras e armazenadas; e mecanismo de execução, onde são executadas as ações do conjunto de regras.

2.5.3 Adaptive Rule Engine for Anomaly Detection in 5G Mobile Edge Computing [18]

No artigo é descrito a implementação e arquitetura de um motor de regras de negócio acrescentado a um sistema de dados de rede de transmissão 5G. Inicialmente são extraídos dados de diferentes sensores e agregada a informação dos sensores para ser guardada no sistema.

Neste motor de regras de negócio, para criar a Base de Regras são combinados dois processos distintos. No primeiro processo, a partir da base dos dados extraídos dos sensores, o sistema aplica pré-processamento de dados e em seguida *Data Mining* e *Feature Extraction* para detetar padrões característicos dos dados e os considerar como regras. No segundo processo são extraídas regras através de conhecimento especializado. Estes dois processos combinados dão origem ao modelo de regras de associação (*Association Rules Model*). Na Figura 6 pode-se verificar esta arquitetura no *Rule Extraction Submodule*.

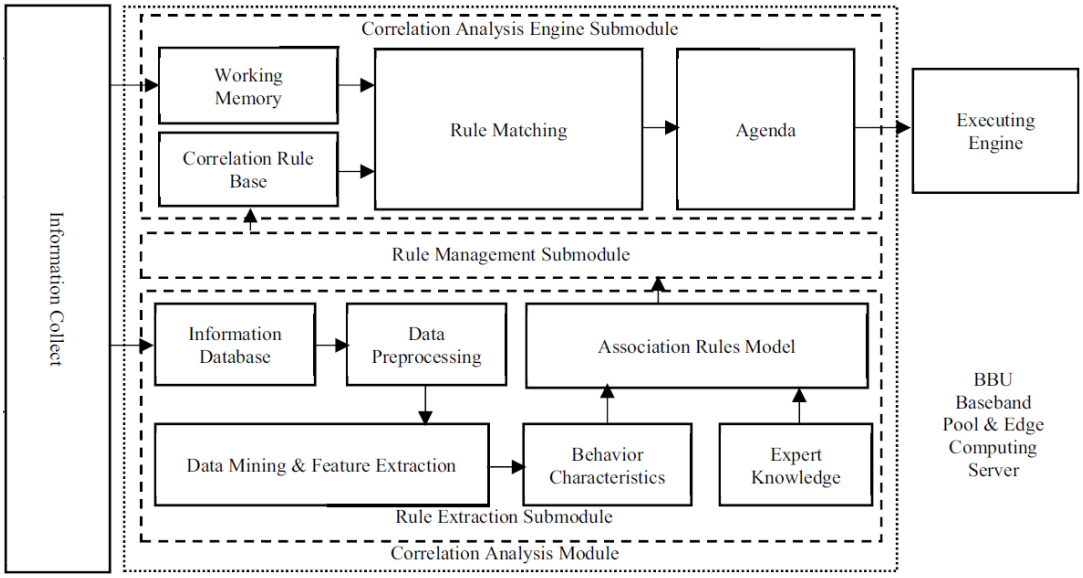


Figura 6 – Arquitetura do sistema de dados de rede de transmissão 5G [18]

A partir do modelo de regras de associação é passado a um módulo de gestão de regras (*Rule Management Submodule*), onde é mantida a correlação das regras com a Base de Regras de correlação (*Correlation Rule Base*).

Após a Base de Regras de correlação estar definida é realizado *Rule Matching* para combinar as regras de negócio com a informação proveniente de sensores. Na Agenda é definida a sequência de execução de regras e, por fim, o Mecanismo de execução, executa as regras.

2.5.4 Comparação de projetos

Em suma, tendo em conta todos os sistemas estudados foi realizada uma comparação de conceitos com o sistema proposto neste projeto de mestrado, disponível na Tabela 1.

Tabela 1 – Comparação de projetos estudados

Projetos/ conceitos	***UI***	***Data Mining***	***Rule Matching***	***Agenda***	**Mecanismo de Execução**	***Azure***	***Spark***	***Java***
2.5.1	✓	✗	✗	✗	✓	✗	✗	✓
2.5.2	✗	✗	✓	✓	✓	✗	✓	✗
2.5.3	✗	✓	✓	✓	✓	✗	✗	✗
Presente Projeto	✗	✗	✗	✗	✓	✓	✗	✗

✓ - conceito abordado ✗ - conceito não abordado

2.6 Tecnologias

Nesta secção são descritas tecnologias que podem ser utilizadas nos diferentes módulos do sistema proposto. Para cada uma das fases são apresentadas as tecnologias e realizada uma comparação das mesmas.

2.6.1 Tecnologias de motores de regras de negócio

Para realizar o motor de regras de negócio foram verificadas tecnologias *Azure*, *Python* e *Drools*, detalhadas nas seguintes subsecções.

2.6.1.1 Azure

O *Microsoft Azure* apresenta um motor de regras, associado ao *Azure Front Door*, que providencia um controlo do comportamento para uma aplicação *web* [19], mas este motor de regras não corresponde ao propósito do presente projeto. Para o propósito deste projeto são verificadas as ferramentas *Data Lake*, *Data Factory* e *Databricks*.

- **Data Lake**

O *Data Lake* é um repositório de armazenamento de grandes quantidades de dados que geralmente proveem de várias fontes independentemente da forma como estão estruturados. Idealmente, no *Data Lake* devem ser armazenados os dados no seu estado original. Pode ser utilizado para consolidar dados da organização num só local central onde podem ser

armazenados tal como se encontram na fonte, sem haver necessidade de uma estrutura formal de como os dados são organizados [20].

No *Data Lake* os dados em bruto podem ser armazenados junto de dados tabulares estruturados para utilização da organização, assim como, dados intermédios gerados no processo de transformação de dados em bruto. O *Data Lake* também permite armazenar dados não estruturados e semiestruturados, como imagens, vídeo ou documentos que podem ser utilizados para *machine learning* ou outros tipos de análises mais avançadas [20].

- **Data Factory**

O *Data Factory* é um serviço *cloud* que gere projetos de extração, transformação e carregamento de dados (ETL), extração carregamento e transformação (ELT) e integração de dados. Permite criar fluxos de dados para transformar e movimentar dados em escala [21].

O *Data Factory* possibilita a criação de *workflows* complexos de transformação de dados e permite criar agendamentos para os mesmos, designados *pilpelines* [21].

- **Databricks**

O *Databricks* é uma plataforma de análise de dados utilizada para serviços *cloud Microsoft Azure*. Apresenta três ambientes para desenvolver aplicações com uso intensivo de dados: *Databricks SQL*, *Databricks Data Science & Engineering* e *Databricks Machine Learning* [22].

O *Databricks SQL* permite realizar consultas e outros tipos de comandos *SQL* ao *Data Lake* e criar diferentes tipos de visualizações de dados para explorar resultados de consultas [22].

O *Databricks Data Science & Engineering* disponibiliza um *workspace* interativo que permite a partilha entre colaboradores com acesso ao *Databricks*. Os dados podem ser carregados através do *Data Factory* ou através de *stream near real-time* no *Data Lake*. Como análise, também é possível realizar *insights* através de *Spark* [22].

O *Databricks Machine Learning* é um ambiente de *machine learning* que disponibiliza serviços de treino de modelos, gestão e implementação de funcionalidades, entre outros [22].

2.6.1.2 Python

O *Python* é uma linguagem de programação interpretada e interativa. Suporta vários paradigmas de programação: programação orientada a objetos, programação procedimental e funcional. Apresenta interfaces para chamadas de sistema e bibliotecas [23]. Uma das grandes mais valias da utilização de *python* é a quantidade de bibliotecas disponíveis nesta linguagem.

Para a realização de um motor de regras de negócio foram verificadas algumas bibliotecas disponíveis de *python*:

- **Durable-rules**: é uma *micro-framework* utilizada para coordenação de eventos em tempo real. Permite detetar e analisar informações de eventos ao combinar dados de várias fontes. A implementação *forward chaining* é utilizada para avaliar factos e fluxos de eventos em tempo real. Esta biblioteca é implementada na linguagem C [24].

- **Rule-engine**: esta biblioteca define regras como *strings* com uma síntese de *Python*, mas com algumas semelhanças de Ruby. Destina-se a permitir *developers* a filtrar objetos *python* com uma regra especificada [25].

- **Business-rule-engine**: esta biblioteca fornece uma interface que permite qualquer utilizador capturar noves regras que definem o comportamento de um sistema e uma forma de processar as regras [26].

2.6.1.3 Drools

O *Drools* é uma solução de *Business Rules Management System* (BRMS) que faculta um motor de regras de negócio central, uma aplicação de criação na web e gestor de regras, um suporte de tempo de execução para modelos *Decision Model and Notation* (DMN) e um plug-in Eclipse para desenvolvimento Java [27]. O *Drools* apresenta duas partes principais: o processo de autoria e processo de execução [28].

No processo de autoria, representado na Figura 7, são criados arquivos para regras que são mantidas através de um *parser*. Este *parser* verifica a gramática das regras e produz uma estrutura intermédia para a descrição das mesmas. No fim do processo de autoria, o *Package Builder* gera e compila o código final para gerar o package final de regras [28].

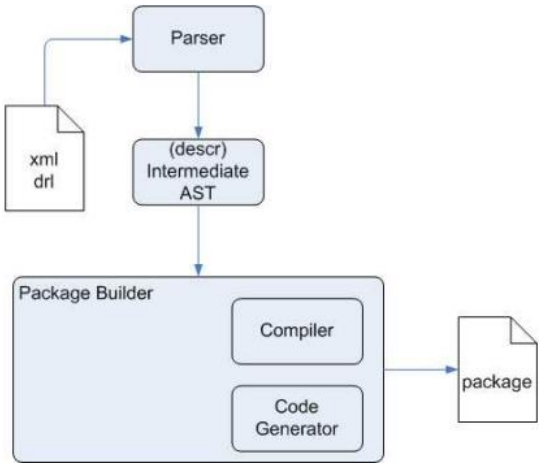


Figura 7 – Componentes de autoria do *Drools* [28]

O processo de execução, representado na Figura 8, através do package gerado no processo anterior extrai a Base de Regras que vai executar na *Working Memory*. Este processo pode instanciar mais do que uma *Working Memory* para um ou mais packages. A *Working Memory* contém vários subcomponentes, *Working Memory Event Support*, *Truth Maintenance System*, e *Agenda* [28].

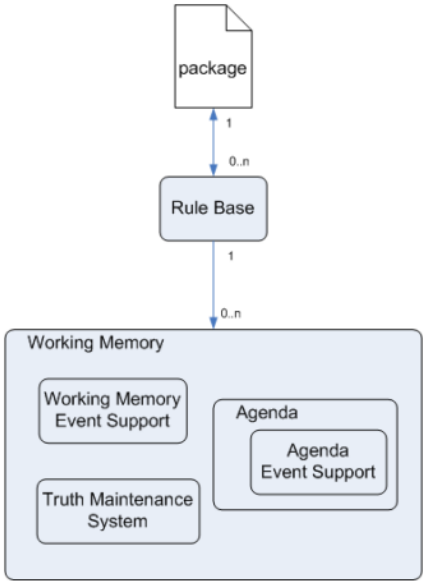


Figura 8 – Componentes do processo de execução do *Drools* [28]

2.6.1.4 Comparação

Com o intuito de analisar as diferentes tecnologias estudadas foi construída a Tabela 2. Nesta tabela são comparadas as vantagens e desvantagens da utilização das diferentes tecnologias, assim como é identificado quais destas são *open source*.

Tabela 2 – Comparação de tecnologias de motores de regras de negócio

Tecnologias	*Open Source*	Vantagens	Desvantagens
Azure	✗	• Alta disponibilidade; • Segurança de dados; • Escalabilidade; • Custo efetivo.	• Requer gestão • Requer conhecimento da plataforma
Python	✓	• Versátil, e de rápida aprendizagem; • A quantidade de bibliotecas disponíveis; • Indicado para realizar protótipos.	• Problemas de performance, é mais lento que outras linguagens; • Problemas com processamento paralelo; • Consumo de memória.
Drools	✓	• Desempenho razoável, por causa do algoritmo utilizado; • Traduz requisitos em regras; • Capacidade de aplicar gestão empresarial às regras; • Permite alterar e regras sem parar a aplicação.	• Não recomendado para projetos com menos de 20 regras; • O *Drools* apresenta uma curva de aprendizagem. É necessário verificar a sintaxe e a execução das regras.

Tendo em conta a comparação realizada, o *Drools* poderia ser uma opção viável para a resolução do problema uma vez que é uma solução *open source* especifica para a implementação de um motor de regras de negócio mas, visto que a organização já aplica tecnologias *Azure*, é um requisito desenvolver o projeto com tecnologias *Azure*.

2.6.2 Tecnologias de análise

Para realizar as análises, foram verificadas as tecnologias Power BI e Tableau, detalhadas nas seguintes subsecções.

2.6.2.1 Power BI

O Power BI apresenta um conjunto de serviços que permitem tornar dados não relacionados em informação coesa representada através de *report* dinâmicos. Possibilita a extração de dados de variadas fontes para realizar *reports* e visualizações de dados que podem ser necessários para uso pessoal ou para análises a nível corporativo [29].

O Power BI apresenta 3 plataformas distintas: o Power BI d*esktop*, Power BI *service* e Power BI *mobile*. Estes são utilizados para criar, partilhar e visualizar *reports* de negócio [29].

Tipicamente, o fluxo de trabalho no Power BI começa pela criação do *report* no Power BI *Descktop*, onde é efetuada a ligação às fontes de dados, por vezes, são acrescentados campos através da utilização da linguagem DAX e criada a visualização dos dados. De seguida o *report* é publicado no Power BI *service* e partilhado com os utilizadores da organização. Ao partilhar o *report* no Power BI *service* este também fica disponível no Power BI *mobile* [29].

2.6.2.2 Tableau

O *Tableau* é uma ferramenta de exploração, análise e apresentação de dados. Apresenta uma versão gratuita, *Tableau Public*, e versões para desktop, servidor e online pagas. As principais diferenças entre a versão gratuita e a versão paga são [30]:

- a variedade de fontes de dados que podem ser utilizadas – menos opções na versão publica;
- a forma como são partilhados os ficheiros – a versão paga pode partilhar com a versão gratuita, o que não acontece na versão gratuita;
- a forma de guardar ficheiros – a versão paga permite guardar localmente ou online, enquanto a versão publica só permite guardar online.

A versão gratuita pode ser uma boa forma de adaptação ao software, mas os ficheiros criados pelo utilizador estão associados ao perfil do mesmo e estão públicos para o acesso de qualquer pessoa [30].

2.6.2.3 Comparação

Com o intuito de comparar as tecnologias estudadas para representação de análises, foi construída a Tabela 3. Nesta tabela são comparadas as vantagens e desvantagens da utilização das tecnologias, assim como é identificado se são tecnologias *open source*. De notar que ambas as tecnologias apresentam uma versão gratuita, mas nesta comparação está a ser considerado um contexto empresarial em que as versões gratuitas não dão resposta às necessidades do projeto.

Tabela 3 – Comparação de tecnologias de análise

Tecnologias	Open Source	Vantagens	Desvantagens
Power BI	X	• Custo em relação ao *Tableau* é mais barato • Aprendizagem rápida da utilização dadas as semelhanças com Excel • *Updates* constantes de funcionalidades • Quantidade de fontes de dados que podem ser utilizadas • Integração com Excel • Visualizações customizáveis	• Interface apresenta bastante informação • Linguagem DAX ainda apresenta algumas limitações • Capacidades de dados nas versões gratuitas • Configurar visuais apresenta algumas limitações • Relações entre tabelas apresenta regras que por vezes não possibilita ligações
Tableau	X	• Utilização intuitiva • Vários tipos de visualizações • Quantidade de fontes de dados que podem ser utilizadas • Utilização de diferentes linguagens de *scripting* • Aplicação *mobile* eficiente	• Custo elevado • Não apresenta agendamento ou notificações de *reports* • Não permite customizar visuais • Parâmetros apenas estáticos e com apenas um valor • Limite de processamento de dados

Considerando a comparação realizada, ambas as tecnologias são reconhecidas pelas visualizações apresentadas, no entanto o Power BI permite personalizar visualizações ao contrário do *Tableau*, apresenta um custo não tão elevado e adaptável à utilização da organização, daí ser considerado o Power BI como a ferramenta mais indicada a resolução do problema.

Neste caso a tecnologia definida pela organização é também o Power BI, por já ser uma tecnologia aplicada na organização.

3 Solução Prévia

Tal como referido, o presente projeto tem como vista a melhoria de uma solução já existente na organização. A solução que já se encontrava disponível na organização foi construída com o intuito de dar resposta à necessidade de analisar dados de vendas e devoluções. Por ser uma solução focada neste tipo de análises, limita a realização de outro tipo de validações que possam ser necessárias.

No presente capítulo é apresentada a arquitetura da solução existente e os diferentes componentes da mesma.

3.1 Arquitetura

A solução previamente desenvolvida utiliza a definição de alertas, que são casos em que é detetada uma possibilidade dos dados não corresponderem ao normal. Os casos detetados como alertas são posteriormente analisados por equipas específicas para que, através uma análise mais exaustiva, sejam detetadas possíveis falhas nos sistemas ou anomalias.

Estes alertas são parametrizados diretamente na base de dados que está ligada à *framework,* o que possibilita a execução de todo o processo ETL. Ou seja, o utilizador, ao parametrizar o alerta, tem de construir uma *query* específica para cada alerta necessário de deteção e identificar todas as diferentes parametrizações que, na sua maioria, são transversais a todos os alertas.

No final do processo de ETL, o resultado é carregado em ficheiros do tipo parquet[1] disponibilizados em diferentes pastas, tendo em conta os diferentes alertas. Com base nestes ficheiros gerados pelo processo ETL é criado o *dataset* utilizado para as análises de alertas realizadas no Power BI.

A Figura 9 representa a arquitetura da solução previamente desenvolvida com os diferentes componentes referidos.

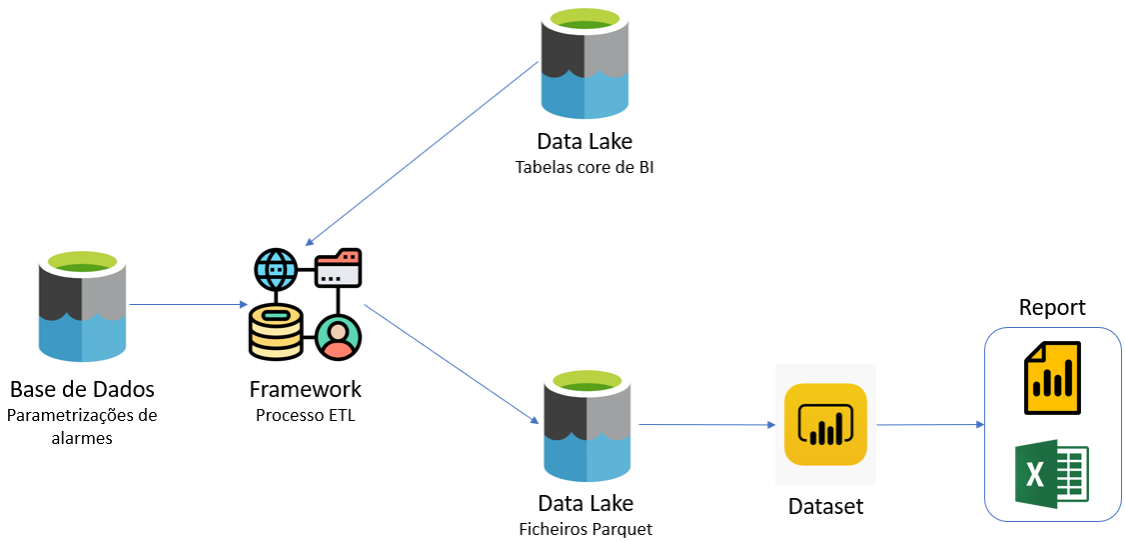


Figura 9 – Arquitetura da Solução prévia.

Nas seguintes secções são abordados cada um dos componentes que constituem o a solução previamente desenvolvida.

3.2 Parametrizações

As parametrizações são realizadas na tabela de configurações do processo ETL, que se encontra numa base de dados SQL Server. Cada configuração é designada de metadado, que consiste num conjunto de informações resumidas que permitem a execução do processo ETL.

[1] Ficheiro de dados, orientado a colunas, projetado para armazenamento de dados de forma eficiente e recuperável. Fornece esquemas de compactação e codificação de dados com desempenho melhorado para operar com dados complexos e em massa [48].

Esta tabela de configurações é transversal para todos os processos, no entanto, na Tabela 4, são apenas descritos os campos utilizados na parametrização dos alarmes.

Tabela 4 – Tabela de campos de parametrização de regras da solução prévia

Campo	**Descrição**
ETLMetadataId	Identificador do metadado.
PackageType	Tipo de processo a ser realizado. No caso dos alarmes é um processo de carregamento de dados.
Category	Categoria do Metadado. No caso dos alarmes é considerada a categoria de validações.
FlowGroup	Ordem de execução dos diferentes processos.
SourceConnectionId	Identificativo da ligação utilizada para a extração do processo ETL.
SourceObject	Query de extração de dados.
Strategy	Tipo de carregamento de dados. No caso dos alarmes é de inserção apenas.
DestinationConnectionId	Identificativo da ligação utilizada para carregamento do processo ETL.
DestinationTable	Tabela de destino do carregamento do processo ETL.
ControlField	Campo de controlo do metadado.
MetadataGroup	Grupo de metadados a que pertence.
ExecutionEnabled	Execução do metadado ativa ou inativa.
Description	Descrição do metadado.

3.3 Framework

A *framework* desenvolvida em *Azure Data Factory* (ADF), permite realizar todo o processo ETL tendo em conta a configuração de metadados realizada numa base de dados SQL Server específica do motor de ETL.

Os diferentes metadados podem ter uma execução automática ou manual, ambas configuradas através de *triggers* automáticos e manuais, respetivamente.

O processo de ETL desenvolvido nesta *framework* também efetua o registo de *logs* para auditoria, permitindo assim realizar análises aos processos, detetar casos de sobrecarga do sistema, entre outros,

O *workflow* desta *framework* é genérico, pontualmente podem ser necessários novos desenvolvimentos de diferentes dinâmicas no ADF ou novas fontes de dados, tendo em conta as especificidades dos processos.

A *framework* permite a extração de dados de diferentes fontes de dados, incluindo diferentes tipos de ficheiros e bases de dados. Quando é necessária realizar uma extração de uma fonte de dados que ainda não foi utilizada é necessário configurar a mesma, tal como o caso de novos metadados.

Apesar da *framework* permitir o carregamento de diferentes tipos de dados para diferentes destinos, no caso dos metadados de alertas é realizado o carregamento de dados em ficheiros parquet. Que, por sua vez, são a fontes de dados para o *dataset* utilizado nas análises.

3.4 Dataset

Tendo em conta que a solução previamente realizada apresenta o foco de analisar vendas e devoluções, no *dataset* associado a esta destacam-se os campos de cálculo de vendas e devoluções disponíveis. Na Figura 10 é apresentado o modelo de dados do *dataset*.

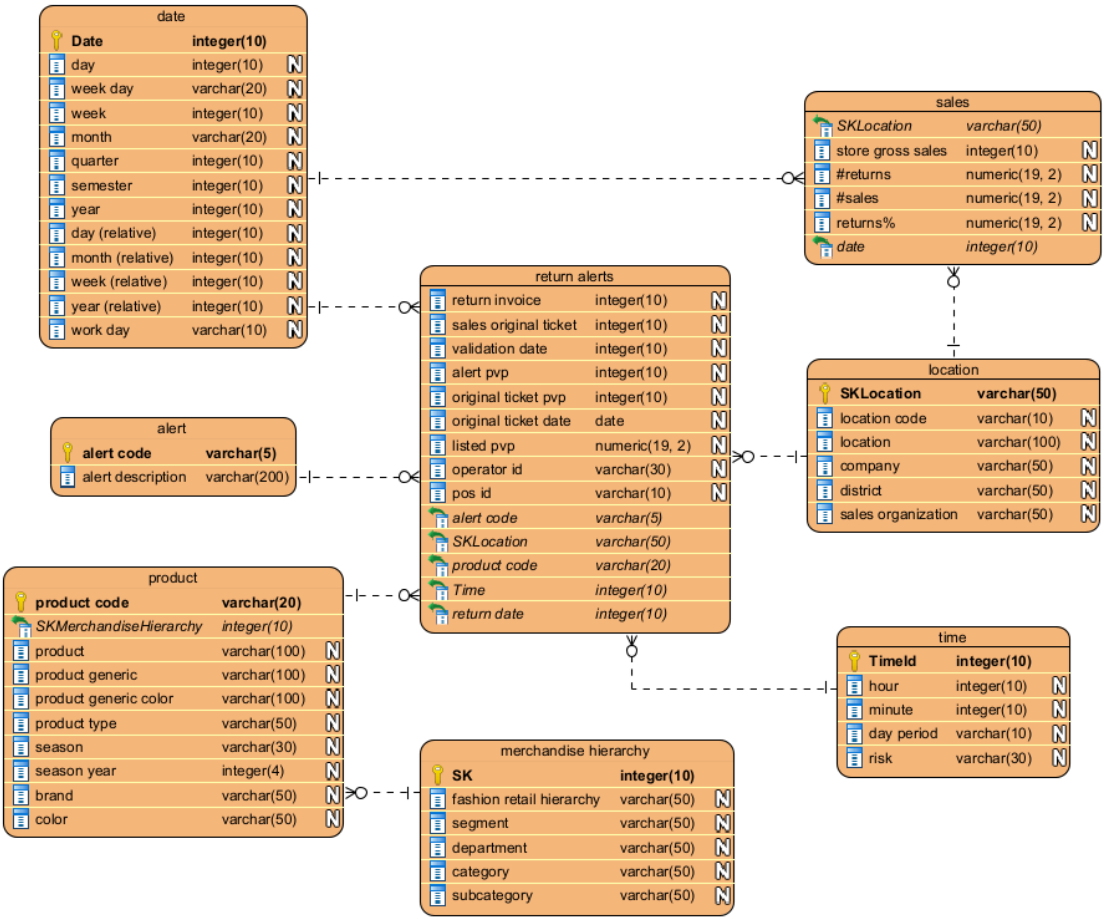


Figura 10 – Modelo de dados do *dataset* da solução prévia

Este *dataset* apresenta tabelas relativas a dimensões: Alert, Date, Location, Merchandise Hierarchy, Product e Time. Assim como, tabelas relativas a factos: Return Alerts e Sales.

- **Alert**

Esta tabela, mapeada manualmente e diretamente no Power BI, apresenta o código do alarme e a sua descrição. Tem de ser atualizada quando é acrescentado um novo alarme.

- **Date e Time**

Estas tabelas são criadas de forma transversal. Apresentam informações de data e tempo, utilizadas para análises temporais. Neste *dataset*, na dimensão de tempo, foi criado um campo extra que identifica o risco do alarme. Alarmes detetados fora do horário de funcionamento normal da loja apresentaram maior risco.

- **Location**

Tabela que apresenta informação de lojas.

- **Product e Merchandise Hierarchy**

Tabelas com informação relativa a produtos e hierarquia dos mesmos.

- **Returns alerts**

Esta tabela apresenta informação relativa aos alertas detetados, informação como valor de venda, valor de devolução, operador, dispositivo de registo e valores comparativos.

- **Sales**

Esta tabela apresenta dados de vendas agregados ao dia e loja. Tem o intuito de apresentar o total das vendas para analisar a quantidade de vendas realizadas e comparar as mesmas com a quantidade de alarmes detetados. As métricas disponíveis nesta tabela apresentam o valor bruto das vendas, a quantidade de vendas e de devoluções, assim como, a percentagem de devoluções realizadas em relação à quantidade de vendas.

3.5 Análises Power BI

Este projeto apresenta uma app de Power BI específica com acesso restrito. Para análise dos alertas despoletados, foi disponibilizado um *report* composto por 4 páginas:

3.5.1 Alert Overview

Nesta vista global são realizadas 4 análises distintas:

- Quantidade de alertas despoletados por zona de loja, onde se pode verificar quais as zonas onde são gerados mais alarmes.

- Quantidade de alertas despoletados por segmento do produto. Neste caso pode-se verificar se os produtos com maior número de alarmes são de mulher, homem, criança ou entre outros.

- Quantidade de alertas despoletados por loja, para detetar qual a loja com maior número de alarmes.

- Quantidade de alertas por tipo de alerta, para verificar os alertas que ocorrem mais frequentemente.

3.5.2 Alert Rates

Nesta vista está disponível uma tabela com a análise do valor de vendas total da loja relativamente ao valor de vendas que apresenta alarme, assim como a percentagem de devoluções realizadas. Na Figura 11 é apresentada a tabela com valores de testes.

sales district	location	alert code	doc date	alert over gross sales %	store gross sales	alert gross sales	# alarms	returns %
0	98	12	2021/06/16	199.8	10.00	-19.98	2	
		30	2021/06/01	28.9	44.97	-12.99	1	
			2021/06/02	447.5	4.02	-17.99	1	300.00
			2021/06/14	42.9	162.99	-69.95	6	20.00
1	88	30	2021/07/29	52.1	24.95	-12.99	1	
5	57	12	2021/06/15	-471.8	-13.98	-65.96	3	300.00
		13	2021/06/15	-42.8	-13.98	-5.98	2	300.00
			2021/06/30	958.8	5.00	-47.94	4	
		30	2021/06/15	-157.2	-13.98	-21.97	3	300.00
			2021/06/30	419.8	5.00	-20.99	2	

Figura 11 – Tabela de cotações de alarmes

Esta tabela permite agregar informação ao distrito de loja, loja, código de alerta e como detalhe final, a data do documento onde foi detetado o alerta.

3.5.3 Alert Details

Página que contém uma tabela com detalhe relativo ao alarme. Apresenta o código do alarme, data do documento de venda, data do documento de devolução, hora da devolução, loja, produto, identificador da fatura de devolução e de venda, valor de devolução e de venda, preço de venda ao público em sistema, identificador de operador que realizou a operação e dispositivo onde foi realizada a operação.

Esta página permite aos utilizadores do *report* realizar posteriormente, noutros sistemas da organização, uma análise mais exaustiva para determinar qual poderá ser o problema associado ao alerta obtido.

3.5.4 Alert Description

Esta página do *report* pretende mostra apenas informação relativa aos alertas, para isso é apresentada uma tabela que representa um glossário de alertas. Esta tabela contém o código e a descrição dos diferentes alarmes disponíveis para análise. Na Figura 12 é apresentada a tabela disponível nesta página.

alert code	alert description
13	Devoluções sem código de documento de vendas associado

Figura 12 – Tabela de descrição de alertas

4 Análise de Valor

A análise de valor é um processo aplicado a projetos, que tem como finalidade comparar se o produto requisitado pelo cliente dá resposta ao problema, tendo em conta o menor custo atendendo ao desempenho pretendido e à confiabilidade necessária [31].

Para realizar a análise de valor de um projeto podem ser consideradas diferentes técnicas. Na realização deste projeto foi considerado relevante usar o modelo de Desenvolvimento de Novo Conceito, Proposta de valor e Modelo *Quality Function Deployment* (QFD).

Após reunião com a professora doutora Susana Nicola, foi definido que não seria justificável usar os métodos *Analytic Hierarchy Process* (AHP) ou *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS), visto que ambos se referem a métodos de auxílio à tomada decisão e, no presente projeto, as tecnologias utilizadas já eram um requisito.

4.1 Modelo de Desenvolvimento de Novo Conceito

O modelo de desenvolvimento de novo conceito. do inglês *New Concept Development (NCD)*, trata-se de um processo iterativo não sequencial que permite uma gestão de projeto flexível e com capacidade de resposta a incerteza [32]. Este modelo consiste em cinco elementos distintos representados na Figura 13, são estes: identificação de oportunidade, análise de oportunidade, enriquecimento da ideia, seleção de ideias e, por fim, definição de conceito [33].

O modelo NCD surge de uma ideia para uma nova oportunidade de negócio, passando depois por todos os passos seguintes sem ter que seguir uma ordem fixa, até o conceito estar definido. O conceito já é a definição do projeto, tendo em conta as suas principais características e benefícios para o cliente [33].

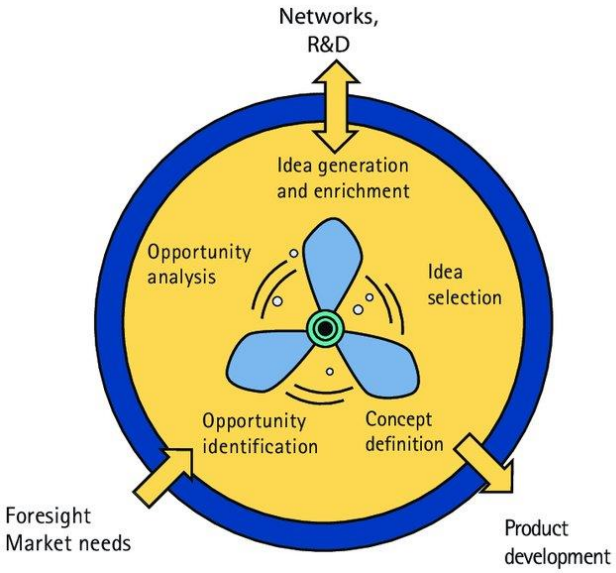


Figura 13 – Modelo de Desenvolvimento de Novo Conceito [33]

Nas seguintes secções são analisados os cinco elementos do modelo NDC aplicados ao presente projeto.

4.1.1 Identificação de oportunidade

Nesta etapa, tal como o nome indica, é onde a organização identifica oportunidades de negócio que podem apresentar algum propósito para a empresa. A oportunidade pode ser novo processo de negócio ou uma atualização de um produto existente [34].

A oportunidade do presente projeto surge com a necessidade de melhoria na validação de dados. Como já foi referido, o processo utilizado anteriormente não era prático na criação de novas regras de negócio e não permitia notificar utilizadores de anomalias.

4.1.2 Análise de Oportunidade

Após a identificação de oportunidades, é necessário analisar a informação adicional capaz de identificar oportunidades específicas para o negócio, verificar se a oportunidade traz valor ao

negócio. Neste passo, são realizadas análises de tendências e estudos de mercado [34]. Com o propósito de realizar análise de oportunidade, foi verificada a importância da qualidade de dados na organização.

Segundo estudos realizados, durante o processo de BI, quando ocorre a extração de dados, a taxa de erros de campo ronda os 5%, mesmo sendo utilizadas medidas de prevenção de erros. Foi, também, detetado que 40% dos dados extraídos são dados poluídos [35].

4.1.3 Geração e Enriquecimento da ideia

A ideia de um novo conceito pode passar por várias iterações e alterações quando interage com os diferentes elementos do modelo NCD. O desenvolvimento e seleção de ideias deve ser um processo contínuo na organização, para permitir inovação. Tarefas de gestão de topo devem promover iniciativas que incentivem a criatividade e inovação para suscitar a geração de novas ideias [33].

Na fase inicial do projeto foram debatidas várias ideias para um possível projeto, foram consideradas como principal foco as seguintes ideias:

- Desenvolver um mecanismo que permitia a validação de dados, com o intuito de detetar falhas que não são detetáveis através de uma análise agregada a dados;
- Criar um processo que permita detetar erros de forma rápida e clara para permitir realizar ações de correção eficazes;
- Ter um processo em que são definidas regras de negócio de forma simplificada.

4.1.4 Seleção da Ideia

Após serem definidas ideias de conceitos, devem ser selecionadas as ideias certas. Para isso deve haver um processo de seleção claro para a organização. Caso haja uma inovação radical a incerteza é alta e muitos aspetos da inovação podem ficar em aberto [33].

Tendo em conta que foi possível juntar as ideias consideradas mais relevantes indicadas na secção anterior, a solução apresentada tem como principal objetivo fornecer uma ferramenta

que será capaz de identificar anomalias e notificar as mesmas, assim permitirá a definição de uma ação rápida na resolução do problema.

4.1.5 Definição do Conceito

O último passo do modelo NCD é a definição de conceito. Este passo trata-se de uma definição que inclui os principais benefícios e características do projeto para o cliente, uma explicação da tecnologia a ser produzida e a forma como a organização irá beneficiar com o produto. Por vezes, antes do conceito ser formalmente definido, pode ser necessário reavaliar informação de previsão de mercado [33].

Tendo em conta as oportunidades detetadas e as ideias que trariam maior valor para o negócio, foi considerado o desenvolvimento de um motor de regras de negócio. Através de uma base com parametrizações de regras de negócio, o mecanismo gera um processo que deteta anomalias nos dados. Estas anomalias detetadas são disponibilizadas ao utilizador final através de análises em Power BI e/ou notificadas via email.

Esta ferramenta substitui uma ferramenta já existente na organização e pretende ser uma versão melhorada da mesma. A ferramenta utilizada anteriormente disponibilizava análises com anomalias detetadas, mas a parametrização de novas regras era realizada através de um processo técnico que necessitava de um novo desenvolvimento. O novo processo pretende facilitar a parametrização de novas regras de negócio, simplificando e diminuindo o tempo necessário na parametrização das mesmas.

Outra mais valia identificada no projeto desenvolvido, é o facto da nova ferramenta disponibilizar um sistema de notificação de anomalias via e-mail, algo que o processo anterior não disponibilizava.

Por vezes, podem ocorrer erros que não são detetáveis ou que são detetáveis tarde demais para se realizar alguma correção. Este projeto dota a organização de um processo que deteta falhas em tempo útil, para ocorrer uma ação de correção rápida e eficaz do problema. Claro está que para isso também é necessária uma definição eficaz de regras de negócio.

4.2 Proposta de Valor

A proposta de valor é definição de como um produto ou serviço representa valor perante um segmento de clientes específico. Deve descrever a maneira como a organização é diferenciada dos concorrentes, de forma a justificar a aquisição numa determinada a empresa e não numa empresa concorrente [36]. Para criar valor, na execução de um projeto é necessário compreender as necessidades de *stakeholders* e adaptar o projeto às mesmas. Com vista a alcançar este objetivo, é fundamental definir de forma clara o valor que é pretendido criar e definir mercado alvo, para possibilitar a definição de estratégias capazes de atingir este mercado.

Como auxílio na definição da proposta de valor associada ao presente projeto, foi definido o modelo Canvas e o Elevator Pitch relativos ao projeto.

4.2.1 Value Proposition Canvas

O modelo de Value Proposition Canvas apresenta dois lados, o lado de Segmento do Cliente e a Proposta de Valor. O lado do segmento do cliente é onde é definido o entendimento do cliente. O lado da Proposta de Valor é mapeado na forma de como é pretendido criar valor para dar resposta às necessidades do cliente [37].

O lado do cliente deve definir Customer Jobs, as tarefas que o cliente pretende completar com o produto; Pains, problemas do cliente que vão ser eliminados com a utilização do produto; e Gains, o benefício do cliente em utilizar o produto. O lado da Proposta de valor deve definir Products & Services, a lista de produtos ou serviços; Pain Relievers, elementos que vão remover as frustrações do cliente; e Gain creators, tudo o que seja novo ou que melhore a experiência do utilizador [37].

Através do *template* disponível em [38] foi realizado o *Value Proposition Canvas* do projeto disponível na Figura 14.

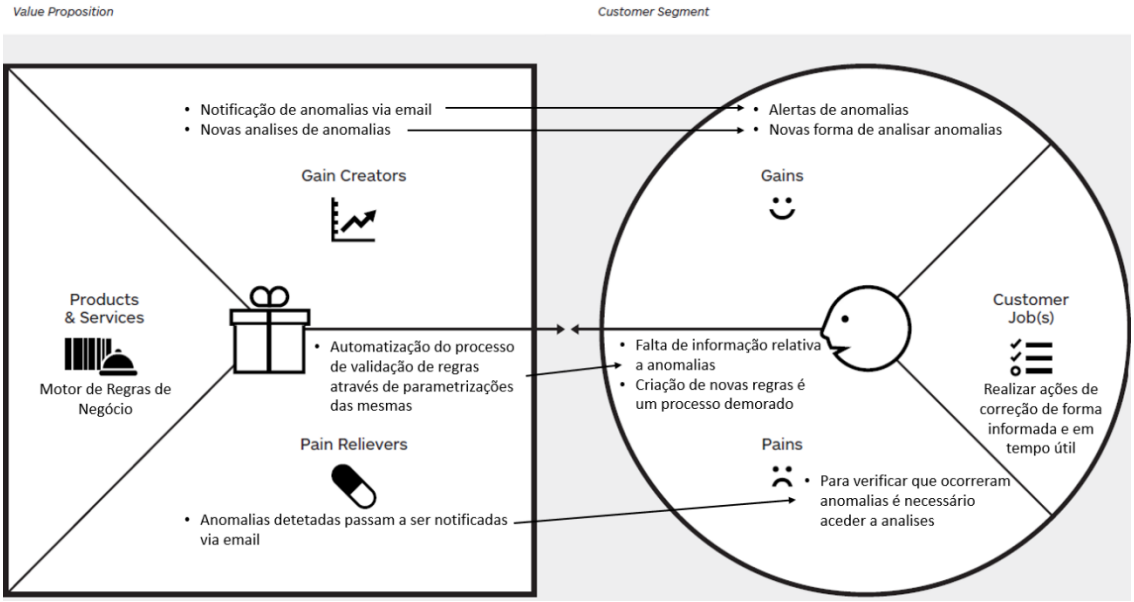


Figura 14 – Value Proposition Canvas do projeto

Foi definido como serviço o Motor de Regras de negócio que pretende dar resposta à necessidade que o cliente tem de realizar planos de ação de correção de forma informada e em tempo útil.

Este serviço tem como intuito dar resposta a queixas de falta de informação e demora no processo de acréscimo de novas regras, disponibilizando assim a automatização do processo de validação de regras através de parametrizações das mesmas. Foi também notado, que para verificar a ocorrência de uma anomalia é necessário aceder a análises. O projeto disponibiliza assim, a possibilidade de notificar em caso de anomalia se, ao realizar a parametrização da regra, for definido como tal.

Como ganhos, é disponibilizada a nova opção de notificação em caso de anomalia detetada e novas análises de anomalias que apresentem informação detalhada das mesmas.

4.2.2 Elevator Pitch

O *elevator pitch* consiste numa mensagem planeada e praticada que apresenta sucintamente a ideia empreendedora que se pretende apresentar [39].

O presente projeto pretende dotar equipas de análise de negócio e administradores de sistemas de uma ferramenta capaz de detetar e notificar falhas ou anomalias de dados. Com a finalidade, caso necessário, de estes utilizadores poderem proceder a um plano de ação na

correção da falha ou anomalia detetada. Tendo em conta esta descrição, foi definido o *elevator pitch* disponível na Tabela 5.

Tabela 5 – *Elevator Pitch* do projeto

For (*target customer*): Analistas de negócio e administradores de sistema **Who** (*statement of the need or opportunity*): Precisam de analisar dados de falhas ou executar correções **The** (*product or service*): Motor de Regras de Negócio **Is a** (*product or service category*): Um serviço de deteção de falhas **That** (*quantified statement of most compelling benefit provided to the customer*): Notifica no caso de ser detetada uma anomalia e disponibiliza análises de anomalias **Unlike** (competitors and competitive alternatives statement of primary differentiation of the product or service): Sistema utilizado anteriormente **Our** (*statement of primary differentiation of the product or service*): Facilita a parametrização de novas regras de negócio no caso de ser detetada essa necessidade e notifica via email os utilizadores no caso de ser detetada uma falha

4.3 Método QFD

O método *Quality Function Deployment* (QFD) é um processo que apresenta um conjunto de ferramentas para definir requisitos do cliente, tornando estes requisitos em especificações de engenharia e metas mensuráveis do projeto. A metodologia QFD disponibiliza um conjunto de matrizes para facilitar a progressão do projeto. [40].

A utilização deste método reduz a possibilidade de alteração tardia no projeto, o que se traduz num menor tempo de desenvolvimento e menor custo. É um método estruturado que regista as decisões tomadas, apresenta foco no cliente, garantindo assim que as necessidades do mesmo são atendidas. O método QFD, também permite realizar comparações com concorrentes, este tipo de análise no auxílio da tomada de decisão de design do projeto [40].

Com o intuito de traduzir os interesses e necessidades dos clientes em características de design é utilizada uma matriz de relacionamento designada *House of Quality*. Esta ferramenta trata-se de uma matriz no formato semelhante ao de uma casa, que revela a relação entre o que o cliente quer e os parâmetros do projeto [40].

A matriz *House of Quality* apresenta as seguintes secções [40]:

1. **Requisitos do cliente**: lista das necessidades do cliente. Também designada por WHATs;

2. **Fator de importância**: numa escala de 1 a 5 é representado o nível de importância de cada função para o cliente, tendo em conta que 5 representa o nível mais alto;

3. **Teto**: recursos de design e requisitos técnicos para alinhar o produto com as necessidades do cliente;

4. **Corpo**: nesta secção são classificados através de símbolos a relação entre os recursos e requisitos técnicos com os requisitos do cliente. Cada símbolo representa um valor numérico 0,1, 3 ou 9;

5. **Telhado**: indica como os requisitos interagem entre si, podem apresentar uma relação de interação positiva, uma relação de interação negativa ou nenhuma interação;

6. **Comparação de Concorrentes**: comparação de produto dos concorrentes em relação ao cumprimento dos requisitos do cliente;

7. **Importância Relativa**: resultado do cálculo do total das somas de cada coluna multiplicado pelo fator de importância. Este cálculo permite classificar cada requisito do cliente e determinar, assim, onde se deve alocar mais recursos;

8. **Fundações**: especificações técnicas dos requisitos do cliente.

No contexto do presente projeto, foram determinadas as principais necessidades de administradores de sistema e analistas de negócio no que diz respeito à utilização de um motor de regras de negócio.

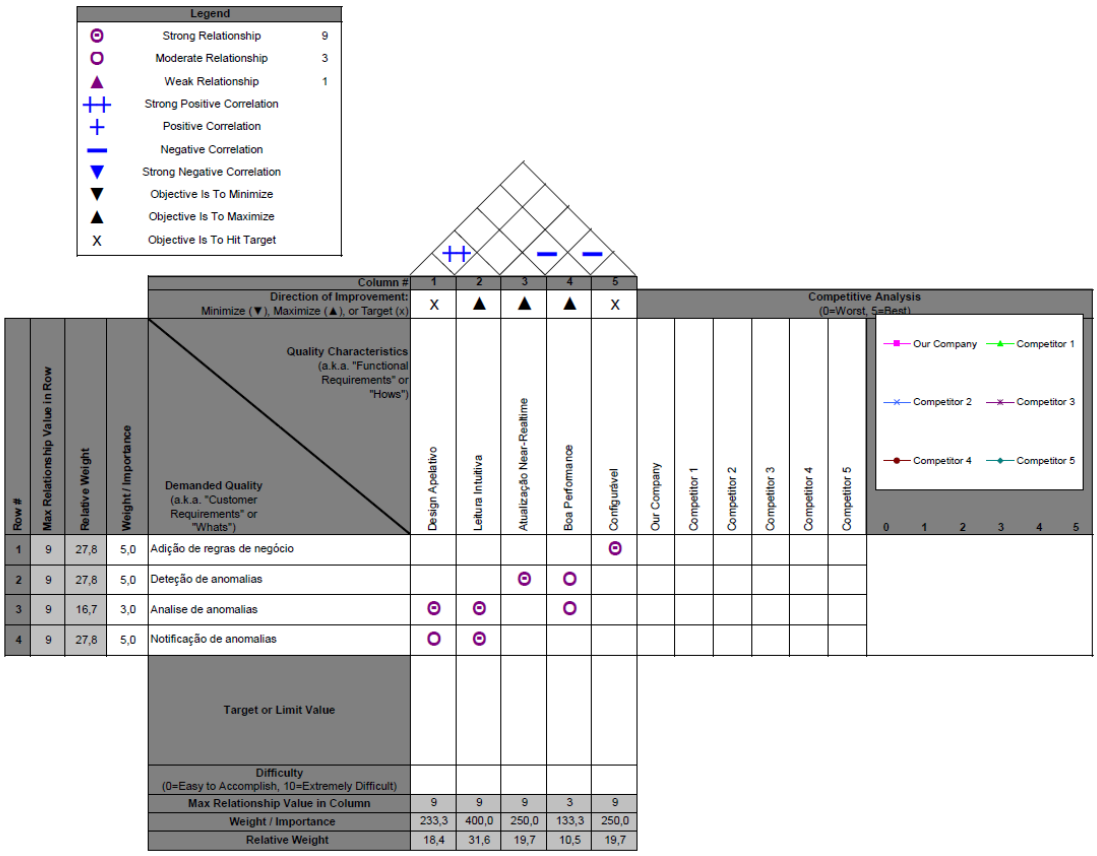
Foram identificados os seguintes requisitos de cliente:

- Adição de regras de negócio
- Deteção de anomalias
- Analises de anomalias
- Notificação de anomalias

Foram identificados os seguintes requisitos técnicos:

- Design apelativo
- Leitura intuitiva
- *Near-Realtime*
- Boa Performance
- Configurável

Tendo em conta as necessidades do cliente e os requisitos técnicos foi elaborado *House of Quality* representado na Figura 15.



Legend		
⊙	Strong Relationship	9
O	Moderate Relationship	3
▲	Weak Relationship	1
++	Strong Positive Correlation	
+	Positive Correlation	
−	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
X	Objective Is To Hit Target	

Column #					1	2	3	4	5	Competitive Analysis (0=Worst, 5=Best)					
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	X	▲	▲	▲	X						
Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Quality Characteristics (a.k.a. "Functional Requirements" or "Hows") / Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	Design Apelativo	Leitura Intuitiva	Atualização Near-Realtime	Boa Performance	Configurável	Our Company	Competitor 1	Competitor 2	Competitor 3	Competitor 4	Competitor 5
1	9	27,8	5,0	Adição de regras de negócio					⊙						
2	9	27,8	5,0	Deteção de anomalias			⊙	O							
3	9	16,7	3,0	Analise de anomalias	⊙	⊙		O							
4	9	27,8	5,0	Notificação de anomalias	O	⊙									
				Target or Limit Value											
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)											
				Max Relationship Value in Column	9	9	9	3	9						
				Weight / Importance	233,3	400,0	250,0	133,3	250,0						
				Relative Weight	18,4	31,6	19,7	10,5	19,7						

Figura 15 – *House of Quality* aplicada ao projeto

5 Análise

No presente capítulo é realizada uma análise do projeto. Através da engenharia de requisitos, foram definidas Partes interessadas, Requisitos funcionais e Requisitos não funcionais do projeto.

5.1 Partes Interessadas

Partes interessadas, ou *stakeholders*, são entidades que influenciam e podem ser a afetadas pelo sucesso de um projeto, de forma direta ou indireta [41].

Durante a análise deste projeto e dada a relevância dos *stakeholders*, foram identificadas entidades que influenciaram o desenvolvimento do sistema analítico criado. Foram identificados os seguintes *stakeholders*:

- **Administração** – entidade que tira proveito do serviço oferecido pelo sistema;
- **Gestores de Negócio** – determinam a relevância do mecanismo de regras criado;
- **Administrador de Sistemas Analíticos** – entidade que garante o bom funcionamento dos Sistemas Analíticos da Organização;
- **Analistas de Negócio** – entidade que utiliza o sistema;
- **Administradores de processos** – entidade que utiliza o sistema;
- **Administrador do sistema** – entidade que gere o sistema.

5.2 Identificação de Requisitos

O processo de engenharia de requisitos pretende descrever as necessidades do cliente, para isso, durante este processo, são analisados documentados e validados todos os requisitos e restrições do projeto [42].

Neste projeto, durante o processo de análise de requisitos, foram identificados requisitos funcionais e requisitos não funcionais.

5.2.1 Requisitos Funcionais

Requisitos funcionais são utilizados para descrever os serviços que o sistema deve disponibilizar e permitem especificar as funcionalidades necessárias para garantir estes serviços [41].

No caso de requisitos serem mal interpretados, a implementação não é a espectável pelo cliente. Isto leva à necessidade de estabelecer novamente requisitos, atrasos na entrega de um projeto, e, consequentemente, aumento de custos [41].

A especificação de requisitos funcionais deve ser completa, na medida em que todos os serviços referidos pelo cliente devem ser definidos de forma consistente, não podem haver requisitos contraditórios [41]. Durante a análise de requisitos funcionais foram identificados atores do sistema, casos de uso e análises a ser implementadas e abordadas nas seguintes secções.

5.2.1.1 Atores do sistema

Um ator do sistema representa o papel que uma entidade, humano ou outro sistema, desempenha enquanto interage com o sistema. Cada ator tem pelo menos uma função no sistema [42].

Neste projeto foram identificados quatro atores, são estes:

- **Analistas de Negócio** – este ator utiliza o sistema para analisar anomalias mais específicas, com o intuito de detetar fraude ou falhas no sistema;

- **Administradores de Processos** – este ator tem como principal interesse receber alarmísticas de falhas de integrações no sistema;
- **Administrador do sistema** – tem o papel de acrescentar regras ao sistema;
- **Sistema** – tem como função alertas no caso de ser detetada alguma anomalia.

5.2.1.2 Casos de Uso

Uma das formas mais usadas para representação de requisitos funcionais de um sistema é a representação de casos de uso. Neste tipo de representação, são identificados atores envolvidos na interação com o sistema e o tipo de interação que cada um deles tem com o sistema [41].

Na Figura 16 é representado o diagrama de casos de uso, onde são representados os casos de uso juntamente com as respetivas interações que o utilizador final pode ter com o sistema. Foram identificados sete casos de uso distintos no sistema.

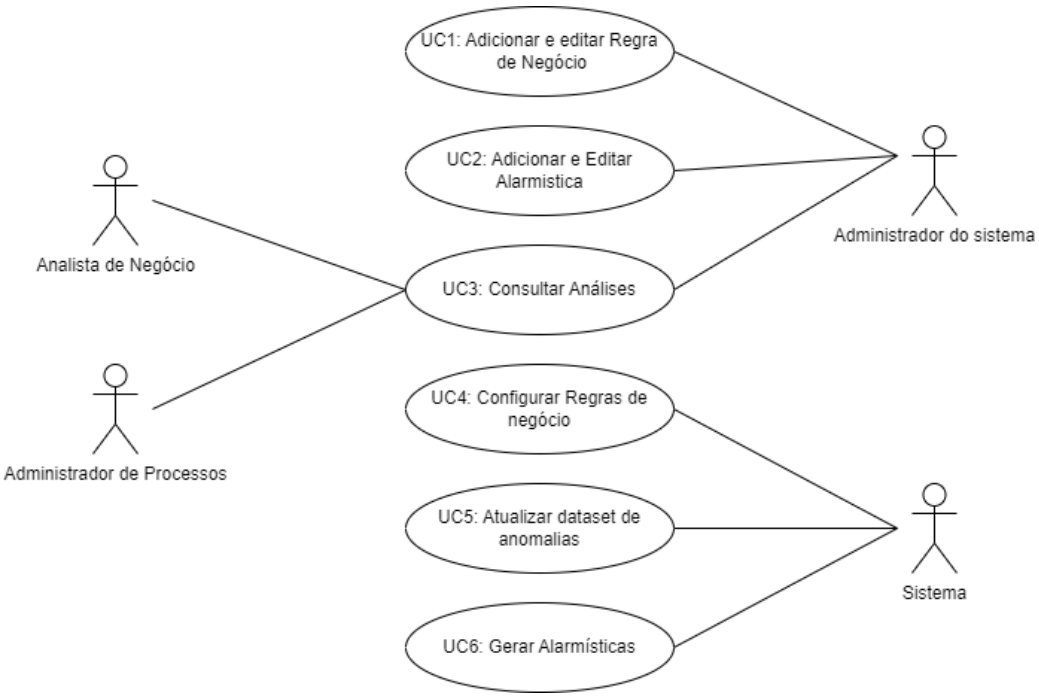


Figura 16 – Diagrama de Casos de Uso

As seguintes secções descrevem sucintamente casos de uso identificados no sistema e utilizadores que interagem com os mesmos.

- **UC1: Adicionar e Editar Regra de Negócio**

Este caso de uso é executado pelo Administrador do Sistema. O administrador de sistema deve adicionar ou editar as parametrizações de regras de negócio. O sistema guarda a atualização realizada na parametrização da regra.

Para representação do fluxo básico do caso de uso foi realizado o diagrama de sequência de sistema, disponibilizado na Figura 17. Neste diagrama estão representados os passos necessários para criar regra de negócio.

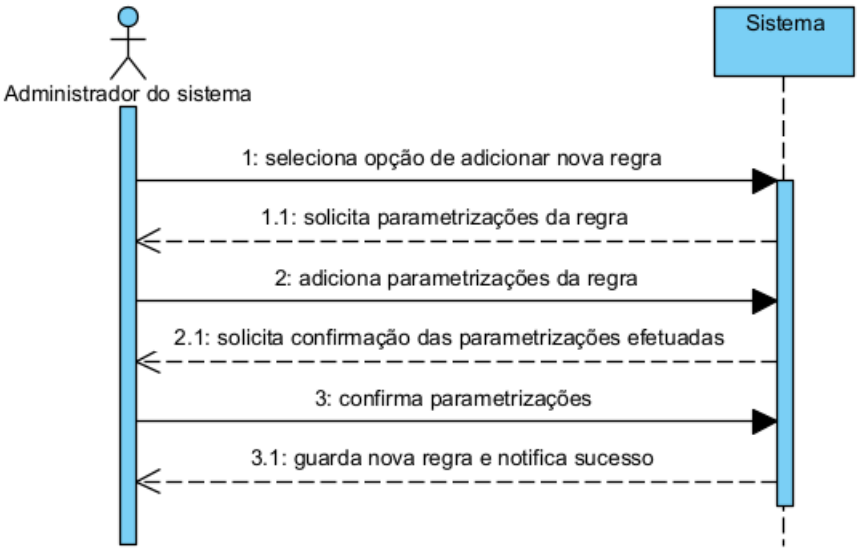


Figura 17 – Diagrama de Sequência de Sistema UC1: Adicionar regra de negócio

Na Figura 18 é representado o diagrama de sequência de sistema com os passos necessários para a edição de regra de negócio.

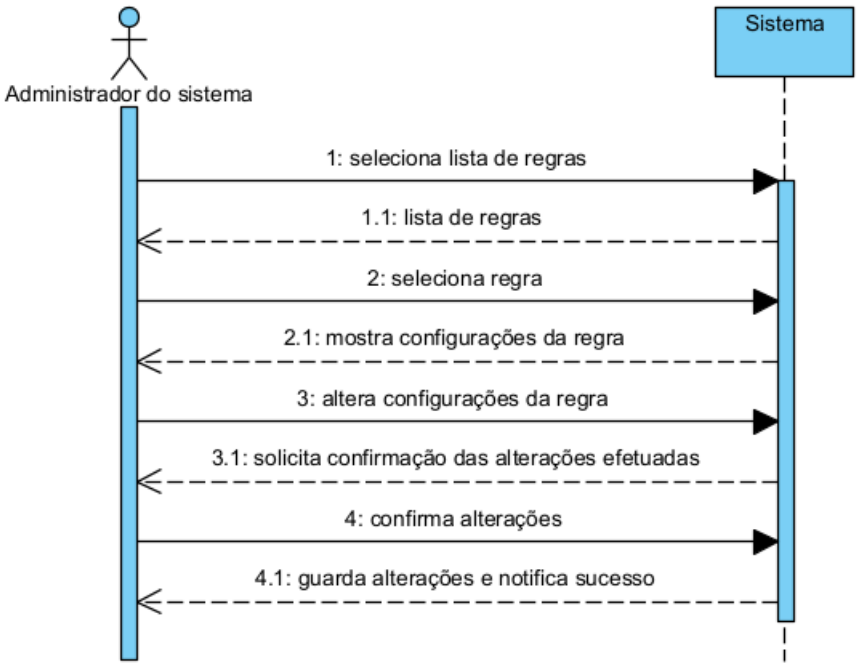


Figura 18 – Diagrama de Sequência de Sistema UC1: Editar regra de negócio

- **UC2: Adicionar e Editar Configurações de Alarmística**

Este caso de uso é executado pelo Administrador do Sistema. O administrador de sistema deve adicionar ou editar as parametrizações de alarmísticas. O sistema guarda a atualização realizada na parametrização de alarmísticas.

Para representação do fluxo básico do caso de uso foi realizado o diagrama de sequência de sistema, disponibilizado na Figura 19. Neste diagrama estão representados os passos necessários para criar alarmística.

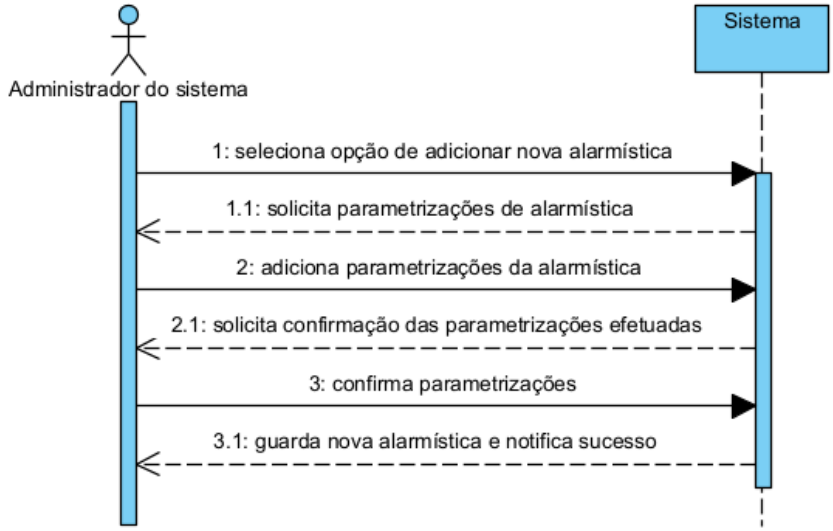


Figura 19 – Diagrama de Sequência de Sistema UC2: Adicionar configurações de alarmística

Na Figura 20 é representado o diagrama de sequência de sistema com os passos necessários para a edição de alarmística.

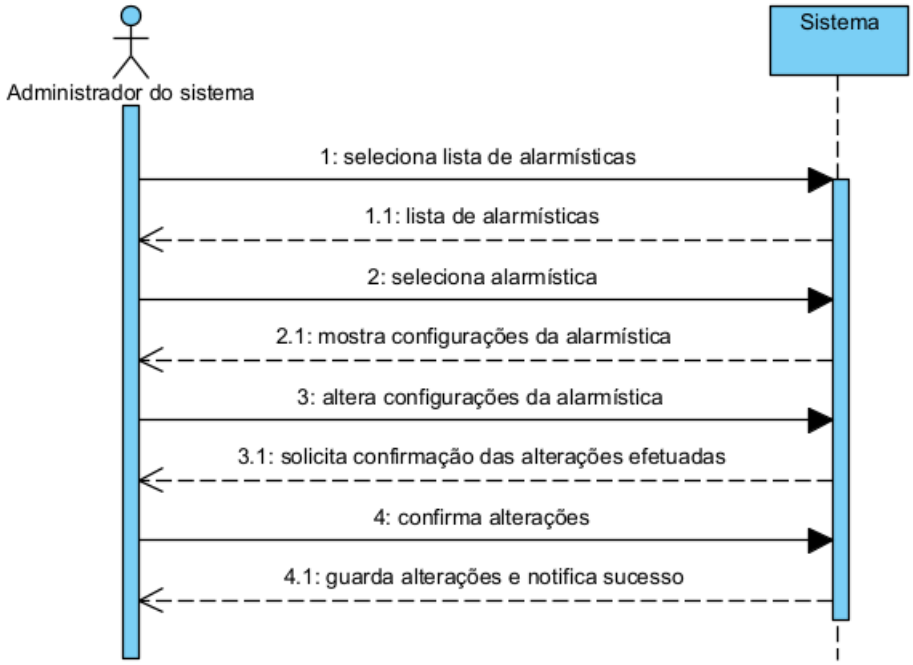


Figura 20 – Diagrama de Sequência de Sistema UC2: Editar configurações de alarmística

- **UC3: Consultar Análises**

Todos os utilizadores do sistema podem ter interação com este caso de uso. O utilizador seleciona da lista de análises disponíveis, a que quer verificar. O sistema mostra a análise que é pretendida verificar.

Para representação do fluxo básico do caso de uso foi realizado o diagrama de sequência de sistema, disponibilizado na Figura 21. Neste diagrama estão representados os passos necessários para consultar análises.

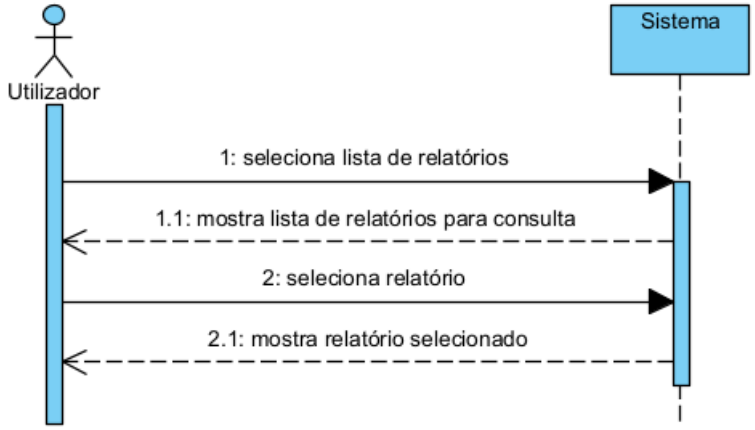


Figura 21 – Diagrama de Sequência de Sistema UC3: Consultar Análises

- **UC4: Configurar regras de negócio**

Este caso de uso é executado pelo sistema. Após a parametrização das regras, o sistema configura as regras de negócio como metadados para serem executados pela *Framework* ETL.

Para representação do fluxo básico do caso de uso foi realizado o diagrama de sequência de sistema, disponibilizado na Figura 22. Neste diagrama estão representados os passos necessários para configurar regras de negócio.

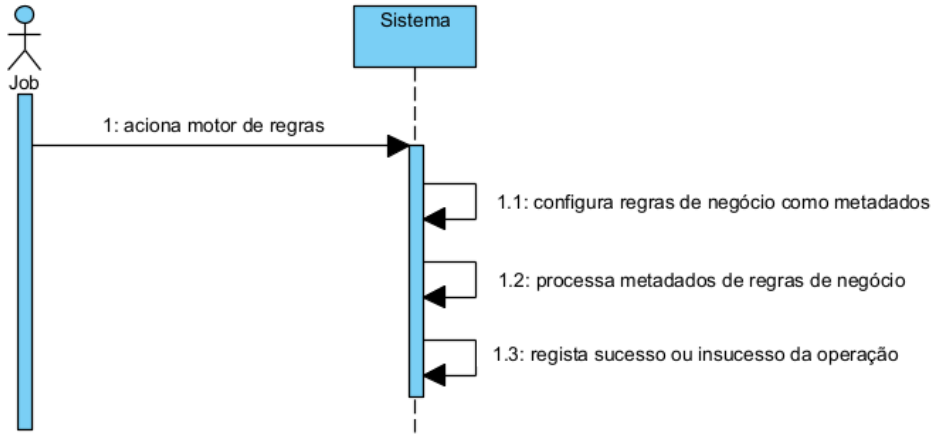


Figura 22 – Diagrama de Sequência de Sistema UC4: Configurar Regras de Negócio

- **UC5: Atualizar dataset de anomalias**

Este caso de uso é executado pelo sistema. Após o processo de validação de regras, o sistema atualiza a fonte de dados do *dataset* de anomalias.

Para representação do fluxo básico do caso de uso foi realizado o diagrama de sequência de sistema, disponibilizado na Figura 23. Neste diagrama estão representados os passos necessários para atualizar o *dataset* para análise de anomalias.

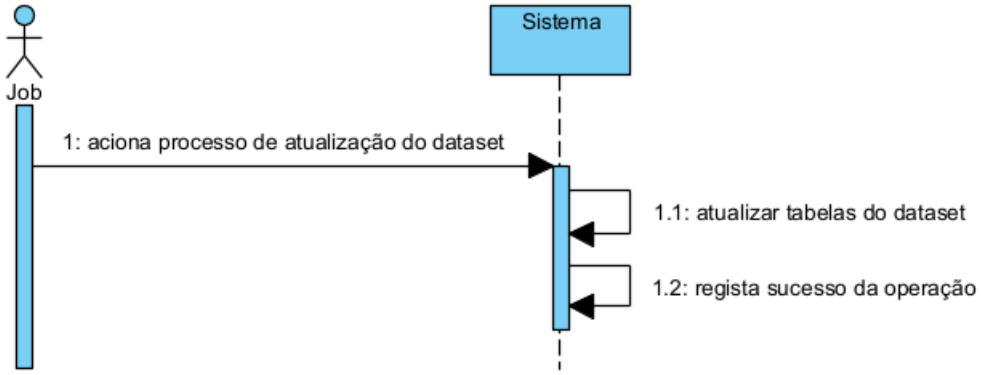


Figura 23 – Diagrama de Sequência de Sistema UC5: Atualizar o *dataset* de anomalias

- **UC6: Gerar Alarmísticas**

Este caso de uso é executado pelo sistema. Após o processo de validação de regras, o sistema notifica apenas novas anomalias no caso de serem detetadas.

Para representação do fluxo básico do caso de uso foi realizado o diagrama de sequência de sistema, disponibilizado na Figura 24. Neste diagrama estão representados os passos necessários para gerar alarmísticas.

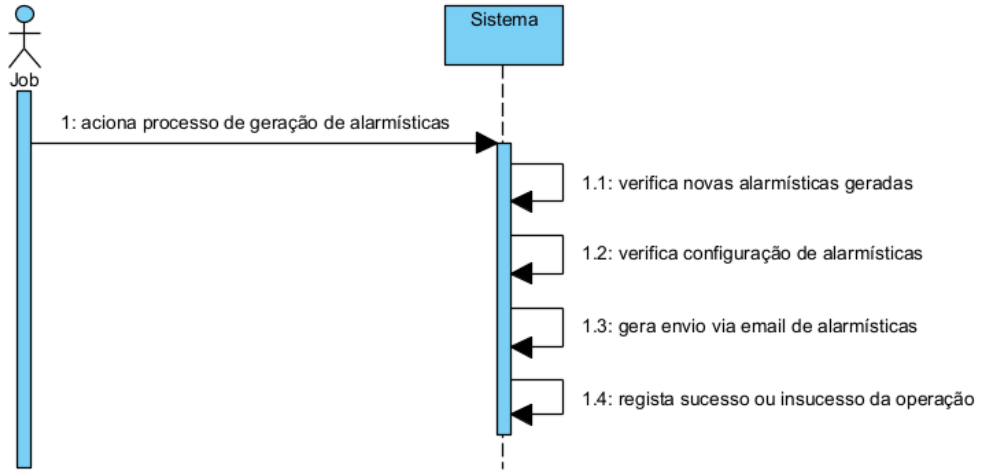


Figura 24 – Diagrama de Sequência de Sistema UC6: Gerar alarmísticas

5.2.1.3 Análises

Tendo em conta as necessidades apresentadas, foram definidas para implementação as seguintes análises:

- **Análise de Frequência de Anomalias:** análise que apresenta a frequência com que as regras não são cumpridas e regras que apresentam mais vezes falhas. Permite detetar regras que possam estar mal formuladas, detetar ações de melhoria que possam ocorrer para evitar anomalias mais frequentes.

- **Análise de anomalias:** análise que disponibilize detalhadamente anomalias detetadas. Fundamental para ajudar a detetar o porque da anomalia;

- **Análise de tendências e anomalias através de *insights* de Power BI:** recurso à funcionalidade de *insights* de Power BI para detetar e notificar de forma automática anomalias, tendências, entre outros padrões nos dados.

5.2.2 Requisitos não funcionais

Requisitos não funcionais são restrições que não são diretamente fornecidas pelo sistema aos utilizadores. Estes requisitos podem estar relacionados com as propriedades do sistema, restrições na implementação do sistema, entre outras características do sistema [41].

Nas seguintes subsecções são definidos e listados requisitos não funcionais tendo em conta a sua categoria do modelo FURPS+ [42].

5.2.2.1 Funcionalidade

Funcionalidade representa as principais funcionalidades do sistema que não são representadas através de casos de uso. Pode incluir aspetos como auditoria, interoperabilidade, compatibilidade e segurança [42]. Neste projeto foram identificados os seguintes requisitos de funcionalidade:

- Os *reports* devem estar disponíveis em Power BI Service;
- O sistema deve apresentar um mecanismo que despolete notificações, via email, de anomalias detetadas;
- As análises de anomalias devem ser atualizadas diariamente;
- Verificar apenas registos de anomalias novos, anomalias já detetadas não devem ser repetidas;
- Limitações de acesso a *reports*.

5.2.2.2 Usabilidade

Usabilidade representa requisitos referentes à interface de utilizador. São estes requisitos de acessibilidade, design de interface, documentação e padrões [42]. Neste projeto foram identificados os seguintes requisitos de usabilidade:

- *Reports* devem apresentar um design apelativo e de leitura intuitiva;
- Manuais de utilizador para os *reports* criados.

5.2.2.3 Desempenho

Desempenho representa requisitos de desempenho de software, são considerados aspetos como tempo de resposta do sistema, disponibilidade, consumo de memória e utilização da CPU [42]. Neste projeto foram identificados os seguintes requisitos de desempenho:

- *queries* realizadas à base de dados otimizadas em termos de performance;
- Integrações de dados otimizadas em termos de performance.

5.2.2.4 Suportabilidade

Suportabilidade corresponde a requisitos de suporte, são considerados aspetos como testabilidade, adaptabilidade, capacidade de manutenção, compatibilidade, configurabilidade, escalabilidade, entre outros [42]. Neste projeto foram identificados os seguintes requisitos de suportabilidade:

- As regras adicionadas no sistema devem ser validadas pelo administrador do sistema e adicionadas pelo mesmo;
- O Sistema deve permitir a passagem do processo para suporte com documentação e informação relativa ao sistema e despiste de possíveis falhas;
- O sistema deve ser configurável e genérico, numa primeira fase mais genérico, mas com a possibilidade de melhorar.

5.2.2.5 Restrições adicionais (+)

Restrições adicionais especificam restrições de design, implementação, interface e restrições físicas [42]. Neste projeto foram identificadas restrições de implementação.

4.2.3.6.2 Restrições de implementação

Restrições de implementação limitam a construção do sistema [42]. Neste projeto foram identificados os seguintes requisitos:

- O servidor da base de dados é Azure Data Lake o que implica a utilização de linguagem SQL para *querying* de dados, através de *databricks*;
- Todos os processos de ETL realizados através de Azure Data Factory;
- Devem ser utilizadas tecnologias Azure;
- Os *reports* devem ser disponibilizados em Power BI Service;
- Utilização de tecnologias Azure e Power BI.

6 Proposta de Solução

Neste capítulo foi abordada a proposta de solução pensada para resolução do problema identificado. Para isso foi idealizada a arquitetura da solução, selecionadas as tecnologias a utilizar durante a implementação do projeto, desenhados os processos do sistema através de diagramas de sequência, apresentado o BPMN, o modelo de relacional do *dataset* e, por fim, descrito o diagrama de componentes do sistema.

6.1 Arquitetura da Solução

A arquitetura da solução permite definir os principais elementos estruturais do software a ser desenvolvido e padrões de design a utilizar para dar resposta aos requisitos de restrições definidas [42].

Inicialmente, quando foi realizado o planeamento do projeto, foi pensada que a arquitetura do projeto passaria por retirar a *framework* ETL da solução prévia que, seria substituída por uma nova *framework* de Motor de Regras. Esta solução tinha uma arquitetura semelhante à solução prévia, apenas apresentava a substituição de uma *framework* por outra *framework* semelhante, mas com especificidades diferentes.

Na Figura 25 está representada a arquitetura pensada inicialmente, em que o motor de regras de negócio comunicaria com o SQL Server para obter as parametrizações das regras e o Data Lake, para extrair casos que não se encontravam coerentes com as regras. No final de execução, o Motor de Regras despoletava notificações dos alertas gerados via email e

guardava o resultado dos alarmes em ficheiros parquet. Por fim, o *dataset* é montado com base nos alertas guardados nos ficheiros parquet, para depois serem criados os *reports*.

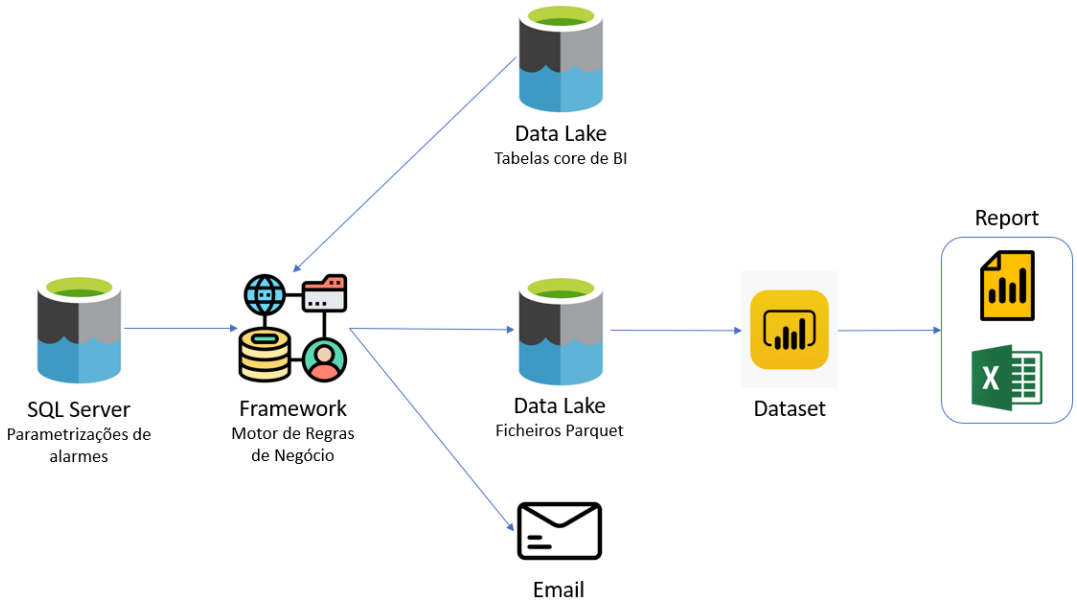


Figura 25 – Arquitetura inicial da solução

No entanto, visto que a Framework ETL apresentava bastantes especificidades e devido à complexidade de reproduzir um processo ETL que correspondesse a necessidades semelhantes à *framework* inicial, foi repensada a arquitetura proposta.

Apesar de ter sido tentada a implementação desta arquitetura inicialmente definida, chegou-se à conclusão de que a solução final poderia não apresentar benefícios significativos relativamente à solução prévia. Foi então repensada uma nova arquitetura para solucionar o problema, representada na Figura 26.

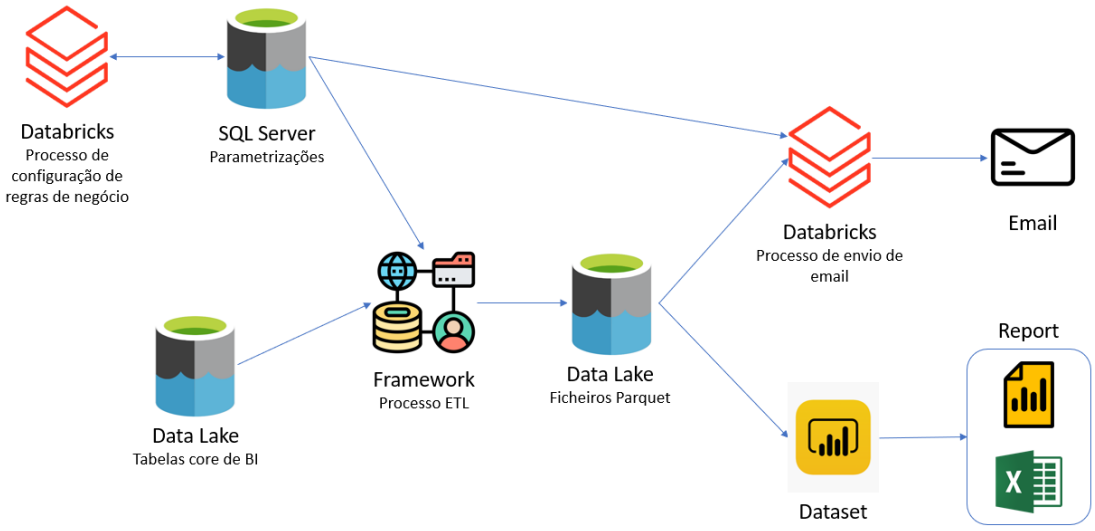


Figura 26 – Arquitetura final da solução

A arquitetura final da solução aproveita a Framework ETL já disponível e é implementado à parte, o processo de configuração de regras de negócio e o processo de envio de notificação.

O processo de configuração de regras lê as parametrizações de regras realizadas na base de dados SQL Server e converte as configurações de regras em configurações de metadados para serem executados pelo ETL *framework*.

Após execução da ETL *framework*, são gerados os ficheiros parquet que vão ser utilizados, tal como na versão inicial, como fonte do *dataset*, mas neste caso também são utilizados para processar o envio de notificação de alertas via email. O processo de envio de email lê as parametrizações de alarmísticas realizadas na base de dados SQL Server e envia email com os novos alertas detetados tendo em conta estas parametrizações.

6.2 Tecnologias Selecionadas

Tendo em conta o estudo de tecnologias realizado na secção 2.6 e o problema identificado na secção 1.2, pretende-se que o presente projeto, motor de regras de negócio, permita a adição de novas regras de uma forma rápida e intuitiva. De modo a permitir aos utilizadores receber alarmísticas em tempo útil e, com isto, conseguirem realizar um plano de ação eficaz de forma informada na resolução da anomalia.

Na Tabela 6 são identificadas as tecnologias que são pretendidas utilizar na realização do projeto, assim como a sua utilização.

Tabela 6 – Tecnologias a aplicar no projeto.

Desenvolvimento	**Designação**	**Utilização**
Motor de regras de negócio	Databricks	Processo de configuração de regras
	Python	Linguagem de programação utilizada para configurar regras
	SQL	Linguagem de programação para *querying* dos dados
Análise	Power BI	Representação de análises
	Tabular Editor	Modelo de dados para alterar o *dataset* de validações
	DAX	Linguagem de programação para fórmulas e *querying* de dados
Alarmística	Databricks	Processo de envio de email
	Python	Linguagem de programação para envio de email
	SQL	Linguagem de programação para *querying* dos dados
Outros	SourceTree	Controlo de versões de desenvolvimento do projeto

6.3 Processos

Considerando os desenvolvimentos idealizados na arquitetura final deste projeto pode-se destacar três processos distintos. São estes: o processo de configuração de regras, o processo de ETL e o processo de envio de alarmísticas. Estes processos são relevantes para o

cumprimento dos requisitos funcionais do projeto. Nas seguintes secções são descritos os fluxos dos processos de configuração de regras e de envio de alarmísticas. Como o processo de ETL já se encontrava desenvolvido pela organização não foi desenhado o fluxo do mesmo.

6.3.1 Processo de configuração de regras de negócio

Este processo prende-se com a necessidade de conversão de configurações de regras de negócio em configurações de metadados para posteriormente serem executadas pela ETL Framework.

A Figura 27 representa o fluxo executado para a configuração de nova regra de negócio. O processo inicia com um pedido à base de dados para obter as configurações das regras de negócio. Após obter a configuração das regras, o processo deve verificar, para cada regra, qual a dimensão a que está associada e configurá-la tendo em conta a sua dimensão. Por fim, adiciona o metadado à base de dados e guarda o log de sucesso do processo de configuração da regra. Após execução de todas as regras o processo é concluído.

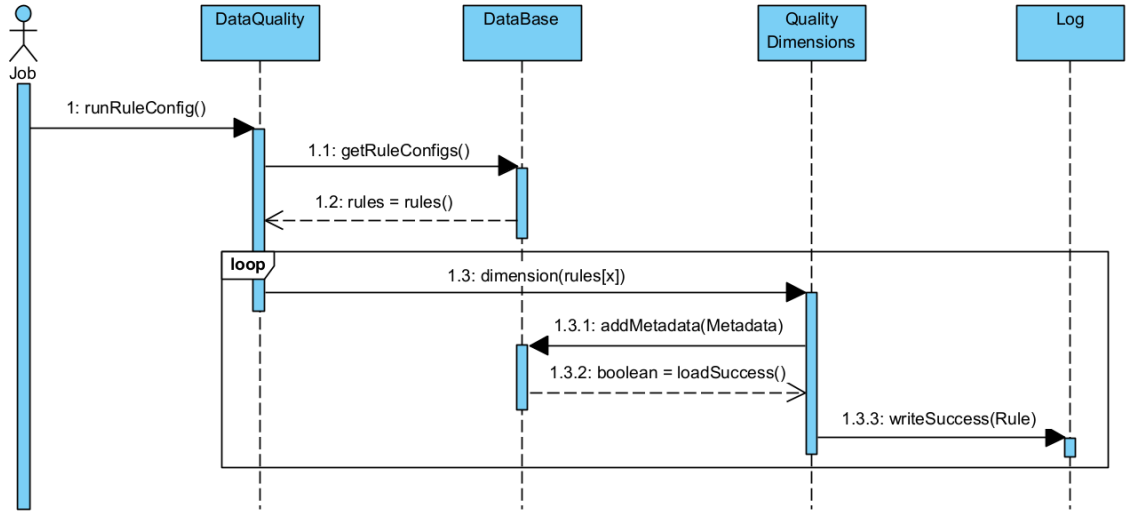


Figura 27 – Diagrama de sequência do motor de regras de negócio

Neste fluxo foi optado pela divisão das regras de negócio em dimensões de qualidade de dados, visto que cada dimensão apresenta as suas especificidades e, assim, é possível realizar validações distintas.

6.3.2 Processo de envio de alarmísticas

Este processo é executado para enviar aos utilizadores alertas de detenção de alarmes via email. Na Figura 28 foi idealizado o fluxo de envio de alarmísticas. Este processo inicia-se com dois pedidos realizados à base de dados. No primeiro pedido foram extraídas as configurações de alarmísticas, no segundo pedido são extraídas as datas dos últimos envios de alarmísticas.

Para cada alarmística o processo vai verificar se existem novos alarmes detetados. No caso de serem detetados novos alarmes, o processo envia email aos utilizadores parametrizados nas configurações, guarda log de email enviado e atualiza a data de última execução da alarmística na base de dados. No caso de não haver novos alarmes, o processo passa à execução da próxima alarmística. Este processo é concluído quando todas as alarmísticas são executadas.

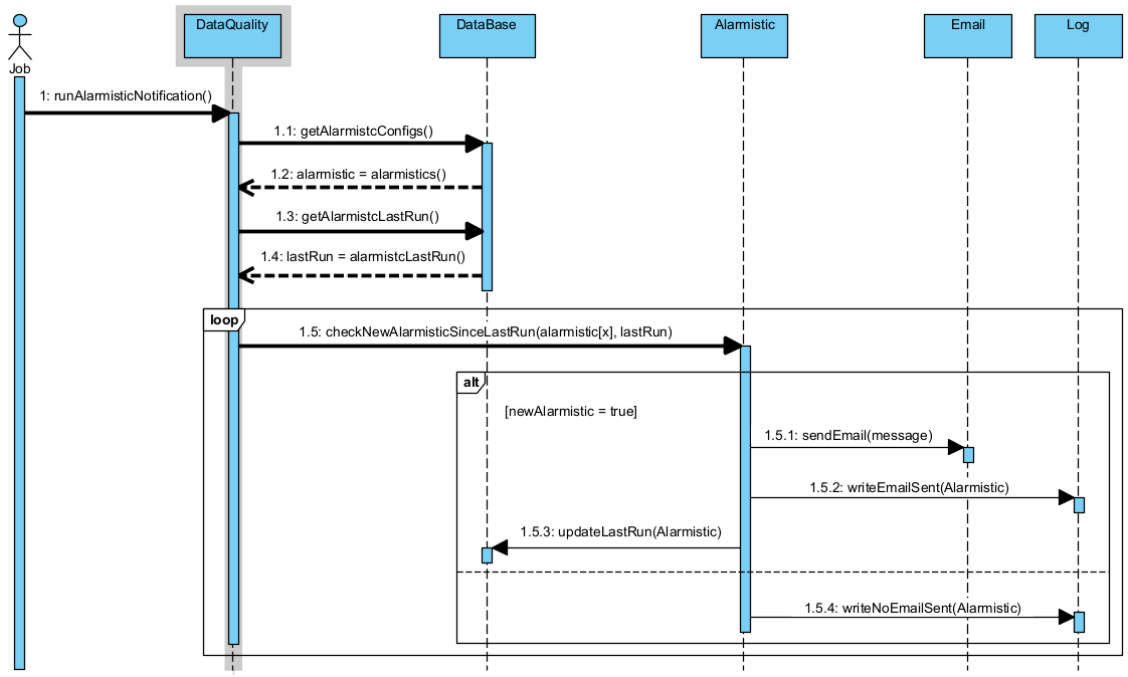


Figura 28 – Diagrama de sequência do envio de novas alarmísticas

6.4 Modelo relacional do dataset

Visto que a solução prévia já estava a ser utilizada pelo negócio, foi necessário projetar esta solução tendo em conta que seria necessário dar resposta às visualizações que já eram utilizadas. Daí o modelo relacional do *dataset* da solução proposta ser semelhante ao modelo da solução prévia.

Relativamente ao modelo utilizado previamente, foram alterados nomes de tabelas, uniformizados nomes de chaves primarias e estrangeiras de tabelas, criada tabela de regras com uma parametrização automática tendo em conta o resultado dos alertas gerados e acrescentadas novas métricas. Na Figura 29 está representado o modelo relacional do novo *dataset*.

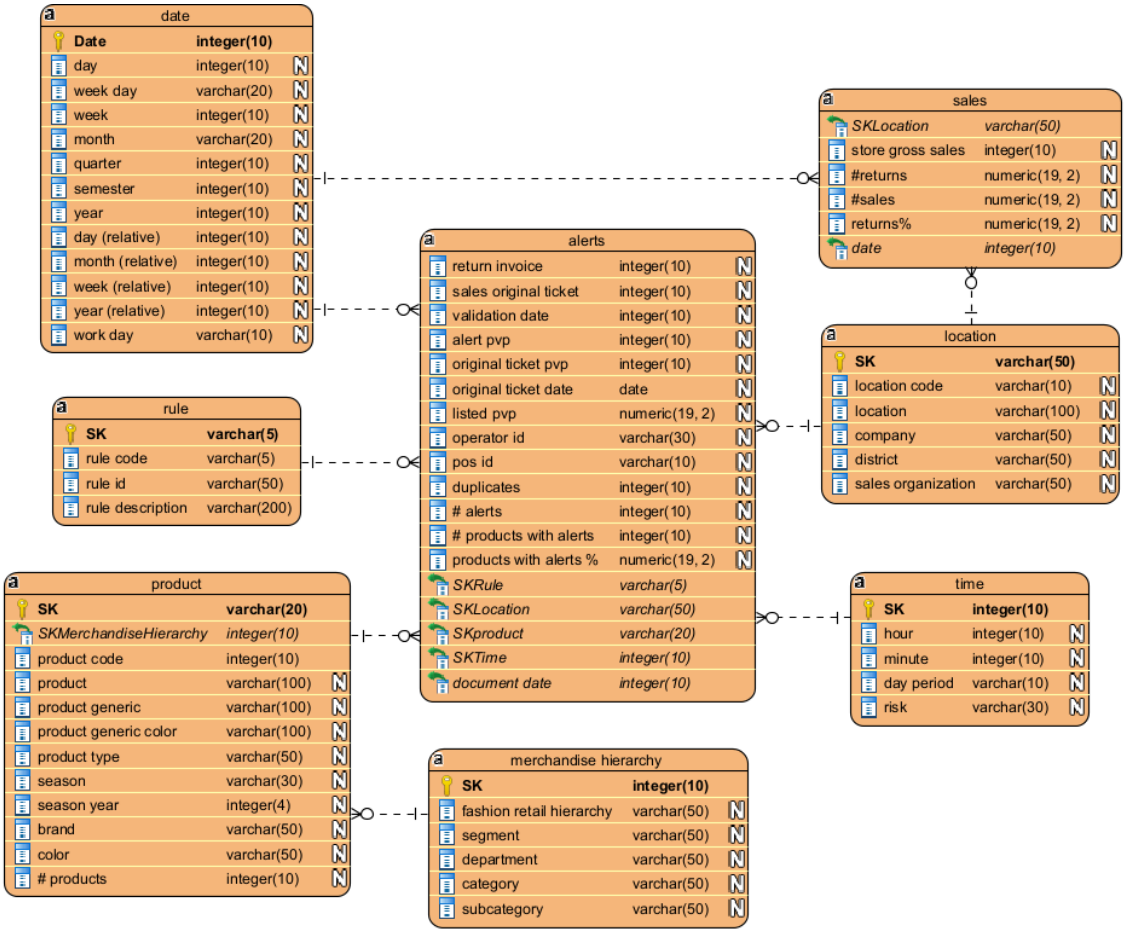


Figura 29 – Modelo de dados do *dataset* da solução

Na solução anterior, como as regras aplicadas apresentavam foco nas vendas e devoluções, não eram necessárias métricas relativas a produtos. Neste *dataset*, como foram acrescentadas regras relativas a campos de produtos, foi necessário desenvolver métricas que permitissem analisar alarmísticas de produtos despoletadas. Foram então criadas as seguintes métricas:

- **# alerts** – quantidade de alertas despoletados
- **# products with alerts** – quantidade de produtos distintos que apresentam alarme

- **products with alerts %** – quantidade de produtos distintos que apresentam alarme a dividir pela quantidade total de produtos

6.5 Diagrama de Componentes

Os diagramas de componentes têm o intuito de identificar os diferentes componentes do sistema, assim como as ligações entre os mesmos [43]. Este projeto é composto por 7 componentes. Na Figura 30 é representado o diagrama de componentes.

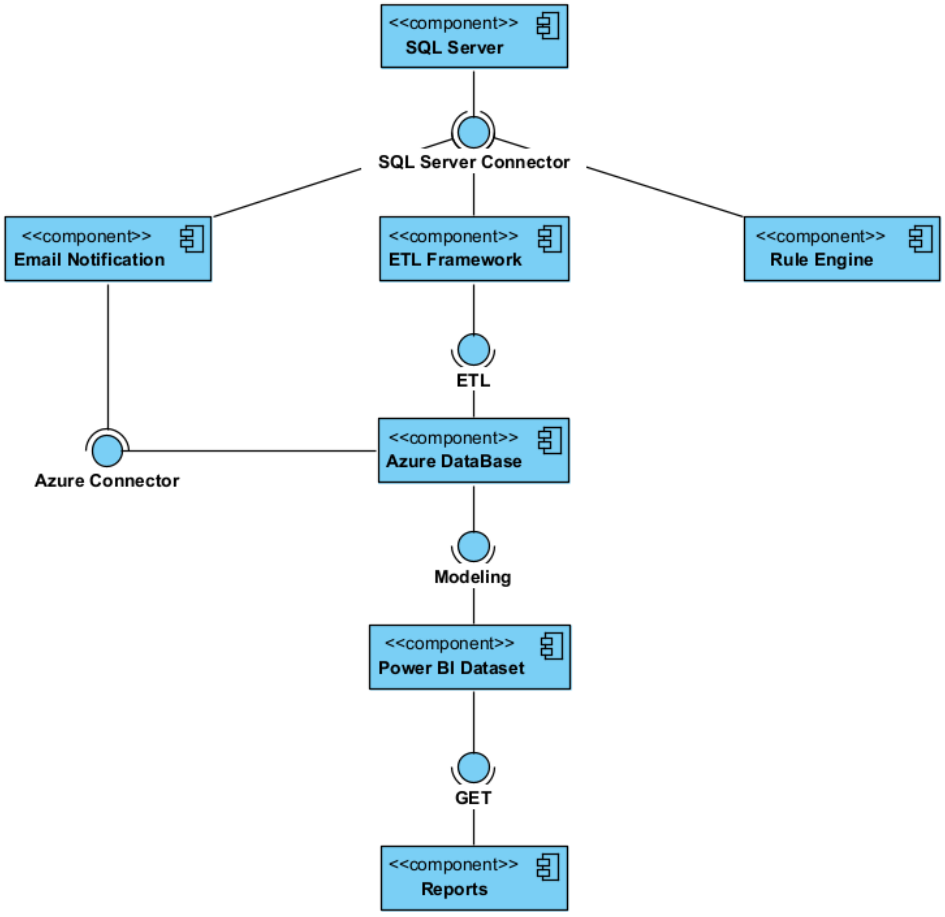


Figura 30 – Diagrama de Componentes

O presente projeto apresenta uma base de dados *SQL server* de utilização específica para configurações de processos. Esta base de dados, através de uma ligação *SQL Server Connector*, é utilizada pelo Motor de Regras de negócio e pela ETL Framework.

O Motor de Regras, após obter as configurações das regras disponíveis na base de dados SQL Server, realiza as operações necessárias para conseguir transformar as regras em metadados e

acrescenta os metadados à tabela de configurações dos mesmo que se encontra no SQL Server. Após guardar as novas configurações, o motor de regras vai despoletar o processo da ETL Framework.

Por sua vez, o ETL Framework vai aceder às configurações dos metadados e respetivos dados de extração para gerar ficheiros parquet que são guardados na base de dados Azure.

A notificação via email é realizada com base nas configurações de notificações presentes no SQL server, com base nos resultados de alertas obtidos que estão disponíveis nos ficheiros parquet.

Com o intuito de disponibilizar uma visão global dos alertas gerados via *report,* é utilizado um *dataset* cuja modelação é gerada a partir dos dados existentes nos ficheiros parquet e dados provenientes de tabelas tratadas em BI, são exemplos destas as tabelas de produto e de lojas, entre outras. Por fim, para desenvolver *reports* é realizada uma ligação ao *dataset* através do Power BI via pedido GET data.

7 Implementação

Neste capítulo são abordados todos os procedimentos efetuados para implementar a solução do problema. A implementação da solução foi dividida em 3 componentes distintos: Motor de Regras de Negócio, Alarmística e Análises.

7.1 Motor de Regras

O processo associado ao motor de regras foi desenvolvido na linguagem Python no Databricks, no entanto, o processo desenvolvido no Databricks foi complementado com informação de parametrizações e procedimentos provenientes da base de dados SQL Server.

No desenvolvimento do motor de regras, pode-se destacar 3 fases, são estas a criação da tabela de parametrizações onde são identificados os campos necessários para configurar as regras como metadados; o processo de conversão das regras em metadados, desenvolvido no Databricks; e o job que vai correr o *notebook* [2]de Databricks de forma automática.

7.1.1 Parametrizações

A tabela de parametrizações de regras foi criada na base de dados SQL Server, a mesma base de dados em que são configurados os metadados do ETL Framework. A tabela de

[2] Interface web para um documento que contem código executável, visualizações e texto narrativo [49]

parametrizações de regras possui o conjunto de campos descrito na Tabela 7. Estas parametrizações foram pensadas tendo em conta a necessidade de converter as regras em metadados.

Tabela 7 – Tabela de parametrizações de regras de negócio

Campo	**Descrição**
RuleId	Identificador da Regra
RuleCode	Código numérico da Regra
ValidationDimension	Dimensão de validação da regra.
Folder	Pasta onde deve ser guardado o ficheiro parquet com os alarmes detetados.
SourceConnectionId	Identificativo da ligação utilizada para validação da regra.
SourceTable	Tabela onde é realizada a validação
SourceResults	Colunas de resultados que devem ser apresentadas (separadas por virgulas)
SourceFilters	Filtros a realizar na tabela de validação
SourceJoin	No caso de ser realizado *join*, colunas de *join* (separadas por virgulas)
ComparisonTable	Tabela de comparação ou *join* dependendo da dimensão de validação selecionada
ComparisonColumns	Colunas de resultados que devem ser apresentadas (separadas por virgulas)
ComparisonFilters	Filtros a realizar na tabela de comparação.
ComparisonJoin	Colunas de *join* separadas por virgulas. Colunas neste campo têm de apresentar a ordem dos campos igual ao campo *SourceJoin*

Campo	Descrição
ManualQuery	*Query* manual da regra, utilizada no caso das parametrizações não conseguirem dar resposta às necessidades da regra
ControlField	Campo de controlo da Regra
MetadataGroup	Grupo de metadados a que a regra deve pertencer
ExacutionEnabled	Execução da Regra ativa ou inativa
Description	Descrição da Regra

De notar que para além da preocupação em conseguir converter estas configurações em metadados, foi também tido em conta as especificidades de cada validação a realizar. Por esse motivo, foi acrescentado o campo de *ValidationDimension* onde é definido o tipo de validação a realizar, tendo em conta as diferentes dimensões de qualidade de dados abordada na secção 2.4.1.

7.1.2 Processo de configuração de regras

O processo de configuração de regras foi desenvolvido no Databricks. Neste processo foram extraídas as parametrizações de regras disponíveis na base de dados SQL Server e reescritas na mesma base de dados, após convertidas em metadados.

O primeiro passo deste processo passou por realizar ligação à base de dados para extrair o *dataframe*[3] com as parametrizações de cada regra. Como cada regra apresenta as suas especificidades foi criado um ciclo para criar os metadados um a um.

Dentro deste ciclo, inicia-se o processo realizando uma validação do campo *ExecutionEnabled*, caso a regra se encontre com um estado de execução inativo, é realizado diretamente *update* da regra nas configurações de metadados para estado de execução inativo, passando assim

[3] Estrutura de dados que organiza os dados numa tabela bidimensional de linhas e colunas [50]

para a próxima regra. Caso a regra se encontre com o estado de execução ativo é realizada a verificação da dimensão da regra, representada na Figura 31.

Switch case for each ValidationDimension

```
def switch(validationDimension, row, metadata, destinationName):
  validationDimension = validationDimension.lower()
  if(validationDimension == 'completeness') :
    completenessDimension(row, metadata, destinationName)
  elif(validationDimension == 'consistency') :
    consistencyDimension(row, metadata, destinationName)
  elif(validationDimension == 'conformity') :
    conformityDimension(row, metadata, destinationName)
  elif(validationDimension == 'integrity') :
    integrityDimension(row, metadata, destinationName)
  elif(validationDimension == 'timeliness') :
    timelinessDimension(row, metadata, destinationName)
  elif(validationDimension == 'uniqueness') :
    uniquenessDimension(row, metadata, destinationName)
  elif(validationDimension == 'query') :
    queryDimension(row, metadata, destinationName)
  else :
    loadResultLog(row['RuleId'], "E", "Invalid dimension")
```

Figura 31 – Função de verificação do tipo de validação

Cada tipo de dimensão exige um desenvolvimento específico, por este motivo foram criados métodos diferentes para cada tipo de dimensão. Cada método utiliza os parâmetros da regra para montar uma *query* de extração para o metadado. A única semelhança nos métodos são os campos iniciais de *SELECT* que apresentam o código, nome e descrição da regra. Esta alteração permite obter, de forma automática, a informação da regra.

Foram então criados 7 métodos distintos para configurar as regras de negócio. Os seis métodos inicialmente desenvolvidos referenciam dimensões de qualidade de dados, mas, tendo em conta que poderia ser necessário realizar outro tipo de validações, foi criado outro método que permite configurar a regra logo em formato de *query*.

O método de *completeness* tem o intuito de criar uma *query* com a capacidade de validar se a informação se encontra disponível. Para isso, o processo realizado neste método vai ler a tabela onde deve ser realizada a validação, os campos de resultado e os filtros que devem ser realizados nesta tabela para identificar os campos que não se devem encontrar vazios. O resultado deste método é uma *query* básica com um filtro associado.

O método *consistency* cria uma *query* que verifica se duas tabelas distintas apresentam os mesmos valores. Para isso, neste método são criadas duas *subqueries* que numa *query* final são comparadas através de *join* dos dados.

O método *conformity*, apresenta um processo semelhante ao método de *completeness*, mas os filtros realizados nesta *query* validam a formatação dos dados. A construção desta *query* necessita da tabela onde é feita a validação, os campos de resultados e os filtros da formatação dos dados.

O método *integrity* cria uma *query* que valida a ligação entre duas tabelas. Para isso, o processo necessita de informação relativa a ambas as tabelas para comparação, campos de ligação entre as tabelas, campos de resultados e, no caso de ser aplicado, filtros. A *query* final apresenta um *join* de duas tabelas.

O método *timeliness* valida o campo de data da tabela a ser validada. Para isso, o método agrupa a tabela por um determinado conjunto de campos e filtra a data mínima que o campo deveria apresentar. Este método utiliza a tabela de validação, os campos de resultado e os filtros a aplicar.

O método *uniqueness* valida se um determinado conjunto de campos é chave da tabela. Este método agrupa os campos selecionados e realiza um contador para verificar se o número de campos é maior que um, representando assim duplicados. Para montar esta *query* o processo necessita da tabela de validação, campos de resultado e, caso seja necessário, filtros da tabela.

O método *query* considera apenas a parametrização disponível no campo *ManualQuery* para realizar a configuração do metadado. Este tipo de parametrização permite realizar uma *query* específica que permita dar resposta a necessidades de validação, que as outras opções não consigam corresponder.

Após a montagem do comando SQL realizada pelos diferentes métodos, foi chamado um procedimento na base de dados SQL Server para acrescentar a configuração do metadado. Este procedimento valida se o metadado já se encontra disponível na base de dados. Caso o metadado já se encontre disponível atualiza o registo do metadado, caso contrário acrescenta o novo registo.

Por fim, foram verificados os diferentes *MetadataGroup* configurados nas regras e, para cada um, é gerada a execução do ETL Framework. Como este processo de configuração de regras de negócio é construído no mesmo *workspace* de Databricks que a *framework,* é possível forçar a execução dos metadados configurados.

7.1.3 Logs

Durante o processamento de configurações de regras, de modo a facilitar a correção de algum erro ou de obter a informação da etapa em que o mesmo se encontra, foi acrescentado o registo de *logs*. Estes apresentam variadas mensagens que descrevem de forma clara o que está a ocorrer no processo. A Figura 32 mostra alguns casos de registos de *logs*.

Neste processo foi considerado relevante a separação de dois tipos de *logs*, são estes:

- ***Logs* do tipo I** – correspondem a mensagens de informação, podem ser registados em diferentes etapas do processo.

- ***Logs* do tipo E** – correspondem a mensagens de erro, podem ocorrer no caso de falhar algum campo fundamental na parametrização.

	RuleLogId	RuleLogDate	RuleId	LogLevel	Message
1	470	2022-10-13 10:05:38.5878227	VendasUnifoCar	I	Consistency Dimension loaded with success
2	469	2022-10-13 10:05:38.5878227	Rule_VendasUnifoCar	I	Category: Validations | FlowGroup: 25
3	468	2022-10-13 10:05:38.5721958	VendasUnifoCar	I	No comparison filters defined
4	467	2022-10-13 10:05:38.5721958	ProductsInSales	I	Integrity Dimension loaded with success
5	466	2022-10-13 10:05:38.5565718	Rule_ProductsInSales	I	Category: Validations | FlowGroup:
6	465	2022-10-13 10:05:38.5565718	ProductsInSales	I	No source filters defined
7	464	2022-10-13 10:05:38.5565718	ProductsInSales	I	No source filters defined
8	463	2022-10-13 10:05:38.5565718	ProductSeasonNull	I	Completeness Dimension loaded with success
9	462	2022-10-13 10:05:38.5409699	Rule_ProductSeasonNull	I	Category: Validations | FlowGroup: 25
10	461	2022-10-13 10:05:38.5254992	ProductSeasonFormat	I	Conformity Dimension loaded with success
11	460	2022-10-13 10:05:38.5254992	Rule_ProductSeasonFormat	I	Category: Validations | FlowGroup: 25
12	459	2022-10-13 10:05:38.5254992	ProductCodeUnique	I	Timeliness Dimension loaded with success

Figura 32 – Tabela de registo de *logs* do processo de configuração de regras

O registo do log é realizado através da chamada de um procedimento existente na base de dados SQL Server. Este procedimento vai receber os parâmetros de identificador da regra, tipo de log e mensagem para realizar uma inserção na tabela apresentada na Figura 32.

7.1.4 Job

Para realizar as validações das regras diariamente, foi necessário criar um Job de execução para o *notebook*. O Job é utilizado para executar, sem necessidade de interação, código desenvolvido em Databricks [44].

Foi então configurado o Job como se pode verificar na Figura 33. Para configurar o job foi-lhe atribuído nome, identificado o caminho do *notebook* onde é realizada a configuração das regras e atribuído um cluster que disponibiliza recursos necessários para a execução do *notebook*.

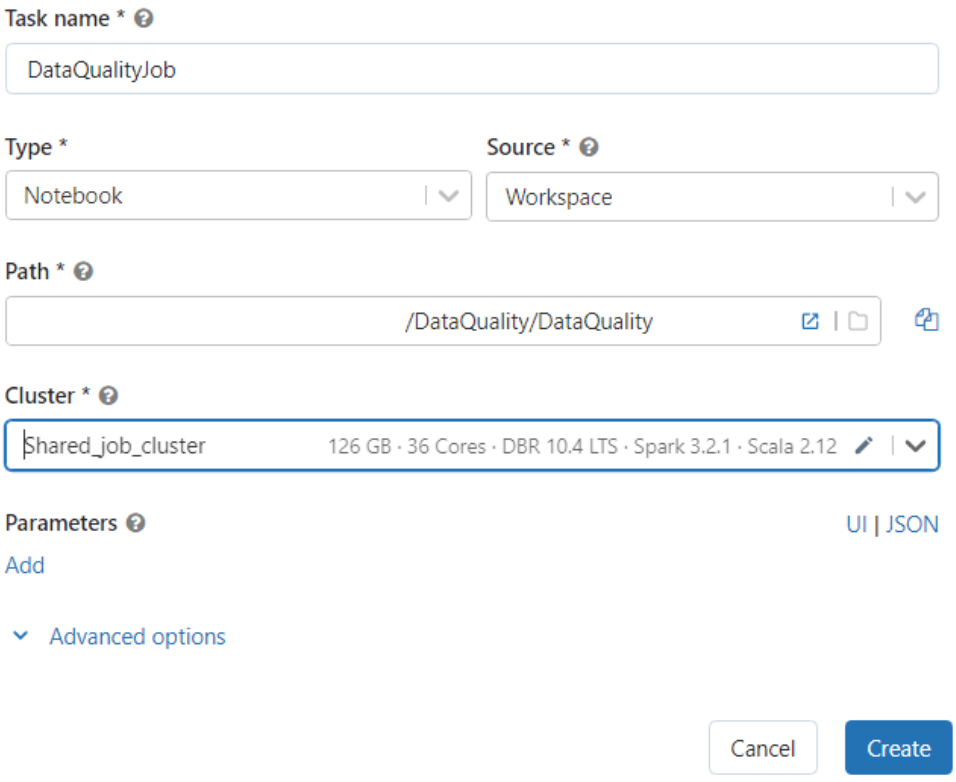


Figura 33 – Criação de Job no *databricks*

Após criação do job, foi atribuído um agendamento para execução do Job. Tendo em conta as necessidades do negócio ficou programado que o job é executado diariamente, a partir das dez horas.

7.2 Alarmística

Tal como o processo de configuração de regras de negócio, o processo associado à notificação de alarmísticas foi iniciado com desenvolvimento em Python no Databricks e complementado com informação de parametrizações e procedimentos provenientes da base de dados SQL Server.

Apesar deste processo não ter sido concluído, nesta secção é apresentada a parte do processo desenvolvida. No SQL Server foram criadas as tabelas de parametrizações da alarmística e de controlo de execuções das alarmísticas e o procedimento de atualização da tabela de controlo de execuções. No Databricks foram apenas realizadas ligações à base de dados para extração das tabelas de parametrizações e controlo do SQL Server.

7.2.1 Parametrizações

A tabela de parametrizações de alarmísticas foi criada na base de dados SQL Server, base de dados onde foram configuradas parametrizações do projeto. Esta tabela possui o conjunto de campos descrito na Tabela 8. Todos os campos de parametrizações, exceto a descrição, são obrigatórios na execução da alarmística.

Tabela 8 – Tabela de parametrizações de alarmísticas

Campo	Descrição
SubscriptionId	Identificador da alarmística
Name	Nome da Alarmística
RuleId	Identificador da Regra
SendTo	Destinatários do email
SendType	Tipo de envio de email (diário, semanal, mensal)
HourToSend	Hora de Envio
ExecutionEmabled	Execução da alarmística ativa ou inativa
Description	Descrição da alarmística

7.2.2 Controlo de execuções

Este controlo de execuções tem o intuito de guardar em sistema a última execução da alarmística para não ser realizado o envio de notificações repetidas. Caso contrário, os utilizadores estariam constantemente a receber notificações que poderiam já se encontrar resolvidas.

A tabela de controlo de execuções apresenta apenas dois campos que são o identificador da alarmística (*SubscriptionId*) e o campo de controlo (*ControlFieldValue*) que é a data da última execução da alarmística.

O campo de controle é atualizado sempre que é enviado email de notificação de alarme com a data e hora do sistema naquele momento. Para realizar esta atualização foi criado um procedimento que, no caso de já haver controle associado ao *SubscriptionId*, atualiza o campo *ControlFieldValue*, caso não haja um controle associado, cria uma linha de controlo.

7.3 Análises

Para análise dos alarmes despoletados pelo ETL Framework, foi criado um *dataset* no tabular editor e um conjunto de *reports* no Power BI desktop. Nesta sub-secção é abordada a implementação de cada um deles

7.3.1 Dataset Rule Engine

O *dataset* foi criado tendo em conta o *dataset* que já se encontrava disponível na solução prévia ao projeto, visto que o *dataset* já estava a ser utilizado pela organização e já fornecia resposta a necessidades do negócio.

De qualquer das formas, apesar de não ser necessário realizar grandes alterações, foi criado um *dataset* à parte do já existente, com o intuito de uniformizar nomes de chaves de tabelas e acrescentar novas tabelas.

Como o novo processo de alarmes permitiu disponibilizar novos campos, foi criada uma tabela com ligação aos ficheiros parquet localizados no data lake. Esta tabela trata os dados dos ficheiros e serve como base para outras tabelas. A Figura 34 mostra transformações realizadas na tabela (alguns nomes de campos são cortados por uma questão de proteção de dados).

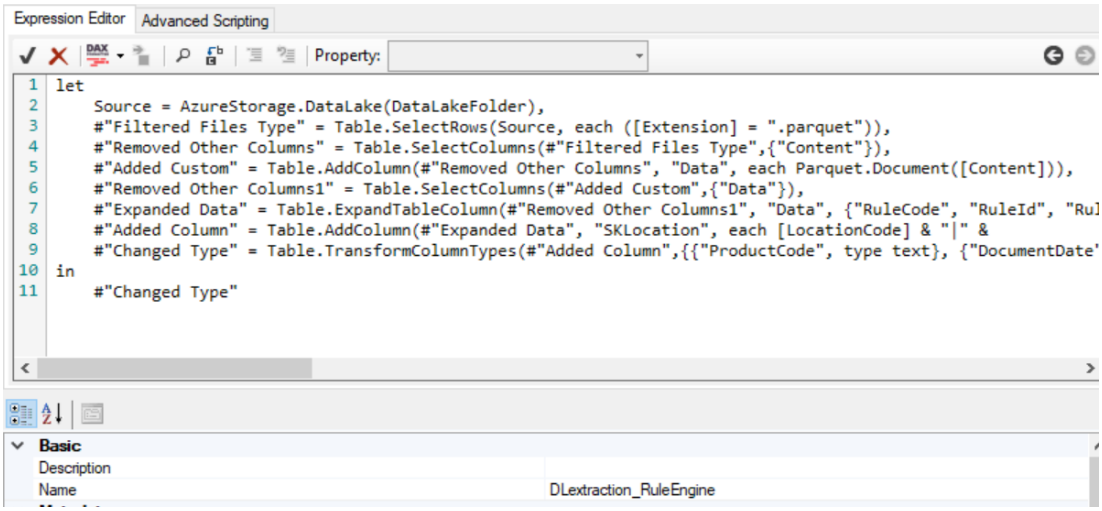


Figura 34 – Tabela de transformações de ficheiros parquet

Esta tabela serve como fonte para as tabelas *alert* e *rule*. A tabela de rule vai apresentar o nome, código e descrição das regras, sem ser necessária uma alteração manual como era feita

anteriormente. A tabela de *alert* vai apresentar campos dos alertas que foram definidos nas parametrizações em conjunto com os que já se encontravam disponíveis.

Foram, também, disponibilizadas novas métricas relativas a análises de produto, visto que foram realizadas validações de produtos para testar a solução desenvolvida. As métricas acrescentadas:

- **Products with alerts %**: percentagem de produtos que apresentam erros relativamente ao valor total de produtos
- **# produts with alerts**: número distinto de produtos com alerta

O *dataset* apresenta assim informação de produto e hierarquia mercadológica, de loja, de data e hora, de alertas, de vendas e regras. Na Figura 33 pode verificar-se a lista de tabelas disponíveis.

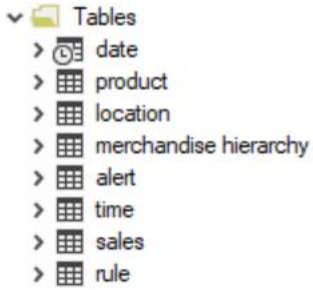


Figura 35 – Tabelas do *dataset* Rule Engine

A implementação do *dataset* ocorreu em paralelo com a realização do *report*, devido à necessidade de novas métricas.

7.3.2 Report

Tendo em conta que já havia uma solução prévia que estava a ser utilizada pelo negócio, foi necessário implementar os *reports* que já estavam a ser utilizados, com a ligação ao novo *dataset*. Nesta troca de ligação foram apresentados alguns erros devido à alteração realizada a nomes de tabelas no *dataset*, mas após a troca dos campos, o *report* ficou coerente com a solução já disponível.

Além das alterações realizadas aos *reports* que já se encontravam disponíveis, foi desenvolvido um *report* com análises a alertas gerados por validações de campos da

dimensão produto. Este *report* é composto por duas páginas, a primeira página disponibiliza um *overview* dos alertas despoletados enquanto a segunda apresenta uma vista detalhada das falhas detetadas na dimensão produto.

A primeira página do *report* apresenta uma visão geral das regras despoletadas com 3 análises distintas:

- um gráfico de barras com o número de alarmes despoletados por distrito.
- um gráfico de colunas com o número de alarmes despoletado por regra.
- uma tabela com detalhe de código e nome da regra, número total de alarmes despoletados, número de produtos distintos com alarme e percentagem de produtos com alarme tendo em conta a quantidade total de produto.

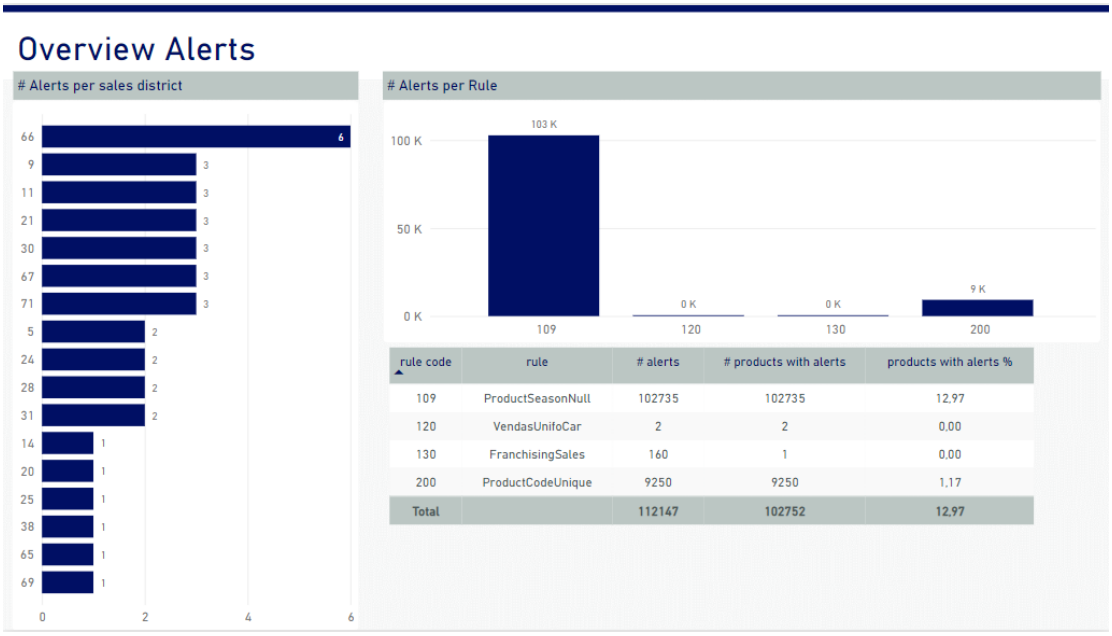


Figura 36 – Página de *overview* do report de alertas

A segunda página apresenta filtro de regra na lateral esquerda do *report* acompanhada por uma tabela com detalhe do código de regra, data de validação, código, designação e *season* do produto, valores duplicados (campo específico de validações realizadas a regras com dimensão unicidade de qualidade de dados) e métrica de número de alertas despoletados. Na Figura 37 é disponibilizada a página de detalhes de alertas de produtos com dados de teste.

rule code	validation date	product code	product	current season	duplicates	# alerts
109	2022-10-14	150	150 - CAR SEAT GR 0+ GREY			1
200	2022-10-14	150	150 - CAR SEAT GR 0+ GREY		2	1
109	2022-10-14	151006019	151006019 - PANTS JERSEY DRESS B, Dark Blue, 3/4			1
109	2022-10-14	151006021	151006021 - PANTS JERSEY DRESS B, Dark Blue, 4/5			1
109	2022-10-14	151006023	151006023 - PANTS JERSEY DRESS B, Dark Blue, 5/6			1
109	2022-10-14	151006026	151006026 - PANTS JERSEY DRESS B, Dark Blue, 6/7			1
109	2022-10-14	151006028	151006028 - PANTS JERSEY DRESS B, Dark Blue, 7/8			1
109	2022-10-14	151006031	151006031 - PANTS JERSEY DRESS B, Dark Blue, 8/9			1
109	2022-10-14	151006035	151006035 - PANTS JERSEY DRESS B, Dark Blue, 10,5			1
109	2022-10-14	151006038	151006038 - PANTS JERSEY DRESS B, Dark Blue, 11/14			1
109	2022-10-14	152	152 - BASE ISOFIX PLUS			1
109	2022-10-14	153	153 - BATHTUB WITH REDUCTO			1
200	2022-10-14	153	153 - BATHTUB WITH REDUCTO		2	1
109	2022-10-14	154006006	154006006 - PANTS DENIM DARK BLU, Dark Blue, 6/9M			1
109	2022-10-14	154006008	154006008 - PANTS DENIM DARK BLU, Dark Blue, 9/12M			1
109	2022-10-14	154006009	154006009 - PANTS DENIM DARK BLU, Dark Blue, 12/18M			1
109	2022-10-14	154006011	154006011 - PANTS DENIM DARK BLU, Dark Blue, 18/24M			1
Total						111985

Figura 37 – Página de detalhe de alertas de produtos do *report* de alertas

8 Avaliação e Experimentação

No presente capítulo foi abordada a forma como seria realizada a avaliação e experimentação da solução. Inicialmente, foram identificadas as grandezas associadas ao projeto e as hipóteses do problema.

Tendo em conta as hipóteses identificadas, são apresentados os indicadores de avaliação e as fontes de informação. Por fim, é apresentada a metodologia de avaliação de cada um dos indicadores identificados.

8.1 Grandezas

O presente projeto tem como principal foco a criação de um motor de regras de negócio que permita aos utilizadores detetar anomalias, tanto através de análises Power BI, como através de notificações via email. São então, identificadas como grandezas a exatidão nos dados apresentados, a satisfação dos utilizadores com a solução apresentada e o tempo de resposta do sistema.

Os valores identificados como anomalias devem ser exatos. Os utilizadores necessitam ter a informação correta para garantir o bom funcionamento dos sistemas e para identificar um plano de ação eficaz na resolução da anomalia detetada.

Tendo em conta que este projeto tem uma componente de apoio à decisão do utilizador, a satisfação do utilizador é fundamental para medir o sucesso do projeto. As análises realizadas devem ser claras e concisas e as notificações de anomalias devem ser esclarecedoras na identificação do problema.

Por fim, o sistema terá de apresentar as anomalias detetadas em tempo útil para ocorrer a resolução do problema, daí a necessidade de um sistema com um tempo de resposta rápido.

8.2 Hipótese

As hipóteses são proposições ou declarações preditivas específicas, devem ser definidas com clareza e ter a possibilidade de ser testadas. São definidas hipóteses tendo em conta a questões do problema que é analisado [45]. No presente projeto foram identificados dois tipos de hipóteses, hipótese alternativa e hipótese nula, tendo em conta as grandezas identificadas na secção anterior.

8.2.1 Hipótese Alternativa (H_1)

A hipótese alternativa representa as expectativas sobre o valor possível que o parâmetro hipotético pode assumir [45].

Hipótese Alternativa: A solução desenvolvida notifica e disponibiliza anomalias detetadas nos dados, através de um processo automatizado.

A hipótese apresentada é decomposta nos seguintes critérios:

- O sistema deteta anomalias nos dados;
- O sistema notifica via email no caso de anomalia;
- O sistema atualiza de forma automática análises de anomalias.
- O sistema apresenta análises de anomalias de forma clara para análise dos utilizadores.

8.2.2 Hipótese Nula (H_o)

Para confirmar ou não a hipótese de pesquisa é elaborada uma hipótese nula. Esta funciona como um complemento da hipótese alternativa em que o parâmetro hipotético é equiparado a um valor único e testado [45].

Hipótese Nula: A solução desenvolvida não notifica nem disponibiliza anomalias detetadas nos dados, através de um processo automatizado.

A hipótese apresentada é decomposta nos seguintes critérios:

- O sistema não deteta anomalias nos dados;
- O sistema não notifica via email no caso de anomalia;
- O sistema não atualiza de forma automática análises de anomalias.
- O sistema não apresenta análises de anomalias de forma clara para análise dos utilizadores.

8.3 Indicadores e Fontes de Informação

Após a definição de hipóteses, segue-se a identificação de indicadores de avaliação e fontes de informação capazes de avaliar os indicadores identificados.

8.3.1 Indicadores

Para verificar se as hipóteses definidas são cumpridas são definidos indicadores de avaliação de hipóteses. No presente projeto são considerados os seguintes indicadores:

- Cumprimento dos requisitos funcionais e não funcionais do sistema
- Análise de satisfação dos diferentes utilizadores do sistema

8.3.2 Fontes de Informação

Para avaliar os indicadores identificados na secção anterior, foram utilizadas, respetivamente, as seguintes fontes de informação:

- Testes de software – utilizados para avaliar a qualidade do software produzido e para evitar falhas no sistema implementado.

- Inquéritos de satisfação – permitem avaliar o grau de satisfação dos utilizadores e podem até mesmo identificar falhas e oportunidades de melhoria no sistema.

8.4 Metodologia de Avaliação

Com o intuito de verificar as hipóteses definidas e tendo em conta os indicadores e fontes de informação identificados foram utilizados dois tipos de metodologias de avaliação do projeto. São estes testes de software e inquéritos de satisfação aos utilizadores.

8.4.1 Testes de Software

Os testes de software possibilitam a deteção de erros que um programa possa apresentar e são fundamentais para avaliar a qualidade do software desenvolvido. Estes testes podem ser realizados durante ou após a construção do projeto [46]. Neste projeto, os testes de software são efetuados ao mecanismo de deteção de falhas. Serão realizados testes funcionais para verificar as funcionalidades implementadas no projeto, testes de integração para verificar as ligações entre os componentes do projeto, assim como testes de performance para verificar o tempo de processamento do sistema.

8.4.2 Inquéritos de Satisfação

Os inquéritos de satisfação são uma ferramenta que possibilita perceber se os utilizadores do sistema estão contentes com a solução proposta e, mediante o nível de satisfação, podem sempre realizar sugestões de melhoria para o sistema corresponder às suas necessidades.

Neste projeto, os inquéritos de satisfação foram realizados no fim da implementação do sistema, e as respostas aos mesmos serão avaliadas recorrendo a testes estatísticos.

8.5 Testes de Software

Para testar a solução desenvolvida foram realizados testes de funcionalidade, testes de integração e testes de performance. Nesta secção foram abordados os testes de software que acompanharam o desenvolvimento do projeto.

8.5.1 Testes funcionais

A verificação de testes funcionais foi realizada através da verificação de requisitos funcionais e especificações de *desing* do software [46]. Considerando os requisitos funcionais descritos na secção 5.2.1 foi avaliada a solução implementada.

A solução implementada dá resposta a maior parte das funcionalidades definidas na análise do projeto. A Tabela 9 lista os diferentes casos de uso definidos, se foram concluídos e testados.

Tabela 9 – Validação de casos de uso

Casos de Uso	**Conclusão**
UC1: Adicionar e Editar Regra de Negócio	Concluído e Testado
UC2: Adicionar e Editar Alarmísticas	Concluído e Testado
UC3: Consultar Analises	Concluído e Testado
UC4: Configurar Regras de Negócio	Concluído e Testado
UC5: Atualizar *Dataset* de Anomalias	Concluído e Testado
UC6: Gerar Alarmísticas	Não concluído

Podemos verificar que não foram concluídos todos os casos de uso propostos inicialmente na análise do projeto. No entanto nos casos de uso que não se encontram finalizados foi iniciada a sua implementação.

8.5.2 Testes de integração

A verificação de testes de integração é realizada quando dois ou mais módulos são integrados e testados em grupo [46]. Este tipo testes foram realizados durante a execução do motor de regras de negócio e na criação do *dataset*. No caso do motor de regras, foram realizados dois tipos de testes de integração distintos.

O primeiro teste realizado ao motor de regras de negócio, foi a ligação do *databricks*, onde foi implementada a configuração das regras, com a base de dados SQL Server, onde foram configuradas as parametrizações das regras. Neste teste foi realizada uma liga à base de dados de SQL Server, extraída através de um comando SQL a tabela de configurações e, por fim, realizado um display do resultado extraído da base de dados. A Figura 38 apresenta o teste de integração realizado à ligação entre o *databricks* com a base de dados SQL Server.

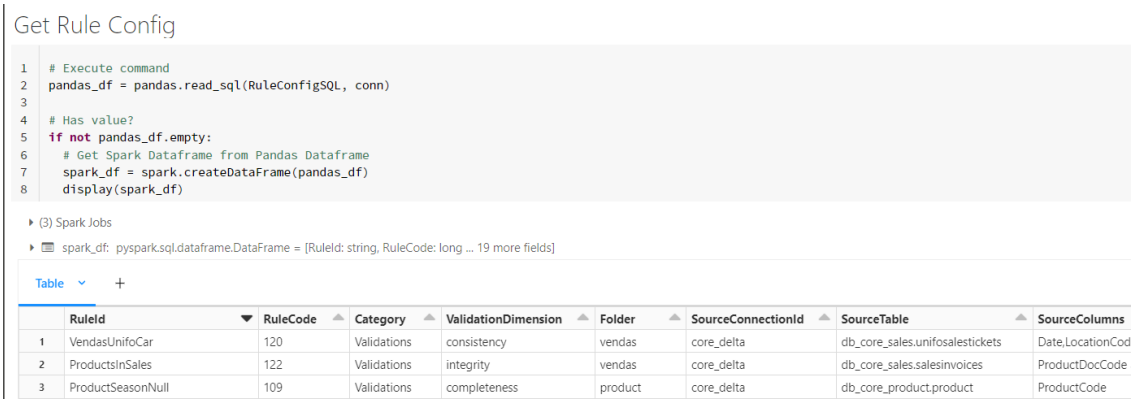


Figura 38 – Teste de integração de ligação ao SQL Server

O segundo teste realizado ao motor de regras foi a ligação do processo de configuração com a Framework de ETL. Neste teste foi aproveitada uma das ligações que a *framework* disponibiliza e acrescentados os parâmetros necessários para executar as regras na *framework*. A Figura 39 ilustra a chamada realizada à *framework* e a ligação realizada com sucesso.

Figura 39 – Teste de integração de ligação à ETL *framework*

Por fim, no desenvolvimento do *dataset,* foi necessário testar a ligação do modelo com o azure data lake. Esta ligação foi necessária para extrair a informação de alarmes contida nos ficheiros parquet. Este teste foi realizado no Power BI com um tipo de ligação ao data lake que é disponibilizada na aplicação.

8.5.3 Testes de performance

Este tipo de testes permite verificar a capacidade de resposta, a disponibilidade e escalabilidade de uma aplicação. Para isso, é realizada uma avaliação de desempenho da aplicação com significativas cargas de trabalho em circunstâncias normais. Este tipo de testes pretende evitar problemas de indisponibilidade em condições de baixos recursos [46].

Os testes de performance foram realizados ao motor de regras de negócio. Estes testes foram realizados com a intenção de verificar se a quantidade de regras configuradas apresentava uma diferença significativa no tempo de processamento do processo. Foram realizados testes com diferentes quantidades de regras para verificar:

- A execução de uma regra demorou 3 minutos e 34 segundos, dos quais 3 minutos e 34 segundos foram gastos na execução do processo ETL e 0,09 segundos foi o tempo gasto na configuração das regras.

- A execução de 7 regras demorou 7 minutos e 58 segundos, dos quais 7 minutos e 57 segundos foram gastos na execução do processo ETL e 0,58 segundos foi o tempo gasto na configuração das regras.

- A execução de 13 regras demorou 11 minutos e 57 segundos, dos quais 11 minutos e 56 segundos foram gastos na execução do processo ETL e 1 segundo foi o tempo gasto na configuração das regras.

- Com os resultados obtidos foi possível verificar que um maior número de regras implica um tempo de processamento da *framework* de ETL significativamente mais demorado. Por outro lado, o processo de configuração de regras apresenta um tempo de processamento relativamente curto e proporcional ao número de regras parametrizadas. Com estes resultados podemos verificar que os diferentes métodos implementados apresentam um tempo de execução relativamente semelhante.

8.6 Comparação de soluções

Nesta secção foi efetuada uma comparação entre a solução utilizada previamente e a solução apresentada neste projeto. Por fim, é apresentada uma tabela de comparação das soluções para uma leitura rápida das vantagens da utilização desta nova solução.

O projeto previamente desenvolvido, apesar de apresentar uma solução capaz de analisar alarmes, implicava um desenvolvimento específico sempre que era necessário criar ou alterar um alarme. No caso de ocorrer a necessidade de acrescentar uma nova validação o processo era demorado e implicava realizar tarefas repetidas.

O processo de acrescentar uma nova regra no projeto previamente utilizado era iniciado por acrescentar a nova regra nas configurações de metadados da *framework* ETL. Contudo, sendo o processo ETL é bastante completo, maior parte das regras apresentam configurações semelhantes. De seguida, era necessário despoletar a regra de forma manual de modo a ficar disponível no *dataset* com o propósito de ser acrescentada à tabela de detalhes de alarmes. Para acrescentar o novo alarme era necessário editar a tabela de alarmes no *dataset* para acrescentar manualmente o novo código e descrição do mesmo.

O presente projeto implica apenas a configuração das regras na tabela de parametrizações da base de dados de SQL Server. Estas parametrizações são suficientes para despoletar a validação da regra no dia seguinte, visto que este processo apresenta uma execução automática que ocorre diariamente.

8.6.1 Tabela de comparação

Com o intuito de facilitar a leitura das diferenças entre as duas soluções, foi criada a Tabela 10. Nesta tabela são comparadas as soluções envolvidas no presente projeto.

Tabela 10 – Comparação de soluções

Funcionalidade	**Solução Prévia**	**Solução desenvolvida**
Parametrização de alarmes /regras	Necessita conhecimentos de SQL para parametrizar regras. Maior parte das parametrizações são semelhantes para todos os alarmes.	Necessita conhecimento da dimensão de qualidade de dados da regra a validar. Permite inserção manual de comando SQL, caso as dimensões não consigam dar resposta á necessidade das regras.
Acrescentar alarme/regra no *dataset*	É necessário acrescentar manualmente código e descrição de novo alarme ao *dataset.*	Não é necessário realizar nenhuma alteração, código e descrição da regra são carregadas de acordo com a parametrização realizada.
ETL Framework	Único mecanismo utilizado na solução para deteção de alarmes.	Utilizado para carregar alarmes detetados.
Separação de regras por dimensão de qualidade de dados	Não realiza distinção entre tipos dimensões de regras.	Separa configurações de regras em dimensões para corresponder às necessidades das regras.
Arquitetura	Básica de um projeto típico de BI, com fontes de informação, processamento de ETL, e visualização de dados através de *reports*.	Acrescentado processo de configuração de regras ao projeto previamente utilizado.
Alarmes detetados	Os alarmes detetados são carregados em ficheiros pelo processamento ETL e analisados através de *reports* Power BI.	Os alarmes são, igualmente, carregados em ficheiros pelo processamento ETL e analisados através de *reports* Power BI.

Funcionalidade	Solução Prévia	Solução desenvolvida
		Posteriormente, é previsto finalizar a implementação do envio de alarmísticas via email.
Dataset	Especificamente montado para a validação de alarmes relativos a vendas e devoluções.	Montado para uma validação mais global de diferentes tipos de regras.
Reports	*Reports* muito focados na apresentação de vendas e devoluções com alarmes.	Para além dos *reports* que já eram disponibilizados foram acrescentados *reports* mais globais de análise de alarmes.

A vantagem mais significativa da utilização da nova solução foi a redução do tempo de desenvolvimento necessário para a criação ou alteração de uma regra de negócio. Na solução anterior, era necessário realizar a montagem de um comando SQL específico para a validação necessária de se realizar. A solução atual permite igualmente a montagem de um comando SQL mas também apresenta um conjunto de parâmetros que permite ao sistema montar o comando SQL internamente. Outro desenvolvimento que era necessário, e que com a utilização desta solução deixa de ser realizado, é a parametrização manual no *dataset* de novas regras de negócio (código e descrição).

9 Conclusão

No presente capítulo foram realizadas as conclusões ao projeto realizado. Para isso, são descritos os objetivos atingidos considerando os objetivos propostos no início do projeto, as contribuições do projeto e, por fim, as limitações durante o desenvolvimento do projeto, assim como, possível trabalho futuro.

9.1 Resultados Alcançados

Relativamente ao projeto idealizado inicialmente, foram cumpridos maior parte dos objetivos que foram propostos. Foi desenvolvido um processo de configuração de regras em metadados, um *dataset* com dados relativos aos alarmes despoletados e um *report* para análise de detalhes dos alarmes detetados.

Numa primeira fase do projeto foi desenvolvido um processo de configuração de parametrizações de regras de negócio em metadados. Para isso, tendo em conta as especificidades de cada regra, foi separada a configuração destas pelo seu tipo de dimensão de qualidade de dados associada às mesmas. Foram então implementados 7 métodos distintos para configurar regras de negócio, nomeadamente: *completeness*, *consistency*, *conformity*, *integrity*, *timeliness*, *uniqueness* e *query*.

Após configuração dos metadados é executada a *framework* ETL, já implementada pela organização. Esta *framework* executa os metadados configurados e carrega os alarmes detetados nos dados em ficheiros. Estes ficheiros, por sua vez, foram utilizados como base no desenvolvimento do *dataset*.

O *dataset* desenvolvido apresenta detalhes dos alertas gerados e ligação a tabelas com detalhes de produtos, lojas, datas, vendas e hora. Por fim, este *dataset* foi utilizado como fonte de dados dos *reports* desenvolvidos para consulta e análise do negócio.

Devido a uma limitação na implementação, não foi possível concluir todos os objetivos inicialmente definidos. Na Tabela 11 são listados os objetivos do projeto e o ponto de situação dos mesmos.

Tabela 11 – Ponto de situação

Objetivo	**Situação**
Motor de verificação de regras de negócio	Realizado
Gerar alarmísticas	Não concluído
Análises capazes de representar anomalias/violações	Realizado

Considerando todo o projeto, o único objetivo que ficou aquém das expectativas foi a geração de alarmísticas, não foi possível concluir a validação de novos alarmes gerados e o envio de emails. Apesar disso, o processo já foi projetado na forma como deve ser desenvolvido.

9.2 Contribuições

A qualidade de dados é essencial para apresentar informação no apoio à decisão, apresentar informação com má qualidade pode levar a tomadas de decisão que não são ideais para a gestão de uma organização, daí a relevância de estudar dimensões de qualidade. A eficácia dos sistemas de informação está diretamente relacionada com a qualidade dos recursos de informação [47].

Avaliar a qualidade dos dados pode ser um processo complexo devido à quantidade de dimensões de qualidade de dados existente. Quando é criada uma solução de qualidade de dados, é necessário entender as relações entre as diferentes dimensões e analisar quais as mais adequadas, tendo em conta os processos de negócio [10].

O presente projeto disponibiliza a configuração de diferentes dimensões de qualidade de dados, tendo em conta que as regras apresentam especificidade distintas. Tem o intuito de diminuir ao máximo a necessidade de desenvolvimento, para configurar regras e possibilitar diferentes tipos de validação.

9.3 Limitações e Trabalho futuro

Apesar do presente projeto corresponder à maior parte dos objetivos propostos inicialmente, na solução idealizada era expectável substituir a *framework* de ETL já desenvolvida pela organização, por uma *framework* especifica de processamento de regras de negócio. Apesar de não ter sido definido nos objetivos que este tipo de implementação seria imperativo, foi tentada a implementação desta solução. No entanto, como a *framework* de ETL é um projeto com bastante dimensão e complexidade, o processo que estava a ser implementado não ia ser vantajoso relativamente à solução que era previamente utilizada. Então, já com uns meses de tentativa de implementação do projeto, foi necessário recomeçar a implementação e o design de uma nova solução que fosse vantajosa para a organização.

Outra limitação deparada durante a implementação deste projeto esteve relacionada com a realização do projeto fora do horário laboral. Em caso de dúvida durante a implementação ou design do projeto, era necessário esperar até ao seguinte dia laboral para ser possível esclarecer dúvidas.

Visto que não foi possível finalizar a notificação de alarmísticas via email devido à limitação descrita inicialmente nesta secção, como trabalho futuro será finalizada a implementação desta funcionalidade. Até porque esta funcionalidade já foi desenhada, definida a sua parametrização, assim como iniciada a sua implementação.

Relativamente às análises realizadas, foi concluído o desenvolvimento do *dataset* e dos *reports* utilizados para análises dos alarmes, no entanto, prevê-se que, com uma maior utilização da solução, seja necessário realizar alterações ao *dataset* e desenvolver novos *reports*.

Posteriormente, também será realizada uma interface gráfica, com o intuito de permitir a diferentes utilizadores interagirem na criação e alteração de regras e de alarmísticas. Esta implementação fornece ao utilizador uma autonomia na validação de regras, deixando, assim

de ser necessário o administrador do sistema gerir a criação ou alteração de novas parametrizações.

Nas validações realizadas neste projeto, o sistema onde são feitas as validações é sempre o mesmo. Como trabalho futuro, o motor de regras deverá permitir ligação a novas fontes de dados, para efetuar validações entre diferentes sistemas.

Referências

[1] M. zur Muehlen and M. Indulska, “Modeling languages for business processes and business rules: A representational analysis,” *Inf. Syst.*, vol. 35, no. 4, pp. 379–390, 2010, doi: 10.1016/j.is.2009.02.006.

[2] I. Graham and J. Wiley, *Business Rules Service Oriented A Pattern Language*. 2006.

[3] J. A. Von Halle, Barbara, Goldberg, Larry, Zachman, *The Business Rule Revolution : Running Business the Right Way*. Happy About, 2006.

[4] T. Morgan, *Business Rules and Information Systems: Aligning IT with Business Goals*. Addison-Wesley Professional, 2002.

[5] E. Rowland Watkins, “Principles of the business rule approach,” *Int. J. Inf. Manage.*, vol. 24, no. 2, pp. 196–197, 2004, doi: 10.1016/j.ijinfomgt.2003.12.007.

[6] M. Chisholm, *How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering*. 2003.

[7] D. Liu, T. Gu, and J. P. Xue, “Rule engine based on improvement rete algorithm,” *2010 Int. Conf. Apperceiving Comput. Intell. Anal. ICACIA 2010 - Proceeding*, pp. 346–349, 2010, doi: 10.1109/ICACIA.2010.5709916.

[8] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, “Methodologies for data quality assessment and improvement,” *ACM Comput. Surv.*, vol. 41, no. 3, 2009, doi: 10.1145/1541880.1541883.

[9] C. Fürber, “Data Quality Management with Semantic Technologies,” *Data Qual. Manag. with Semant. Technol.*, pp. 20–55, 2016, doi: 10.1007/978-3-658-12225-6.

[10] P. H. S. Panahy, F. Sidi, L. S. Affendey, and M. A. Jabar, “The impact of data quality dimensions on business process improvement,” *2014 4th World Congr. Inf. Commun. Technol. WICT 2014*, pp. 70–73, 2014, doi: 10.1109/WICT.2014.7077304.

[11] D. McGilvray, *Executing data quality projects: Ten steps to quality data and trusted information (TM)*, 2ª edição. Elsevier Inc., 2021.

[12] S. Bhandari *et al.*, “An automatic data completeness check framework for open government data,” *Appl. Sci.*, vol. 11, no. 19, 2021, doi: 10.3390/app11199270.

[13] F. Sidi, P. H. Shariat Panahy, L. S. Affendey, M. A. Jabar, H. Ibrahim, and A. Mustapha, “Data quality: A survey of data quality dimensions,” *Proc. - 2012 Int. Conf. Inf. Retr. Knowl. Manag. CAMP’12*, pp. 300–304, 2012, doi: 10.1109/InfRKM.2012.6204995.

[14] R. Mahanti, *Data quality: dimensions, measurement, strategy, management, and governance*. Quality Press, 2019.

[15] M. Yalaoui and S. Boukhedouma, “A survey on data quality: Principles, taxonomies and comparison of approaches.,” *Proc. - 2021 Int. Conf. Inf. Syst. Adv. Technol. ICISAT 2021*, 2021, doi: 10.1109/ICISAT54145.2021.9678209.

[16] H. Zhang and X. Yang, “Rule engine research and implementation in financial system,” *NCM 2009 - 5th Int. Jt. Conf. INC, IMS, IDC*, pp. 1114–1117, 2009, doi: 10.1109/NCM.2009.197.

[17] J. Zhang, J. Yang, and J. Li, “When rule engine meets big data: Design and implementation of a distributed rule engine using spark,” *Proc. - 3rd IEEE Int. Conf. Big Data Comput. Serv. Appl. BigDataService 2017*, pp. 41–49, 2017, doi: 10.1109/BigDataService.2017.17.

[18] P. Sun, L. Luo, S. Liu, and W. Wu, “Adaptive Rule Engine for Anomaly Detection in 5G Mobile Edge Computing,” *Proc. - Companion 2020 IEEE 20th Int. Conf. Softw. Qual. Reliab. Secur. QRS-C 2020*, pp. 690–691, 2020, doi: 10.1109/QRS-C51114.2020.00123.

[19] Microsoft, “Rules Engine for Azure Front Door architecture and terminology | Microsoft Docs,” 2022. https://docs.microsoft.com/en-us/azure/frontdoor/front-door-rules-engine (accessed Jan. 25, 2022).

[20] Databricks, “Introduction to Data Lakes - Databricks.” https://databricks.com/discover/data-lakes/introduction#data-lake (accessed Jan. 26, 2022).

[21] Microsoft, “Introduction to Azure Data Factory - Azure Data Factory | Microsoft Docs,” 2022. https://docs.microsoft.com/en-us/azure/data-factory/introduction (accessed Jan. 25, 2022).

[22] Microsoft, “What is Azure Databricks? | Microsoft Docs,” 2022. https://docs.microsoft.com/en-us/azure/databricks/scenarios/what-is-azure-databricks (accessed Jan. 25, 2022).

[23] Phyton.org, “General Python FAQ — Python 3.10.0 documentation.” [Online]. Available: https://docs.python.org/3/faq/general.html#what-is-python.

[24] PyPI, “durable-rules · PyPI.” https://pypi.org/project/durable-rules/0.33.103/ (accessed Jan. 30, 2022).

[25] PyPI, “rule-engine · PyPI.” https://pypi.org/project/rule-engine/ (accessed Jan. 30, 2022).

[26] PyPI, “business-rule-engine · PyPI.” https://pypi.org/project/business-rule-engine/ (accessed Jan. 30, 2022).

[27] Red Hat Inc, “Drools - Drools - Business Rules Management System (Java™, Open Source.” 2015.

[28] M. Proctor, M. Neale, P. Lin, and M. Frandsen, “Drools documentation,” *Tech. Rep.*, pp. 1–297, 2008.

[29] Microsoft, “O que é Power BI? - Power BI | Microsoft Docs,” 2022. https://docs.microsoft.com/pt-pt/power-bi/fundamentals/power-bi-overview (accessed Jan. 10, 2022).

[30] A. Deardorff, “Tableau (version. 9.1),” *J. Med. Libr. Assoc. 104*, pp. 182–183, 2016.

[31] N. Rich and M. Holweg, “Value Analysis Value Engineering,” p. 32, 2000, [Online]. Available: https://www.urenio.org/tools/en/value_analysis.pdf.

[32] A. Martikainen, “Front End of Innovation in Industrial Organization FACULTY OF TECHNOLOGY Atte Martikainen FRONT END OF INNOVATION IN INDUSTRIAL ORGANIZATION Constructive research Master ’ s Thesis in Industrial Management VAASA 2017,” no. November 2017, 2018, doi: 10.13140/RG.2.2.19644.36481.

[33] J. Paasi, P. Valkokari, and P. Maijala, “Managing Opportunities , Risk and Uncertainties in New Business Creation - Working Report,” *Business*, no. January, 2008.

[34] P. Koen *et al.*, “Providing clarity and a common language to the ‘fuzzy front end,’” *Res. Technol. Manag.*, vol. 44, no. 2, pp. 46–55, 2001, doi: 10.1080/08956308.2001.11671418.

[35] J. Maletic and A. Marcus, “Data Cleansing: A Prelude to Knowledge Discovery,” in *Data Mining and Knowledge Discovery Handbook*, 2nd ed., O. Maimon and L. Rokach, Eds. Springer, Boston, 2009, pp. 19–32.

[36] A. Osterwalder, “The business model ontology a proposition in a design science approach: PhD Dissertation, University of Lausanne,” 2004.

[37] B. A. Osterwalder, Y. Pigneur, G. Bernarda, A. Smith, T. Papadakos, and J. Wiley, “Value Proposition Design: How to create products and services customers want,” *J. Bus. Model.*, vol. 3, no. 1, pp. 81–89, 2015, doi: 10.5278/ojs.jbm.v3i1.1105.

[38] Strategyzer, “VPC Download,” 2021. https://www.strategyzer.com/vpc_thank_you?submissionGuid=db018503-71c9-4638-ba7f-a1eb2ddd0b48 (accessed Feb. 02, 2022).

[39] A. Margherita and D. Verrill, “Elevator Pitch Assessment Model: A Systematization of Dimensions in Technology Entrepreneurship Presentations,” *IEEE Trans. Prof. Commun.*, vol. 64, no. 4, pp. 304–321, 2021, doi: 10.1109/TPC.2021.3110620.

[40] Quality-One, “QFD | Quality Function Deployment | Quality-One,” 2021. https://quality-one.com/qfd/ (accessed Feb. 18, 2022).

[41] I. Sommerville, *Software engineering*, 9th ed. Boston: Addison-Wesley, 2011.

[42] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner’s Approach*, 8th ed. McGraw-Hill, 2014.

[43] A. Silva and C. Videira, “UML, Metodologias e Ferramentas CASE,” 2001.

[44] Databricks, “Create, run, and manage Databricks Jobs,” 2022. https://docs.databricks.com/workflows/jobs/jobs.html (accessed Sep. 11, 2022).

[45] P. J. Lavrakas, “Research Hypothesis,” in *Encyclopedia of Survey Research Methods*, Los Angeles: Sage Publications, Inc, 2008, pp. 731–733.

[46] W. Lewis, “Software Testing Techniques,” *Softw. Test. Contin. Qual. Improv. Third Ed.*, pp. 557–627, 2008, doi: 10.1201/9781439834367.axg.

[47] P. H. Shariat Panahy, F. Sidi, L. S. Affendey, M. A. Jabar, H. Ibrahim, and A. Mustapha,

"A framework to construct data quality dimensions relationships," *Indian J. Sci. Technol.*, vol. 6, no. 5, pp. 4422–4431, 2013, doi: 10.17485/ijst/2013/v6i5.10.

[48] Databricks, "Parquet," 2022. https://www.databricks.com/glossary/what-is-parquet (accessed Sep. 13, 2022).

[49] Microsoft, "Notebooks," 2022. https://learn.microsoft.com/en-us/azure/databricks/notebooks/.

[50] Databricks, "DataFrames," 2022. https://www.databricks.com/glossary/what-are-dataframes.