UNIVERSIDADE FEDERAL DO ABC
Centro de Matemática, Computação e Cognição
Bacharelado em Ciência da Computação

Renan Ferreira Lima

NOVAS ARQUITETURAS DE DADOS E SUAS PERSPECTIVAS

Santo André, SP
2023

RENAN FERREIRA LIMA

NOVAS ARQUITETURAS DE DADOS E SUAS PERSPECTIVAS

Relatório final, apresentado ao Centro de Matemática, Computação e Cognição da Universidade Federal do ABC como exigência para aprovação no Projeto de Graduação em Computação III

Orientador: Prof. Dr. Carlos Alberto Kamienski

Santo André, SP

2023

Dedico este trabalho a minha esposa Camilla e aos meus pais, Reinaldo e Damaris. Vocês são a minha fonte de inspiração e meus maiores incentivadores.

RESUMO

O trabalho tem o objetivo de entender e categorizar as novas arquiteturas de dados, explorando, principalmente, o conceito recente de *Data Mesh*. O *Data Mesh* faz contraponto com os modelos já estabelecidos de arquitetura para coleta, processamento e armazenamento de dados analíticos em larga escala, como o *Data Lake*. Pretende-se entender os motivadores para essa nova abordagem e avaliar seus potenciais benefícios em relação as soluções mais tradicionais. Para isso, foi feita uma revisão bibliográfica, transcorrendo pelas principais arquiteturas de dados para ambientes analíticos da década de 1990 até os últimos anos. Ao longo do trabalho, são apresentadas as perspectivas tecnológicas e necessidades de mercado que contribuíram para o estabelecimento de novos termos e tecnologias. E como estas tecnologias se destacaram e contribuíram para as novas arquiteturas. Por fim, o objetivo é evidenciar como estas transformações têm sinergia com o *Data Mesh* e no que se diferem, apresentando os principais pilares dessa nova arquitetura e alguns paradigmas para sua adoção. Para aprofundar a compreensão do conceito, foi realizada uma modelagem de um sistema de comércio eletrônico usando *Data Mesh*, comparando com uma modelagem usando *Data Lake*. Essa modelagem foi implementada utilizando como base um *template frontend* de uma página de comércio eletrônico, o qual foi adaptado e integrado a um *template backend*. Este, foi restruturado para integrar-se aos bancos e armazenamento de dados que compõe o *Data Mesh* do Projeto.

Palavras-chave: novas arquitetura de dados, *Data Mesh*, *Data Lake*, banco de dados

ABSTRACT

The work aims to understand and categorize the new data architectures, exploring, mainly, the recent concept of Data Mesh. Data Mesh contrasts with the already established architecture models for collecting, processing and storing large-scale analytical data, such as Data Lake. It is intended to understand the motivators for this new approach and evaluate its potential benefits in relation to more traditional solutions. For this, a bibliographical review was carried out, going through the main data architectures for analytical environments from the 1990s until the last years. Throughout the work, the technological perspectives and market needs that contributed to the establishment of new terms and technologies are presented. And how these technologies stood out and contributed to the new architectures. Finally, the objective is to show how these transformations have synergy with Data Mesh and in what they differ, presenting the main pillars of this new architecture and some paradigms for its adoption. To deepen the understanding of the concept, a modeling of an e-commerce system using Data Mesh was carried out, comparing with a modeling using Data Lake. This modeling was implemented using a frontend template of an e-commerce page as a base, which was adapted and integrated into a backend template. This was restructured to integrate with the databases and data storage that make up the Project's Data Mesh.

Keywords: new data architecture, Data Mesh, Data Lake, database

SUMÁRIO

1 INTRODUÇÃO

Os avanços da última década permitiram sistemas robustos capazes de lidar com alto volume de dados e propiciar grande poder de processamento. Entretanto, isso culminou em outras demandas e dificuldades, como analisar grandes quantidades de dados de maneira rápida e precisa, ou ainda, lidar de forma mais eficiente com a necessidade de armazenamento cada vez maior. Este fato tem gerado várias implicações como questões de eficiência financeira, de segurança, de governança, privacidade de dados e limitações técnicas e tecnológicas.

Com um mercado altamente competitivo e conectado, os dados tem um papel principal na gestão estratégica e operacional das organizações. Algumas questões devem ser respondidas corretamente e em tempo hábil para fornecerem insumos para um melhor posicionamento comercial ou uma tomada de decisão mais precisa. Por exemplo, duas questões importantes são: 1) Em que cenário é mais vantajoso oferecer um frete grátis em uma venda online?; 2) Como está a taxa de inadimplência e de oferta de crédito no mercado?.

Prometendo maior velocidade, precisão, controle e geração de valor sobre estes dados, emergiu em 2019 o conceito de *Data Mesh*. O termo foi estabelecido por Zhamak Dehghani, diretora de tecnologias emergentes na Thoughtworks North America. Com foco em sistemas distribuídos, Dehghani é uma precursoras dos pilares do *Data Mesh*. Segundo ela, o *Data Mesh* propõe a ruptura do modelo atual que é a centralização dos dados no *Data Lake*. Para isso apresenta quatro princípios (DEHGHANI, 2022):

a) Propriedade de Domínio (*Domain Ownership*);
b) Dados como um Produto (*Data as a Product*);
c) Plataforma de Dados com Autosserviço (*Self-Serve Data Platform*);
d) Governança Computacional Federada (*Federated Computational Governance*).

Com relação a tecnologias, o *Data Mesh* é primariamente uma abordagem organizacional, sendo estruturado por premissas arquiteturais. Desse modo, não é fundamentado em uma tecnologia específica. No entanto é possível definir padrões e eleger as tecnologias mais aderentes.

Quanto as opções de *softwares* e serviços que podem ser empregadas, destacam-se soluções em nuvens públicas, ou em *Cloud*, que já são utilizadas em *Data Lakes*, como Google Cloud ("Google Cloud Platform", 2023), AWS ("Amazon Web Services (AWS)", 2023) e Azure ("Microsoft Azure", 2023), que são utilizadas para armazenar, processar, observar, automatizar e catalogar dados. Essas funções também podem ser executadas por softwares *open source* equivalentes, como, Minio ("MinIO", 2023), Spark ("Apache Spark™", 2023), Elastic Observability ("Elastic Observability", 2023), Airflow ("Apache Airflow", 2023) e Datahub ("A Metadata Platform for the Modern Data Stack", 2023).

Outro pilar do *Data Mesh* é o foco em casos de uso analíticos que visam simplificar esteiras de dados complexas presentes em *Data Lakes*. Dessa forma, não estabelece padrões para os dados operacionais ou transacionais.

Por se tratar de um conceito fundamentalmente organizacional, trabalhando com a proposta de Propriedade e Arquitetura de dados descentralizada e orientada ao Domínio (DEHGHANI, 2020), o *Data Mesh* não opera bem em alguns cenários. Por exemplo, quando há baixa maturidade na Arquitetura e na Governança Corporativa ou há um baixo nível de esteiras de dados automatizadas (MOURA; SOTTA; RUBINOV, 2023).

Entender as dificuldades do *Data Mesh* envolve explorar os avanços das arquiteturas de dados e suas limitações. Sendo assim, é fundamental analisar e compreender se o *Data Mesh* é algo viável para as organizações em geral e não apenas um conceito promissor. Neste contexto, esse trabalho tem o intuito de apresentar as principais abordagens em relação à Arquitetura de dados nas últimas décadas, sem a ambição de esgotar o tema.

2 REVISÃO BIBLIOGRÁFICA

2.1 Os Precursores do *Data Mesh*

O *Data Mesh* é uma nova abordagem para fornecer, gerenciar e acessar dados em cenários analíticos escaláveis (DEHGHANI, 2022). É um contraponto das arquiteturas de dados centralizadas que foram estabelecidas e aprimoradas nas últimas décadas.

Propõe que através de mudanças técnicas e organizacionais embasadas em uma visão descentralizada, é possível obter ganhos em relação as arquiteturas de dados mais usuais (DEHGHANI, 2022). Mediante isso, é fundamental apresentar alguns modelos anteriores para então explorar o proposto pelo *Data Mesh*.

Dessa forma, os próximos tópicos tratam das arquiteturas de dados que se consolidaram nas últimas décadas, como é o caso do *Data Warehouse* e do *Data Lake.* Além de, introduzir conceitos que são essenciais para entender os eventuais ganhos que o *Data Mesh* pode oferecer.

2.2 *Data Warehouse*

O avanço do poder computacional e a sofisticação das ferramentas e técnicas para análise de dados, em conjunto com necessidades estratégicas de negócios, resultaram no desenvolvimento do conceito *Data Warehouse* (ELMASRI; NAVATHE, 2010). Segundo William H. Inmon, também conhecido como Bill Inmon, reconhecido como o primeiro a usar o termo no início da década de 1990, um *Data Warehouse* é uma coleção de dados orientada a assunto, integrada, não volátil, variável no tempo para suporte às decisões de gerência (INMON, 2005).

Diferentemente dos bancos de dados tradicionais baseados em processamento de transação on-line (sigla OLTP, em inglês), o *Data Warehouse* opera com o processamento

analítico on-line (sigla OLAP, em inglês). Dentre as características do *Data Warehouse*, destacam-se:

a) são estruturados para fornecer dados para o apoio de decisões, e assim, integram múltiplas fontes;
b) costumam armazenar os dados de forma estruturada e multidimensional;
c) trabalham com dados históricos e não voláteis.

Estas características demandam uma preparação dos dados, processo executado por plataformas de ETL que é um acrônimo para *Extract* ou Extração, *Transform* ou Transformação e *Load* ou Carga) (ELMASRI; NAVATHE, 2010). Estas tecnologias capturam dados em bancos de dados transacionais ou em outras origens (extração), aplicam regras de negócios neles (transformação) e carregam no *Data Warehouse* (carga). Como os dados são não voláteis, é necessário que o processo seja incrementado periodicamente, fornecendo uma visão histórica e contínua, e.g., as vendas dos últimos 12 meses em uma empresa.

Uma vez que os dados são carregados, é possível acessá-los a partir de plataformas de *Business Intelligence* (BI). Em relação as análises, estas são feitas a partir das diversas dimensões dos dados. As quais permitem realizar estudos analíticos com os mais variados graus de detalhamento, fomentando visões de negócios mais apuradas.

2.2.1 Relação com o Business Intelligence

O termo *Business Intelligence* (BI) passou a ser repercutido no começo dos anos de 1990 pelo então analista do Gartner Group, Howard Dessner. Esse termo foi usado para designar as plataformas baseadas no OLAP e com o passar do tempo passou a associar as plataformas de *analytics* em geral (WATSON; WIXOM, 2007).

Na camada de BI são consumidos os dados tratados de um *Data Warehouse*, i.e., aqueles que já estão estruturados por assunto, possuem alguma agregação ou regras de negócios aplicadas. Estes dados são os pilares para tomada de decisão e obtê-los requer um

grande esforço de processamento e, consequentemente, um custo elevado. Contudo, antes de chegar a camada de BI, os dados passam por outras camadas do *Data Warehouse*.

2.2.2 Estrutura de um Data Warehouse

Os bancos de dados tradicionais são projetados para o OLTP, e.g., são otimizados para inserções, atualizações e exclusões, com suporte também para requisitos de consulta de informação. Nesse tipo de banco, a estratégia é conciliar a integridade das transações com o processo de consulta. Sendo assim, eles não são otimizados para cenários de OLAP, onde as consultas envolvem agregações e agrupamentos (ELMASRI; NAVATHE, 2010).

Em contraponto, o *Data Warehouse* é projetado exatamente para dar suporte à extração, processamento e apresentação de dados para viés analítico, i.e., contempla o cenário OLAP. Em um *Data Warehouse* é possível consultar volumes expressivos de dados e realizar agregações sem comprometer a disponibilidade do sistema. Além disso, os dados deste tipo de banco podem ter as mais diversas origens. Em termos dos dados analisados, o *Data Warehouse* tende a possuir uma quantidade maior do que a origem, pois as tabelas, na maior parte das vezes, trazem dados históricos e não apenas as transações mais recentes (ELMASRI; NAVATHE, 2010).

Neste cenário, visando questões organizacionais e pertinentes aos requisitos do OLAP, juntamente com as necessidades das plataformas de *BI*. A arquitetura de um *Data Warehouse*, tem sua estrutura otimizada para aproveitar o relacionamento dos dados e construir visões, ou *views*, com regras de negócios.

No âmbito de estrutura, destacam-se as seguintes divisões (ELMASRI; NAVATHE, 2010):

a) *Data Warehouse* em nível empresarial: são estruturas centralizadas e de grande porte, assim requerem investimentos maciços de tempo e recursos;
b) *Data Warehouse* virtual: esse tipo de estrutura oferece visões (*views*) de bancos de dados operacionais que são materializadas para acesso eficiente;

c) *Data marts*: é uma estrutura baseada em subconjuntos da organização, i.e., é mais descentralizada, com foco mais estreito, e.g., *data marts* por departamento.

2.2.3 Modelagem de um Data Warehouse

Para favorecer as análises, são modeladas estruturas que tiram proveito dos relacionamentos inerentes dos dados, estas são denominadas de cubos de dados ou hipercubos, caso haja mais de três dimensões. Neles, os dados são preenchidos em matrizes multidimensionais, propiciando um desempenho de consulta mais satisfatório do que no modelo relacional. Como exemplos de dimensões pode-se citar períodos fiscais, produtos e regiões da empresa (ELMASRI; NAVATHE, 2010).

A partir deste exemplo, seria possível elaborar um cubo no qual cada eixo seria composto por uma das dimensões. Com a técnica de giro, também denominada de rotação, é possível alterar, de forma imaginária, a orientação do cubo. Isso permite obter algo equivalente a uma tabela de vendas regionais para cada produto separadamente, onde cada tabela mostra vendas trimestrais para esse produto região por região.

No cubo é possível aplicar ainda a exibição *roll-up* ou a *drill-down*. Sendo que a *roll-up* sobe na hierarquia, isto é, agrupa em unidades maiores na dimensão, por exemplo mostrando o período em meses ao invés de semanas. Enquanto que a *drill-down* faz o oposto, exibe as dimensões em unidades menores. Ainda no exemplo anterior, ao invés de semanas, o período seria exibido em dias.

O modelo de armazenamento multidimensional envolve dois tipos de tabelas: tabelas de dimensão e tabelas de fatos. Uma tabela de dimensão traz tuplas com os atributos de dimensão do objeto em questão, i.e., para um produto, a tabela de dimensão traria os atributos que especificam aquele produto como: nome, código e descrição. Já a tabela de fatos, traz tuplas para cada fato registrado. Voltando para o exemplo anterior, nesta tabela poderiam ter

os dados de vendas do produto, como: região da venda, quantidade vendida e receita (ELMASRI; NAVATHE, 2010).

Quanto aos esquemas multidimensionais, destacam-se o esquema estrela (*star schema*) e o esquema floco de neve (*snowflake schema*). O esquema estrela tem uma tabela central de fatos e possui apenas uma tabela para cada dimensão, enquanto que o esquema floco de neve é uma variação do primeiro, possuindo mais de uma tabela por dimensão, de modo que estas sejam normalizadas (ISEMINGER et al., 2023).

Quando há um conjunto de tabelas fatos que compartilham algumas tabelas de dimensão, então há uma constelação de fatos (ELMASRI; NAVATHE, 2010). Um exemplo ocorre quando duas tabelas fatos consomem uma tabela de dimensão de algum produto, esta estratégia limita o universo de busca dentro do *Data Warehouse*, otimizando o processo.

Para a otimização de consultas, há uma técnica denominada de indexação *bitmap* que consiste em um vetor de *bits* para cada valor em um domínio (coluna), que está sendo indexado (ELMASRI; NAVATHE, 2010). Quando há baixa cardinalidade, isto é, quando há um número menor de valores distintos dentro da coluna, essa técnica tem um bom desempenho (“6 Indexes”, 2023). Um exemplo, seria criar um índice *bitmap* em uma coluna de números inteiros, onde a coluna de bits poderia ser definida tendo em vista alguma ordem de grandeza, i.e., números maiores que dez, números iguais a dez e números menores que dez. Esta estratégia propicia um baixo tempo de resposta para diversas consultas *ad hoc*, além de otimizar o processamento de agregação, junções e outras operações. Reduzindo assim, a capacidade de hardware necessária (“6 Indexes”, 2023).

O bom desempenho para consultas analíticas e o suporte para análises de dados complexas, foram alguns dos fatores de sucesso do *Data Warehouse,* propiciando impactos positivos para as organizações. Entretanto, desde a consolidação do conceito, fatores operacionais significativos como a construção, a administração e o controle de qualidade (ELMASRI; NAVATHE, 2010), além de desafiantes, também são limitantes.

2.2.4 Desafios e Limitações do Data Warehouse

Uma das principais limitações do *Data Warehouse* é referente a modelagem e utilização dos dados. Antes de realizar a modelagem, é necessário ter uma ampla visão de como os dados serão utilizados para construir modelos dimensionais eficientes. Além de ser necessário controlar continuamente a qualidade e consistência dos dados carregados, tendo em vista que as fontes são heterogêneas e distintas. Assim, mudanças na estrutura de dados ou em estratégias de negócios, podem implicar em refatorações complexas no *Data Warehouse (ADADI, 2021)*.

Outro fator é a modularização, onde, por exemplo, há uma contínua melhoria dos componentes de armazenamento. De forma que estes componentes são constantemente atualizados e, periodicamente, precisam ser substituídos. Este processo deve ser feito sem grande impacto na operação, o que implica em desafios técnicos e operacionais (ELMASRI; NAVATHE, 2010). Além disso, o custo de armazenamento é elevado por causa do volume de dados históricos armazenados e de sua estrutura.

Na camada de negócios os *Data Marts* tendem a se transformarem em silos de dados, representando versões reduzidas do *Data Warehouse* corporativo. Isto implica em desafios para obtenção da fonte única da verdade. Correndo-se o risco de existir dados destoantes para um mesmo tema. I.e., dois indicadores distintos para vendas ou número de clientes, diminuindo a confiança e credibilidade dos dados (FANG, 2015).

Por fim, há uma necessidade de profissionais altamente especializados para implantar e sustentar as esteiras de ETL e BI. Estes fatores implicam em limitações para criação de soluções escaláveis, o que passou a ser uma necessidade constante com a consolidação da Internet (FANG, 2015).

Soma-se a isso, a necessidade de coletar e analisar dados semiestruturados ou não estruturados, além dos dados estruturados consumidos pelo *Data Warehouse*. Além das novas necessidades de negócios em relação à utilização dos dados coletados (FANG, 2015).

De modo que, passou a ser necessário ter mais flexibilidade na previsibilidade de uso dos dados, i.e., sem diretrizes totalmente claras sobre a utilização. Isto, abriu espaço para um novo conceito (FANG, 2015).

2.3 *A Era do BIG DATA*

Em 1965 Gordon Moore estimou que a densidade dos transistores em uma placa de circuito integrado dobrava a cada 2 anos. Esta estimativa é conhecida como "Lei de Moore" e foi aplicada a todos os aspectos da computação, desde capacidade de processamento à tamanho de processadores. Entretanto, as taxas de crescimento dos volumes de dados crescem de forma mais rápida do que a Lei de Moore, mais do que dobrando a cada dezoito meses (CHANG; GRADY, 2019).

Relacionado a isso, está a expansão e avanço da Internet na década de 1990 que mudou a forma de comunicação e aumentou significativamente o volume de dados gerados (ADADI, 2021). Em conjunto, veio o aumento constante do poder de processamento de dados, juntamente com o avanço de técnicas e tecnologias para a análise, o que estabeleceu no começo dos anos 2000, a era do *Big Data* (ADADI, 2021). O *Big Data* pode ser definido a partir de um conjunto de características conhecidas como os "Vs" do *Big Data*. As quatro características principais são (CHANG; GRADY, 2019):

a) **volume**: representa o tamanho do conjunto de dados;
b) **variedade**: representa as diversas origens, ou seja, dados de vários repositórios, domínios e tipos;
c) **velocidade**: representa a taxa de fluxo dos dados;
d) **variabilidade**: representa a mudança em outras características.

Estas características foram determinantes para a construção de novas Arquiteturas de dados (CHANG; GRADY, 2019). Neste contexto, segundo Miloslavskaya, N., & Tolstoy, A. (MILOSLAVSKAYA; TOLSTOY, 2016), pode ser interpretado como um conjunto de dados cujo tamanho e estrutura excedem as capacidades de ferramentas de tecnologia convencionais

para executar operações de coleta, armazenamento e processamento de dados. Por consequência, excedem a capacidade de percepção por um ser humano.

Por ter diversas origens, como IoT (*Internet of Things*), tráfego de rede, vídeos, e-mails, sistemas de vigilância, entre muitas outras, o *Big Data* é composto, majoritariamente, de dados não estruturados. Neste universo, há demandas que implicam em diferentes formas de se utilizar tais dados, as quais interferem diretamente na forma de processamento. Neste contexto, há desde de necessidades de análise em períodos espaçados de tempo preestabelecidas até em análises em tempo real. Dessa forma, pode-se destacar os seguintes tipos de processamento em *Big Data (MILOSLAVSKAYA; TOLSTOY, 2016)*:

a) processamento em lote (ou *batch*) em tempo próximo ao real (ou pseudo-real, ou ainda, *soft real-time*): Consiste no processamento dos dados armazenados em memória não volátil e cujo o objetivo é oferecer um bom desempenho para tratar variáveis nos dados que ocorram em janelas curtas de tempo, i. e., pode ser utilizado como arquitetura em modelos de *machine learning,* fornecendo dados para treinamento de modelos preditivos;
b) processamento híbrido: Combina as estratégias das arquiteturas em lote e em s*tream*, isto é, possui camadas para trabalhar com dados em lote e, assim, atender demandas que não necessitam de processamento em tempo próximo ao real, como por exemplo, aplicativos de recomendações de produtos baseados no perfil do usuário. E camadas s*tream* para dados que precisam ser processados em tempo de coleta, por exemplo, atualizações de perfil de usuário conforme eventos em um site. Este tipo de processamento é utilizado na Arquitetura Lambda, proposta por Nathan Marz por volta de 2011 para permitir análises de dados mais rápidas e em tempo próximo ao real, cenário até então inviável pelos processos tradicionais de ETL (LIN, 2017);
c) processamento *stream* em tempo muito próximo ao real (ou *hard real-time*): Utiliza dados armazenados em mídia volátil no processamento, i.e., em memória. A qual utiliza parâmetros como taxa de entrada de dados para ajustar o processo, retornar uma resposta e persistir os dados em um armazenamento não volátil. Dessa forma, os dados são processados a medida que chegam, de

maneira continua, i.e., em fluxo. Esta estratégia de processamento é utilizada na Arquitetura Kappa, a qual foi apresentada em 2014 por Jay Kreps para atender casos de usos não suportados pela Arquitetura Lambda. Por exemplo, onde o baixo tempo de resposta é determinante. Nesta arquitetura o tempo de resposta é muito baixo e pode ser utilizada, por exemplo, em cenários como de prevenção à fraudes, onde o fator tempo é crítico (LIN, 2017).

Outra característica do *Big Data* é o escopo, o qual difere da TI tradicional, uma vez que a tecnologia e infraestrutura deixam de ser atores principais, abrindo espaço para fluxos de geração de valor através da tecnologia, i.e., através do uso de Ciência de dados e técnicas de computação distribuída (MILOSLAVSKAYA; TOLSTOY, 2016). De modo que, sistemas monólitos tendem a ser substituídos por *clusters* de processamento de dados, que aliados a modelos de *machine learning* e de sistemas inteligentes, viabilizam análises profundas em grandes volumes de dados.

Em geral, o processamento de *Big Data* visa obter resultados relevantes ao analisar essas grandes massas de dados, os quais podem ser oportunidades de negócios, redução de custo, otimizações, entre outros. Para isso, integra diversas técnicas de várias disciplinas, como: bancos de dados e *Data Warehouses*, estatística, aprendizado de máquina, computação de alto desempenho, reconhecimento de padrões, redes neurais, visualização de dados, recuperação de informações, processamento de imagens e sinais e análise de dados espaciais ou temporais (MILOSLAVSKAYA; TOLSTOY, 2016).

Para que o fluxo de valor seja atingido, o *Big Data* utiliza algumas premissas (MILOSLAVSKAYA; TOLSTOY, 2016):

a) os dados devem ser precisos: para obter-se bons resultados é necessário que os dados estejam corretos e venham de fontes confiáveis;
b) ciclo de vida dos dados: os dados devem ser atuais ou refletir o estado atual da organização;
c) os dados devem ser abrangentes: precisam refletir um contexto amplo, de modo que possam ser trabalhados e entregar informações úteis;

d) relevantes: os dados precisam ser reais e aplicáveis à organização que os utiliza.

Para atender a estas premissas é fundamental uma frente de Governança de dados que entenda as necessidades da organização e zele para que as boas práticas sejam aplicadas para obtenção do fluxo de valor. Além de dar visibilidade sobre os dados disponíveis e como utilizá-los corretamente.

2.3.1 O início do Data Lake

Em meio a isso, novas necessidades emergiram e a estratégia de captar grandes volumes de dados culminou em um novo conceito, o *Data Lake* (KHINE; WANG, 2018), termo que foi inicialmente introduzido por James Dixon em 2010 (MILOSLAVSKAYA; TOLSTOY, 2016). Segundo Fang, H (FANG, 2015) "um *Data Lake* é uma metodologia habilitada por um repositório baseado em tecnologias de baixo custo que melhoram a captura, refinamento, arquivamento e exploração de dados brutos dentro de uma empresa. Um *Data Lake* contém uma diversidade de dados brutos não estruturados ou multiestruturados que, em sua maioria, têm valor não reconhecido para a empresa".

Esta necessidade em relação às estruturas de dados, foi evidenciada nos primeiros exemplos de *Data Lake* criados para armazenar *web data* em empresas de *websites* (FANG, 2015). Neste contexto, observou-se que era necessário que essa plataforma oferecesse mais flexibilidade a medida que novas estruturas de dados foram surgindo.

Com este novo conceito vieram novas tecnologias, destacando-se o *Hadoop* e seu ecossistema *open source*. Além de novas abordagens para armazenamento de dados, como os *Object Storages,* ou em português, Armazenamento de Objetos.

Este tipo de armazenamento é comumente utilizado em nuvens públicas, um exemplo é o S3 da Amazon ("Armazenamento S3", 2023), o Cloud Storage da Google ("Cloud Storage", 2023) e o Azure Blob Storage ("Azure Blob Storage", 2023) ou Azure ADLS2 (ESTABROOK et al., 2023) da Microsoft. Diferente do armazenamento em pastas que utiliza

hierarquias e rígidos protocolos de acesso, os *Object Storages* utilizam estruturas denominadas *buckets* para organizar os dados. Tais estruturas, combinam os dados com os metadados definidos pelo usuário, anexando um identificador personalizado. Este possibilita um acesso único e ao mesmo tempo viabiliza a criação de réplicas por várias regiões. Este tipo de armazenamento é capaz de comportar grande volume de dados nos mais diversos formatos como: *Parquet, csv, json,* assim como, imagens e vídeos ("O que é armazenamento de objetos?", 2023).

Esta diversidade de formatos tornam os *Object Storages* compatíveis com os diversos casos de uso do *Data Lake* ("O que é um data lake?", 2023). Esta nova abordagem de armazenamento, representou avanços em relação ao *Data Warehouse*, possibilitando novos cenários de uso.

2.3.2 Comparação entre Data Lake e *Data Warehouse*

Diferentemente do *Data Warehouse,* que tem um custo elevado de armazenamento, o *Data Lake* utiliza armazenamento mais econômico e por isso consegue oferecer mais escalabilidade. No *Data Warehouse* é primordial todo um tratamento de dados através de processos de ETL, além de uma visão multidimensional com tabelas fato e dimensão.

O *Data Lake* passa a preconizar a ingestão dos dados brutos, de maneira centralizada, sob responsabilidade de times de dados corporativos, i.e., times de Engenharia de dados. Os quais, posteriormente, desenvolvem esteiras, de acordo com convenções negócios, para os dados carregados. Isto é, não há uma preocupação inicial com a modelagem dos dados e quais perguntas de negócios eles vão responder, em vez disso, prioriza-se armazenamento conforme capturado da origem ("O que é um data lake?", 2023).

O Quadro 1 a seguir traz um breve comparativo entre a finalidade de um *Data Lake* e de um *Data Warehouse*, ele tem como base um comparativo realizado pela Amazon Web Services (AWS):

Quadro 1 - Comparativo entre Data Lake e Data Warehouse

Características	**Data Lake**	**Data Warehouse**
Dados	Não relacionais e relacionais de diversas origens, como: dispositivos de IoT, sites, aplicações móveis, mídia social e aplicações corporativas	Relacionais de sistemas transacionais, como: bancos de dados operacionais e aplicações de linha de negócios
Esquema/Modelo de dados	Gravado no momento da análise, também conhecido como esquema na leitura	Definido antes da implementação do DW, também conhecido como esquema na gravação
Preço/performance	Oferece alta escalabilidade em relação ao volume e processamento de dados, com um custo de armazenamento menor.	Resultados de consulta rápidos, contudo usam armazenamento de maior custo
Qualidade dos dados	Quaisquer dados, selecionados ou não, isto é, dados brutos	Dados altamente selecionados, que representam a versão central da verdade
Usuários	Cientistas de dados, Desenvolvedores de dados e Analistas de negócios (usando dados selecionados)	Analistas de negócios
Análises	Machine learning, análises preditivas, descoberta de dados e criação de perfis	Geração de relatórios em lote, BI e visualizações

Fonte: Amazon Web Services (“O que é um data lake?”, 2023).

O quadro evidencia o aspecto mais generalista do *Data Lake*, entretanto cabe enfatizar que o propósito do *Data Lake* não é de substituir o *Data Warehouse*, mas sim complementar esta abordagem (“Data Lake vs. Warehouse”, 2023).

Os casos de uso são diferentes, de modo que o *Data Lake* viabilizou novas oportunidades para os dados analíticos, como: *machine learning*, análise de dados em *streaming*, análise de *Big Data* e outros já citados anteriormente. Enquanto o *Data Warehouse*, tem uma boa performance em cenários de dados analíticos estruturados, onde é

preciso utilizar consultas complexas e há forte relacionamento entre os dados (“Data Lake vs. Warehouse”, 2023).

Outra característica do *Data Lake* é a estratégia de trabalhar em camadas, também conhecidas como zonas, onde o dado é tratado e estruturado conforme as necessidades de negócios. Dessa forma, possibilita a utilização de bancos de dados SQL e NoSQL, além dos *Obejct Storages*, assim como, recursos de processamento OLAP e OLTP (MILOSLAVSKAYA; TOLSTOY, 2016).

2.3.3 Camadas de dados em um Data Lake

As camadas possibilitam uma melhor gestão dos dados e viabilizam a integração com um *Data Warehouse.* Onde o *Data Lake* passa a fornecer dados para o *Data Warehouse* a medida que as finalidades e estratégias de uso vão sendo definidas a partir os dados brutos (“Data Lake vs. Warehouse”, 2023). Voltando ao exemplo das nuvens pública, na AWS há arquiteturas para *Data Lakes* que combinam o S3 (*Object Storage*) com *clusters Redshift* (SQL) para análise de *Business Intelligence* e com *DynamoDB* (NoSQL) para dados em tempo real (“Data Lake | Implementations | AWS Solutions”, 2023). Já na Google, há o Cloud Storage (*Object Storage*) combinado com o *BigQuery* (SQL) e com o BigTable (NoSQL) para análises em tempo real (“Cloud Storage como um data lake”, 2023). Também há estruturas semelhantes na Azure (PRASAD et al., 2022) e em outras nuvens públicas.

Quando a arquitetura do *Data Lake* é dividida em camadas, é comum o uso de três: a primeira para dados brutos, a segunda para tratamento de dados e a terceira para disponibilização de dados refinados (BOSWELL et al., 2023). A camada de dados brutos, também denominada *Raw* ou *Bronze*, tem a proposta de ser uma réplica da origem com os dados armazenados em seu formato original como csv ou json. Ou para otimizar custos de armazenamento, em formatos compactados como o Avro, o Parquet ou o ORC (BOSWELL et al., 2023). A camada bruta disponibiliza os dados apenas para leitura, de modo a garantir a

integridade conforme a origem. Quanto a carga de dados, há duas estratégias principais: a carga completa ou a carga incremental.

A carga completa carrega todos os dados da origem sempre que é feita uma nova carga na camada bruta. Esta estratégia é mais viável em cenários onde o volume de dados na origem é pequeno, ou quando não há um controle de versão do dado na origem, ou ainda, quando os dados são sempre recriados na origem. Enquanto que a carga incremental, traz os dados criados na origem no intervalo desde a última ingestão para a camada bruta. A carga incremental é mais convencional em estruturas com grande volume de dados, ou quando se tem controle de versão dos dados na origem, ou ainda, quando há controle de atualização de dados na origem, i. e., os arquivos não são recriados (BOSWELL et al., 2023).

Uma vez na camada bruta, os dados podem ser transportados para a camada enriquecida, a qual é conhecida como *Trusted* ou *Silver*. Nesta camada, os dados passam por filtragens e começam a ser estruturados e segmentados por assuntos. Também podem ser enriquecidos e disponibilizados para áreas que realizam análise avançadas e exploração de dados, como áreas de Ciências de dados. Contudo, nesta camada ainda não há dados em estado adequado para *Business Inteligence* (BOSWELL et al., 2023).

Este tipo de dado pode ser encontrado na terceira camada, também denominada como camada *Final*, *Refined* ou *Gold*. Nesta camada, os dados estão estruturados de maneira a atender casos de uso específicos, de forma que estão prontos para serem consumidos. Esta camada pode ser integrada a um *Data Warehouse*, onde os dados poderão ser utilizados em painéis e relatórios de negócios (“Recommended data layers - AWS Prescriptive Guidance”, 2023). Além disso, os dados na camada *Final* podem ser utilizados por times de ciências de dados para refinar produtos desenvolvidos a partir da camada *Trusted* (BOSWELL et al., 2023).

Outra maneira de estruturar as camadas do *Data Lake* é seguindo o ciclo de vida dos dados. Isto é, ter uma camada para dados mais recentes, e.g., até 6 meses, e outra camada para dados mais antigos, i.e., uma camada de arquivamento. De modo que os dados arquivados possam ser armazenados em *buckets* de menor custo, os quais não são indicados para consultas frequentes. Esta estratégia otimiza custos e contribui para manter apenas os dados necessários no *Data Lake* (MILOSLAVSKAYA; TOLSTOY, 2016).

Quanto ao transporte de dados entre as camadas, este é feito através do processamento de dados com tecnologias específicas. Uma vez que, precisa atender requisitos de volumetria, assim como, de tempo de processamento e de estruturas distintas de dados. Por contemplar estes aspectos, como citado anteriormente, o ecossistema do *Hadoop* foi um dos pilares para a popularização do conceito do *Data Lake,* sendo amplamente utilizado no processamento dos dados (KHINE; WANG, 2018).

2.3.4 Processamento de dados em um Data Lake

A adesão ao *Hadoop* ocorreu devido à sua capacidade de lidar de forma satisfatória com dados heterogêneos. Isto é, de maneira distribuída e resiliente a falhas em cenários de ingestão em lotes (*batch*). Para isso, utiliza o sistema de arquivos *HDFS* (*Hadoop Distributed File System*) e explora o processamento paralelo utilizando o *MapReduce* (KHINE; WANG, 2018).

O *MapReduce* foi inicialmente apresentado em 2004 por Jeffrey Dean e Sanjay Ghemawat, ambos do Google (DEAN; GHEMAWAT, 2008). A proposta desta estratégia é de otimizar o processamento de dados em *clusters*, de modo a organizar e obter o melhor aproveitamento e distribuição dos nós (FOWLER; SADALAGE, 2013). O nome *MapReduce*, ou em português, Mapear e Reduzir remete as inspirações em linguagens funcionais que utilizam operações de Mapeamento e redução em coleções (FOWLER; SADALAGE, 2013).

Esta estratégia pode ser exemplificada considerando um cenário de clientes e pedidos. Neste cenário, um pedido é composto por um ou mais itens, os quais representam algum produto. Para obter o acumulado de vendas de determinado produto, será necessário acessar diversos pedidos, os quais estão distribuídos em diversos nós do *cluster*. Este tipo de contexto, é um caso de uso do *MapReduce.* Onde a primeira etapa seria aplicar o Mapeamento, ou *Map*, a qual recebe um agregado e retorna alguns pares de chave-valor (FOWLER; SADALAGE, 2013).

Voltando ao exemplo, o *Map* receberia o pedido e retornaria para cada item um chave-valor. Assim, cada par teria a identificação do produto como chave e um Mapa embutido com a quantidade e preços como valores. Uma característica do *Map* é sua independência de outras funções, podendo ser paralelizado de maneira segura, o que implica em maior eficiência de processamento (FOWLER; SADALAGE, 2013).

Em relação a função *Reduce*, esta recebe múltiplos resultados do *Map* e sumariza os valores. Então, caso o *Map* retorne dezenas de linhas de chave-valor para um produto, o *Reduce* pode agregar todas as linhas, resultando em uma única linha para o identificador (FOWLER; SADALAGE, 2013). O *framework MapReduce* organiza o processamento de modo que o *Map* seja aplicado corretamente e por todos os nós do *cluster*, além de assegurar, que em seguida, tudo passe pelo *Reduce*.

No *Hadoop* este framework é utilizado para organizar o processamento distribuído. Em linhas gerais, em um *cluster Hadoop*, há um ponto central de falha e escalabilidade, sendo que nos nós ocorrem cópias de blocos de dados, que são processados tendo como base o atributo chave-valor na abordagem do *MapReduce (KHINE; WANG, 2018)*. O resultado final do processamento é persistido em arquivos HDFS, padrão nativo do *Hadoop* (“Hadoop vs. Spark”, 2023). O ecossistema do *Hadoop* é composto por tecnologias como:

a) *Avro*: para serealização e armazenamento de dados;
b) *Pig*: que é uma plataforma para criação de programas e análise de grandes conjuntos de dados;
c) *HBase*: que é um banco de dados NoSQL distribuído, do tipo família de colunas;
d) *Zookeeper*: que é um serviço centralizado para coordenação de aplicações distribuídas;
e) *Hive*: como plataforma de *Data Warehouse* e que usa a linguagem *HiveQL*, a qual é baseada no SQL (“Hadoop MapReduce”, 2023).

O *Hadoop* trouxe grandes avanços na ingestão de cargas massivas de dados e permitiu uma evolução em relação aos tradicionais processos de ETL utilizados pelo *Data Warehouse*. Entretanto, é uma estratégia direcionada ao processamento em lotes, ou *batch,* uma vez que o

MapReduce tem uma latência significativa devido aos processos de leitura e escrita em disco em cada etapa de execução, incrementando os tempos de operação, isto é, latência de *I/O (*"What is Apache Spark?", 2023). Dessa forma tendo em vista casos de uso como o da arquitetura *Lambda* e *Kappa*, onde há o consumo de dados analíticos em tempo real, foi necessário trabalhar com novas abordagens ("Hadoop vs. Spark", 2023). Tornando-se comum, a adoção de tecnologias como o *Apache Spark* e o *Apache Beam* (DEHGHANI, 2019).

2.3.5 Uma nova geração de Data Lake

O *Spark* foi iniciado em 2009 por Matei Zaharia no *AMPLab* da *University of California, Berkeley*. Em 2013, foi doado para a *Apache Software Foundation*, onde recebeu diversas contribuições desde então. O diferencial do *Spark* em relação ao *MapReduce* do *Hadoop*, é o processamento de dados em memória (SALLOUM et al., 2016).

Isto é refletido em uma menor latência de I/O, ao mesmo tempo que tem a mesma capacidade de paralelismo e tolerância a falha do *MapReduce*. Durante o processamento, o *Spark* utiliza uma estrutura de dados denominada *Resilient Distributed Dataset* (RDD), ou em português, conjunto de dados distribuídos resilientes (SALLOUM et al., 2016). Esta estrutura permite que informações específicas sejam recuperadas do armazenamento e possibilita reconstruir os dados se o armazenamento subjacente falhar ("Hadoop vs. Spark", 2023).

O *Spark* não possui um sistema de armazenamento próprio, contudo é integrável com o HDFS, com *Object Storages* e com bancos NoSQL. Quanto ao desenvolvimento, possui APIs para linguagens de programação como *Scala*, *Java*, *Python* e SQL (SALLOUM et al., 2016). Destacam-se como casos de uso do Spark: análises em tempo real, *machine learning* e consultas interativas ("What is Apache Spark?", 2023).

Essas novas perspectivas de análise de dados contribuíram para a adoção do *Spark*, marcando uma nova geração do *Data Lake* (DEHGHANI, 2019). Entretanto, apesar dos ganhos em relação ao tempo de processamento do *Hadoop*, o *Spark* tem um custo maior de

escalabilidade devido ao uso mais intensivo de memória RAM. Dessa forma, são usuais arquiteturas que combinam o uso do *Hadoop* para cenários de processamento em lote e o uso do *Spark* para cenários de baixa latência e processamento tempo real ("Hadoop vs. Spark", 2023).

Fluxos de dados como este, onde há malhas com processamento em lote e outras com processamento em tempo real. Ou eventualmente, ambas estratégias combinadas, impõem maiores dificuldades para implementação e orquestração ("Apache Beam", 2023). Neste cenário, a Google desenvolveu e doou, em 2016, para a *Apache Software Foundation*, o *Beam* ("Apache Beam | Google Open Source Projects", 2023).

O *Apache Beam* é um modelo unificado de programação de esteiras de processamento de dados. Através do *Beam*, é possível criar um programa que define a esteira e, em seguida, executá-lo utilizando diversas plataformas como o *Spark*, *Google Cloud Dataflow* e *AWS Kinesis Data Analytics* ("Modelo de programação para o Apache Beam", 2023).

Os detalhes de baixo nível para execução da esteira como: coordenação de nós no *cluster*, fragmentação de dados e outras tarefas são orquestrados pelo *Apache Beam*. De modo que, o foco durante o desenvolvimento da esteira é a lógica de negócios e não requisitos de infraestrutura, o que facilita o desenvolvimento. Além disso, as esteiras podem ser alternadas de processamento em lote para tempo real, ou vice e versa, sem a necessidade de refatoração da lógica de negócios empregada ("Modelo de programação para o Apache Beam", 2023). Além da versatilidade de processamento, outro movimento que marcou uma nova geração de *Data Lakes* foi o denominado de *Delta Lake*. O qual foi apresentado em 2017 pela *Databricks (ARMBRUST et al., 2020)*.

2.3.6 Outras perspectivas para o Data Lake

A *Databricks* é uma empresa fundada em 2013 por membros do Projeto *AMPLab* da *University of California, Berkeley*, os quais criaram o *Spark* em 2009 ("About Databricks,

founded by the original creators of Apache Spark™", 2023). Em 2017, a empresa introduziu o *Delta Lake*, tornando-o *open source* em 2019 (ARMBRUST et al., 2020).

A proposta do *Delta Lake* originou-se de problemas de consistências em dados armazenados em *Object Storages*. Como já citado anteriormente, este tipo de armazenamento oferece escalabilidade, versatilidade e um baixo custo de armazenamento quando comparados aos bancos de dados. Geralmente, neste tipo de armazenamento, conjunto de dados relacionais são armazenados em arquivos colunares como *Parquet* ou *ORC* (ARMBRUST et al., 2020).

Estes formatos colunares propiciam um desempenho aceitável quando os dados são agrupados por algum atributo, como, por exemplo, um campo de data, de modo a otimizar consultas. Dentro dos *Object Storages,* este particionamento pode ser entendido como uma divisão em diretórios nos *buckets*. Isto é, uma tabela com um campo de data, pode ter seus dados separados em objetos que serão salvos no *bucket*. Dentro dos *bucket*, pode-se separar os objetos por diretórios contendo as datas dos dados, como: *minhatabela/data=2023-01-01/obj1* para dados do dia 1° de janeiro e assim respectivamente, para os demais dias. De modo que, *minhatabela* seria o nome do *bucket* e d*ata=2023-01-01* o nome do diretório, enquanto que *obj1* poderia ser algo como *meusdados1.parquet* ou *meusdados1.orc (ARMBRUST et al., 2020)*.

Outra característica dos arquivos colunares é o de permitir leituras por colunas, ao invés de buscas por todas as linhas. Soma-se a isso, o fato de terem um tamanho de arquivo menor que o equivalente no formato *csv*. Entretanto, este tipo de estratégia de dados implica em alguns desafios (ARMBRUST et al., 2020):

a) sem atomicidade em vários objetos: quando há operações que envolvem vários objetos, há o risco de clientes lerem resultados de operações parciais ou os dados ficarem corrompidos em caso de falha;
b) consistência eventual: os clientes podem acessar objetos atualizados e outros não após uma operação, mesmo que esta tenha sido bem sucedida;
c) baixo desempenho para consultas em múltiplos objetos: mesmo com a estratégia de particionar os objetos em diretórios, o desempenho para listar e

encontrar objetos através de consulta não é muito satisfatório. Além de terem custos atrelados que podem ser elevados;

d) poucas funcionalidades de gerenciamento: o que dificulta a gestão de dados e de ciclo de vida.

Mediante a isso, a proposta do *Delta Lake* é de trazer uma camada de transações ACID para o armazenamento de objetos. De modo que, suporte grande carga de trabalho e seja compatível com consultas: *ad-hoc*, em lotes e em *streaming* (ARMBRUST et al., 2020).

Para isso, o *Delta Lake* utiliza tabelas delta ou *Delta Tables*. Estas são, sucintamente, a composição de objetos no *Object Storage*, onde a camada de dados é armazenada em arquivos no formato *Parquet* com *logs* de operação de transação. As atualizações desses objetos pelos clientes são feitas utilizando protocolos de controle de simultaneidade otimistas, os quais foram adaptados para este tipo de armazenamento. Além disso, foram implementados níveis de isolamento de transações, de modo a assegurar os requisitos do ACID para estes objetos (ARMBRUST et al., 2020). Em termos práticos, em um *Data Lake* com arquivos *Parquet* por exemplo, não há necessidade de grandes adaptações técnicas para fazer a transição para as *Delta Tables*, sendo a principal mudança a captura dos *logs* de transação (ARMBRUST et al., 2021).

A escolha pelo *Parquet*, segundo os desenvolvedores da tecnologia, foi devido o formato ser *open source* e, assim, ser constantemente atualizado. Além de ter uma boa integração com o *Spark* (ARMBRUST et al., 2020).

Além do *Spark*, o *Delta Lake* é compatível com as principais tecnologias *open source* e privadas de armazenamento e processamento de *Big Data*, como: Apache Hive, Trino, AWS Athena, AWS Redshift, Google Cloud Storage e Azure ADLS. Além do ACID, o *Delta Lake* possui camadas de gerenciamento e de desempenho que oferecem recursos como:

a) *time travel* para controle de versão e reversão de estado;

b) operações de *UPSERT*, *DELETE* e *MERGE*;

c) *logs* de auditoria.

Além do *Delta Lake*, há estruturas *open sources* equivalentes. Um exemplo é o *Apache Iceberg*, cujo armazenamento de dados tem muita similaridade técnica com as *Delta*

Tables e foi, inicialmente, criado pelo Netflix (ARMBRUST et al., 2021). Esta nova geração de *Data Lakes* possibilitou uma nova abordagem para a Arquitetura de Dados até então estabelecida. A qual é denominada de Arquitetura de Duas Camadas, i.e., a primeira camada é o *Data Lake* e a segunda o *Data Warehouse*. A Databricks denominou esta abordagem de *Lakehouse (ARMBRUST et al., 2021)*.

2.3.7 Uma nova perspectiva para o Data Warehouse

O *Lakehouse*, também conhecido como *Data Lakehouse,* combina os principais recursos de *Data Warehouse* com os de *Data Lake*. Para isso, utiliza como base de armazenamento as *Delta Tables*, enquanto que para o processamento, utiliza tecnologias como o *Spark (ARMBRUST et al., 2021)*.

Além disso, há os motores de busca, ou *query engines,* os quais possibilitam acesso direto as camadas mais brutas, tornando mais fácil o transporte até a camada de BI. São exemplos, o AWS Athena que é um serviço gerenciado da Amazon, o Delta Engine da Databricks e o Trino que é *open source (ARMBRUST et al., 2021)*.

Este tipo de abordagem é decorrente da dificuldade de transporte e transformação dos dados do *Data Lake* para o *Data Warehouse* na Arquitetura de Duas Camadas. Onde há desafios técnicos e financeiros para se manter a consistência entre as camadas, assim como, manter os dados atualizados e suportar análises avançadas dos dados na camada do *Data Warehouse*. Implicando em esteiras complexas e na ineficiência na utilização dos dados do *Data Lake* (ARMBRUST et al., 2021).

Neste sentido, o *Data Lakehouse* oferece uma estrutura que propõe unir a flexibilidade e escalabilidade do *Data Lake* com o bom desempenho para consultas de dados e uma melhor Governança do *Data Warehouse*. Dessa forma, tem um foco na arquitetura de dados, enquanto que o *Data Warehouse*, o *Data Lake* e o *Delta Lake* têm abordagens baseadas na forma como os dados são gerenciados (BODE et al., 2023).

Apesar dos avanços tecnológicos em relação a forma que os dados são armazenados, processados e acessados, o *Delta Lake*, assim como o *Data Lakehouse,* estão sujeito aos mesmos desafios e limitações do *Data Lake* convencional. A razão é que isto, muitas vezes, engloba mais questões organizacionais e de modelo operacional do que de fato questões técnicas (TEKINER et al., 2023).

2.3.8 Desafios e Limitações do Data Lake

Um dos grandes desafios dessas estruturas de *Big Data* é que quanto maior o volume de dados, mais difícil é analisar e utilizar de maneira adequada estes dados. Uma das principais razões, é de que os dados relevantes acabam ficando ocultos em meio a enorme quantidade de dados não relevantes (INMON, 2016).

Outra questão está relacionada a qualidade e organização dos metadados. Os metadados são de fundamental importância para o entendimento dos dados, eles trazem descrições técnicas e de negócios, além de como os dados se relacionam. Em um *Data Lake*, uma vez que os dados são captados de forma bruta de diversas origens, é essencial que haja metadados precisos e acessível que contextualizem estes dados e possibilitem uma análise útil (INMON, 2016).

Dessa forma, é fundamental uma Governança de dados que promova a catalogação adequada dos metadados e defina critérios para assegurar a qualidade, acesso e privacidade destes. Além de determinar os responsáveis por eles, que seriam os *Data Owners*, ou em tradução literal, os Donos dos Dados. Outro fator determinante, é em relação ao ciclo de vida dos dados, onde é necessário ter padrões claros e bem estabelecidos sobre a coleta, tempo de armazenamento, utilização e expurgo (KHATRI; BROWN, 2010).

Todavia, apesar dos serviços em nuvem oferecem a descentralização física dos dados, através de replicações e distribuição de infraestrutura por regiões geográficas distintas. A estrutura lógica do *Data Lake* são de dados centralizados (DOLHOPOLOV; CASTELLTORT; LAURENT, 2023). Isto implica em desafios e limitações para implantação

de padrões de Governança de Dados, como por exemplo, para definir *Data Owners* ou assegurar a qualidade dos dados.

Esta questão está relacionada ao fato dos processos de dados serem centralizados na Engenharia de dados, cuja equipe torna-se responsável pelo desenvolvimento de esteiras que abrangem os mais diversos domínios de negócios e englobam diferentes tipos de análises (DOLHOPOLOV; CASTELLTORT; LAURENT, 2023). Isto requer que esta equipe tenha que conhecer regras de negócios e contexto de dados para realizar suas entregas. Neste cenário, há o risco da Engenharia de dados acabar responsável pela qualidade e propriedade de dados de outros domínios. Outro limitador é em relação a escalabilidade, em empresas maiores há o risco de haja gargalos no time de Engenharia de dados e demandas tenham que ser despriorizadas. Prejudicando assim, a inovação, geração de novos dados e afetando oportunidades de negócios (TEKINER et al., 2023).

Cabe ressaltar, que esta limitação de escalabilidade não é uma restrição tecnológica, mas sim operacional, o que difere das limitações sofridas pelo *Data Warehouse* antes do surgimento do *Data Lake*. Naquele contexto, as limitações de escalabilidade envolviam fatores operacionais e tecnológicos relevantes. Em meio a isso, uma nova abordagem descentralizada baseada em um novo modelo operacional com uma Governança Computacional Federada, foi articulada pela Zhamak Dehghani. Esta abordagem foi denominada de Data Mesh e foi apresentada em 2019 (TEKINER et al., 2023).

2.4 *O Data Mesh e a descentralização dos dados*

O *Data Mesh* é um contraponto para abordagens centralizadas como *Data Warehouse*, *Data Lake* e suas variações. Além disso, está inserido no mesmo contexto de dados analíticos com alta complexidade e em larga escala (DEHGHANI, 2022). Esta nova abordagem sugere ganhos em relação aos paradigmas anteriores. Destacando uma maior adaptabilidade as mudanças e volatilidades nas estratégias de negócios. Assim como, uma maior agilidade para

o crescimento empresarial e um aumento na geração de valor a partir dos dados (DEHGHANI, 2022).

Para isso, propõe alguns pontos de mudança em relação as abordagens anteriores. Estas mudanças são multidimensionais, isto é, englobam camadas técnicas e organizacionais. Sendo elas (DEHGHANI, 2022):

a) **organizacionais**: a propriedade dos dados antes centralizada em times especialistas como o de Engenharia de dados, passa a ser descentralizada e de responsabilidade dos domínios de negócios onde os dados são produzidos ou usados;
b) **arquiteturais**: os dados deixam de ficar hospedados em Data Warehouse ou Data Lakes e passam a ser oferecidos como produtos de dados por meio de uma malha distribuída cujo acesso segue protocolos padronizados;
c) **tecnológicas**: as soluções tecnológicas que antes abordavam os dados como um subproduto dentro de esteiras, passam a ter uma visão de uma unidade autônoma e dinâmica;
d) **operacionais**: a Governança deixa de ser centralizada e baseada em intervenções humanas para uma Governança Computacional Federada baseada em padrões e políticas automatizadas, contemplando de forma eficiente os nós da malha;
e) **concepcionais**: a abstração em relação aos dados muda, onde estes passam a ser considerados Produtos e os consumidores passam a ser tratados como Clientes. Logo, não é apenas mais algo para ser coletado e armazenado em um repositório, ganhando mais representatividade para os produtores e consumidores;
f) **infraestruturais**: deixa de ter a segmentação entre dados analíticos e operacionais para algo integrado dentro do domínio de negócios.

Estas mudanças não são associadas à uma tecnologia específica, como por exemplo foi o *Hadoop* e seu ecossistema para a arquitetura inicial do *Data Lake*. Deste modo, Dehghani pontua que o *Data Mesh* não se enquadra exatamente em uma arquitetura nova, uma lista de

princípios ou um novo modelo operacional. Segundo a autora, o *Data Mesh* é uma composição de cada ponto, sendo classificado como um paradigma sociotécnico (DEHGHANI, 2022).

Dessa forma, reconhece a interação entre as pessoas em conjunto com a arquitetura e soluções técnicas. Considerando para isso a complexidade das organizações, de modo que pretende ir além do gerenciamento de dados analíticos. Visando assim, uma melhor experiência organizacional, englobando os provedores de dados, usuários e proprietários. Neste sentido, segundo a autora, também pode ser utilizado como um elemento da estratégia de dados corporativos, articulando a arquitetura corporativa e o modelo operacional de forma interativa. Para isso utiliza como pilares quatro princípios:

a) Propriedade de Domínio (*Domain ownership*);
b) Dados como um Produto (*Data as a product*);
c) Plataforma de Dados com Autosserviço (*Self-Serve Data Platform*);
d) Governança Computacional Federada (*Federated Computational Governance*).

2.4.1 Propriedade de Domínio (Domain Ownership)

O princípio da Propriedade de Domínio é definido como a descentralização da propriedade dos dados analíticos para os seus respectivos Domínios de negócios. Isto é, a propriedade passa a ser de quem faz a captura dos dados operacionais e define as regras de negócios. Ou ainda, dos consumidores daqueles dados que aplicam novas regras e geram novos Produtos (DEHGHANI, 2022).

Dentro do Domínio de negócios, os dados tem todo seu ciclo de vida gerenciado de maneira independente. Contudo, alinhado com as diretrizes de Arquitetura e Tecnologia aplicadas na malha de dados. As motivações para a Propriedade de Domínio são (DEHGHANI, 2022):

a) compartilhamento de dados escalável: o que reflete na capacidade de aumentar a escalabilidade de acordo com as premissas organizacionais, implicando em:

mais de fontes de dados, mais consumidores e maior diversidade de casos de uso;

b) otimização continua: por estar sincronizado com as mudanças de negócios e sem amarrações externas, permitindo e propondo otimizações;
c) maior agilidade: pois há menos sincronizações entre times e não há dependências de times centralizados como o de Engenharia de dados.
d) aumento da veracidade de dados: isto é, remove a lacuna entre a origem dos dados e onde eles são utilizados de forma analítica. Reduzindo eventuais ruídos e facilitando modificações;
e) aumento da resiliência de soluções: de forma que os processos passam a ser mais diretos, removendo etapas intermediárias. Como por exemplo, esteiras de transformação entre as camadas de um *Data Lake*.

Tendo isso em vista, um dos pontos chave desse princípio é o de definir os Domínios de negócios. Deve-se determinar os contextos de dados existentes e avaliar como eles podem se integrar. Para isso, o *Data Mesh* é orientado pela estrutura de negócios, i.e., não é embasado por fronteiras estabelecida pelas soluções tecnológicas (DEHGHANI, 2022).

Isto é um contraponto do Modelo Operacional de estruturas como *Data Lakes* e *Data Warehouses*, onde a propriedade dos dados gira em torno das times que executam a tecnologia, como os times de *Data Warehouse* e de *Pipeline*. Dessa forma, todos os dados analíticos da organização ficam com a propriedade centralizada nestes times.

Segundo Dehghani, este tipo de estrutura não está de acordo com a estrutura organizacional dos negócios modernos, o que compromete o potencial de uso dos dados. Em virtude disso, para uma estrutura em que a Propriedade dos dados seja de seus respectivos domínios, a autora sugere a aplicação do *Domain-Driven Design*, o *DDD*, ou em português: Design Orientado a Domínio (DEHGHANI, 2022).

2.4.1.1 Definindo Domínios de dados utilizando o *DDD*

O *Domain-Driven Design* é um conceito de decomposição de design de software e alocação de times de acordo com a estrutura de negócios. De modo que, utiliza como base na modelagem do software, a linguagem utilizada por cada Domínio de negócios. Este conceito, foi apresentado por Eric Evans em 2003 (DEHGHANI, 2022).

O *DDD* se consolidou desde então, influenciando a modelagem organizacional e está em sinergia com o processo de digitalização dos negócios. Além disso, foi amplamente adotado pela Arquitetura de Software, sendo empregado na construção de microsserviços, decompondo sistemas grandes e complexos em serviços distribuídos fundamentados nos Domínios de negócios (DEHGHANI, 2022).

Em termos de estrutura, sua adoção resultou em equipes multidisciplinares de negócios e tecnologia responsáveis por Domínios específicos. Para o DDD, um Domínio é uma esfera de conhecimento, influência ou atividade, tendo um objetivo de negócios definido, para o qual os resultados são otimizados (DEHGHANI, 2022).

Embora os dados operacionais frequentemente já usem o *DDD*, no contexto analítico o mesmo não ocorre. Segundo Dehghani, no *Data Mesh* isso pode ser feito através do Design Estratégico proposto pelo *DDD* (*DDD's Strategic Design*), o qual trabalha com o conceito de Contexto Limitado.

Um Contexto Limitado refere-se a aplicabilidade de um modelo específico, tendo a equipe deste contexto a clara compreensão do que deve ser um consistente e o que pode ser desenvolvido de maneira independente de outros contextos. Para entender essas delimitações o DDD utiliza o mapeamento de contexto, que define explicitamente o relacionamento entre Contextos Limitados e Modelos Independentes (DEHGHANI, 2022).

No *Data Mesh* um Contexto Limitado seria um Produto de Dados, o que engloba os dados, seus modelos e sua propriedade. Em organizações que já utilizam microsserviços ou Arquitetura Orientada a Domínios, a extensão para definir os Domínios de Dados é mais simples. Neste caso, seria aplicar a mesma modelagem e decomposição utilizada para os dados operacionais nos dados analíticos (DEHGHANI, 2022).

Dehghani apresenta três tipos principais de dados que podem ser gerados a partir dos Domínios de Dados definidos (DEHGHANI, 2022):

a) dados analíticos (nativos) alinhados a origem: os quais são criados e compartilhados pelas equipes proprietárias dos sistemas de origem;
b) dados analíticos agregados: Quando entre um consumidor e um produtor ocorre uma agregação dos dados analíticos, originando um novo Produto de Dados;
c) dados analíticos alinhados ao consumidor: Os quais são adequados a uma finalidade de uso, isto é, projetados para um consumo de dados específico.

Conforme exposto, dentro dos domínios, os dados são tratados como Produtos. Sendo este outro princípio do *Data Mesh* (DEHGHANI, 2022).

2.4.2 Dados como um Produto (Data as a Product)

Um dos riscos ao se descentralizar os dados é o de criar silos e, por consequência, impossibilitar o funcionamento do *Data Mesh*. Em virtude disso, Dehghani apresentou o princípio de Dados como um Produto (DEHGHANI, 2022).

Baseando-se no livro de Marty Cagan, definiu que um novo Produto de Dados é resultado da intersecção de três características: Ser viável, Dê valor e Seja Utilizável. Contudo, para se ter um novo Produto de Dados, algumas premissas devem ser consideradas. Sendo elas (DEHGHANI, 2022):

a) encontrável: um Produto de Dados deve ser fácil de ser descoberto e, além disso, deve possibilitar acompanhar sua evolução e relação com outros produtos;
b) endereçável: deve oferecer um endereço permanente e exclusivo para os consumidores do Produto de Dados acessá-lo de forma programática ou manual;

c) compreensível: após a descoberta, é preciso que os consumidores do Produto de Dados consigam entender de maneira eficiente a semântica e o contexto daquele Produto. Evitando assim, o uso incorreto;
d) confiável e verdadeiro: para isso pode oferecer algumas métricas e fluxos para os consumidores, como por exemplo: a linhagem dos dados, a frequência de atualização e o nível de confiança de determinado indicador;
e) acessível nativamente: deve possibilitar as diversidades de acesso seguindo os perfis de consumidores finais, i.e., para consumidores que precisem consumir através de consulta SQL deve haver essa opção. Da mesma forma, para atender os consumidores que precisarem do Produto de Dados no formato de arquivo e assim por diante;
f) interoperável e combinável: para possibilitar correlações de maneira eficiente, assim como, ser integrado a outros Domínios. Além disso, um Produto de Dados deve seguir padrões corporativos de esquema de dados, semântica, entre outros;
g) valor independente: avaliar se aquele Produto de Dados de fato agregará valor, isto é, se representará ganhos operacionais ou estratégicos. Um Produto de Dados requer grande estruturação, logo é imprescindível investir de maneira calculada;
h) seguro: devem ser aplicadas políticas de acessos para que apenas os consumidores qualificados acessem o Produto de Dados. As políticas de acesso podem funcionar de maneira centralizada, mas sua concessão pode ser feita em tempo de acesso, por exemplo, baseando-se no perfil do consumidor.

Segundo Dehghani, os Dados como um Produto introduzem uma nova unidade de Arquitetura Lógica denominada *Data Quantum*. Caracterizada por controlar e encapsular todos os componentes estruturais necessários para compartilhar Dados como Produto. Sendo eles: Metadados, Códigos, Política e Declaração de dependência de infraestrutura de forma autônoma (DEHGHANI, 2022).

Este princípio, além de evitar silos como citado anteriormente, pretende estimular uma cultura de inovação baseada em dados, de modo a simplificar a descoberta e melhorar a qualidade dos dados analíticos. Assim como, estabelecer a abordagem de contato de dados (DEHGHANI, 2022).

Os contratos de dados são acordos firmados entre produtores e consumidores dos Produtos de Dados. O intuito deles, é que as alterações não gerem inconsistências e instabilidade no *Data Mesh*. Trabalhar com Produto de Dados também implica em novos papéis, Dehghani destaca os seguintes (DEHGHANI, 2022):

a) Proprietário do Produto de Dados (*Data Product Owner*): este papel requer conhecimento profundo dos dados e do domínio no qual estão inseridos, assim como, dos consumidores. De modo que, zele pelas métricas de qualidade e de sucesso. Além de, melhorar o tempo de entrega, refinar o Produto de Dados e satisfazer as necessidades dos consumidores;
b) Desenvolvedor do Produto de Dados (*Data Product Developer*): esta função constrói, mantém e atende os Produtos de Dados do Domínio, sendo pares dos Desenvolvedores de Aplicação do Domínio.

Uma equipe pode atender um ou mais Produto de Dados. Além de, existir a possibilidade de ser necessário criar uma nova equipe caso não haja um Domínio de negócios adequado. Para disponibilização de um Produto de Dados é necessário utilizar uma plataforma, que é o próximo princípio a ser abordado (DEHGHANI, 2022).

2.4.3 Plataforma de dados com Autosserviço (Self-Serve Data Platform)

Para viabilizar os princípios anteriores e possibilitar a adoção do *Data Mesh*, é necessário que os Domínios consigam criar seus Produtos de Dados e os consumidores consigam acessá-los. Neste contexto, deve haver uma padronização na malha para que ela seja funcional, eficiente e confiável (DEHGHANI, 2022).

Dehghani propõe que isto seja feito a partir de uma Plataforma de dados. A qual deve ser agnóstica ao Domínio, i.e., permita sua utilização independente dos Produtos de Dados que serão disponibilizados. Além disso, está Plataforma deve ter automação e ser baseada em autosserviço para evitar gargalos como os que ocorrem em plataformas centralizadas (DEHGHANI, 2022).

Além dos gargalos, há também a barreira técnica e tecnológica, de modo que a Plataforma proposta no Data Mesh visa remover essas fricções e evitar que os Domínios tenham que se concentrar em requisitos de baixo nível como: configuração de redes, infraestrutura, segurança e esteiras (DEHGHANI, 2022).

A proposta que Dehghani apresenta é que a Plataforma construa, sobretudo, uma boa experiência de uso. De modo que, viabilize a descoberta de dados pelo lado dos consumidores, assim como, facilite o consumo destes, por exemplo utilizando APIs. Além disso, ofereça escalabilidade para atender o aumento da malha e demandas de consumo (DEHGHANI, 2022).

Já pelo lado do produtor, a Plataforma deve facilitar a catalogação dos dados, a disponibilização, o processamento, entre outras necessidades. Neste contexto, a automação evita erros humanos, aumenta a escalabilidade e torna o processo mais eficiente (DEHGHANI, 2022).

Dehghani também indica que esta estratégia:

a) reduz o custo total da propriedade descentralizada dos dados, habilitando maior previsibilidade;
b) possibilita que as equipes se concentrem no ciclo de vida e melhoria da qualidade dos dados, melhorando os Produtos entregues;
c) mobiliza mais desenvolvedores para os Produtos de dados, uma vez que reduz a necessidade de especialização;
d) automatiza as políticas de Governança para criar padrões de segurança e conformidade para todos os produtos de dados.

Neste sentido, tem sinergia com o próximo princípio que é o de Governança Computacional Federada.

2.4.4 Governança Computacional Federada (Federated Computational Governance)

A proposta da Governança Computacional no *Data Mesh* é de garantir a autonomia de maneira segura e em conformidade com as diretrizes regulatórias e da Organização. Para isso, apresenta um Modelo Operacional com base em uma estrutura federada para tomada de decisão e atribuição de responsabilidade (DEHGHANI, 2022).

Dessa forma, propõe uma equipe composta por representantes dos times de Domínio, Plataforma de Dados, assim como times especialistas, e.g., Jurídico, Segurança e *Compliance*. Esta equipe tem o papel de viabilizar a interoperabilidade do Data Mesh, definindo padrões que posteriormente serão executadas através de automação na malha (DEHGHANI, 2022).

Esta abordagem visa não ter intervenções humanas frequentes e processos que dificultem a geração de valor através dos dados. Cenário característico do modelo operacional da Governança convencional. Em contrapartida, institui a automação das políticas diretamente nos Produtos de Dados, equilibrando autonomia, agilidade e responsabilidade. Além dos Produtos de Dados, são considerados ainda padrões globais que envolvem todo o Data Mesh da organização (DEHGHANI, 2022).

Os benefícios de uma Governança Computacional Federada, segundo Dehghani, são (DEHGHANI, 2022):

a) mais interoperabilidade: permite aumentar o nível de agregação e correlação entre os Produtos de Dados, enriquecendo a malha;
b) menos incompatibilidade: estabelece elo entre os Domínios, reduzindo incompatibilidades e integrando melhor a malha;
c) aumenta a viabilidade de políticas: permite que as políticas como de Governança, Segurança e *Compliance*, sejam aplicadas de maneira efetiva por todo o *Data Mesh*. E com isso, favorece a diminuição de riscos;
d) menos tratativas manuais: reduz a necessidade de intervenções constantes e sobrecarga de sincronização dos domínios.

O princípio da Governança Computacional Federada, assim como, os demais princípios são determinantes para o correto funcionamento do *Data Mesh*. A seção seguinte apresenta como eles estão conectados.

2.4.5 Interação dos Princípios

Delghani enfatiza que os quatro princípios do Data Mesh foram propostos para serem suficientes e complementares. Além disso, cada um envolve desafios próprios para serem corretamente implementados (DEHGHANI, 2022). Entretanto, a implementação de um princípio pode gerar desafios para a implementação de outros. Isto fica mais evidente caso seja considerado o princípio de Propriedade de Domínio (*Domain Ownership*) (DEHGHANI, 2022).

Uma vez implementada a Propriedade de Domínio, espera-se que o princípio de Dados como um Produto (*Data as a Product*) seja implementado e evite que silos sejam criados. Já do princípio de Plataforma de Dados com Autosserviço (*Self-Serve Data Platform*), espera-se que este capacite as equipes de domínio e efetive sua utilização. Enquanto que, para o princípio de Governança Computacional Federada (*Federated Computational Governance*), espera-se que este deverá aumentar o engajamento e reduzir o isolamento do Domínio (DEHGHANI, 2022).

Além disso, Dados como um Produto dependerá que o princípio de Plataforma de Dados com Autosserviço reduza o custo de propriedade, para que assim, aumente a viabilidade do Produto. Também dependerá dos padrões estabelecidos pela Governança Computacional Federada para se conectar aos demais Produtos de Dados (DEHGHANI, 2022).

Já a Governança Computacional Federada, dependerá que o princípio de Plataforma de Dados com Autosserviço aplique de maneira consistente e confiável as políticas definidas para o *Data Mesh*.

Em um cenário ideal, caso os princípios consigam superar os desafios de implementação e atendam os requisitos propostos, o *Data Mesh* é estabelecido e os dados passam a ser democratizados, escaláveis e, logo, a organização passa a ser mais orientada por eles (BODE et al., 2023).

Além do *Data Mesh*, uma abordagem denominada *Data Fabric* está em ascensão. Assim como o primeiro, objetiva por uma organização mais orientada a dados e redução de gargalos (BODE et al., 2023).

2.4.6 Comparação com o Data Fabric

Assim como o *Data Mesh*, o *Data Fabric* ("Data Mesh vs Data Fabric", 2023) também é uma malha de dados. Entretanto, o *Data Fabric* tem um viés mais tecnológico do que organizacional. Ele é baseado em um modelo central para Governança e Catalogação de Dados, o qual é o otimizado para fazer varreduras e correlações entre as origens de dados da organização (BODE et al., 2023).

Para isso, utiliza recursos como *Artificial Intelligence* e *Machine Learning* ("Data Mesh vs Data Fabric", 2023). Como o *Data Fabric* não é uma abordagem organizacional, assim não engloba um novo Modelo Operacional para estruturação de Produtos de Dados como ocorre no *Data Mesh* (BODE et al., 2023).

Contudo, ambos podem ser aplicados em conjunto, apesar de serem concebidos para atuar de maneira independente. O uso em conjunto pode ser complementar, onde através do *Data Fabric* é possível avaliar novos Produtos de Dados ou revisar correlações no *Data Mesh* (BODE et al., 2023).

Entretanto, para ambas as abordagens ainda há poucos casos empíricos e, baseado nisso, a próxima seção traz alguns pontos que devem ser observados em relação ao *Data Mesh*.

2.4.7 Desafios e Limitações do Data Mesh

Por ser uma abordagem relativamente recente, não há muitos casos de uso do *Data Mesh* ou publicações empíricas sobre o tema. Baseados nisso, Bode, Kühl, Kreuzberger e Hirschl publicaram o artigo Data Mesh: Motivational Factors, Challenges, and Best Practices, com algumas entrevistas sobre a adoção do *Data Mesh* pelas Organizações (BODE et al., 2023).

Ao todo foram 15 entrevistas, entre novembro de 2022 e janeiro de 2023, feitas com especialistas e gestores. No geral, a maioria concorda com os princípios apresentados por Dehghani para a implantação do *Data Mesh*. Contudo, atribuem um valor maior aos princípios de *Domain Ownership* e *Data as a Product (BODE et al., 2023)*.

Os principais fatores que motivam a adoção de um *Data Mesh* incluem (BODE et al., 2023):

a) a possibilidade de reduzir gargalos;
b) aproveitar o conhecimento de dados e de negócios de cada Domínio;
c) quebrar silos, isto é, dados isolados dos demais sistemas corporativos (“Silos de dados”, 2023);
d) estabelecer a Propriedade dos dados: No sentido de não ter mais um time centralizado com a Propriedade dos dados analíticos corporativos, para um modelo onde cada Domínio tenha Propriedade sobre seus Produtos de dados;
e) adotar uma arquitetura moderna;
f) reduzir redundâncias.

O principal desafio para a adoção de um *Data Mesh* é a dificuldade de implementar o conceito de Governança Computacional Federada. Uma vez que há desafios relacionados ao desconhecimento de dados sensíveis dentro dos Domínios e dificuldades relacionadas a automação das políticas e observabilidade da conformidade pelos Produtos de Dados (BODE et al., 2023).

Outro fator, é a mudança de propriedade dos dados. Na visão dos entrevistados, as novas responsabilidades implicam em um volume maior de trabalho e novas demandas. Além

de que, as atividades comerciais dos domínios não são de fornecimento de dados, o que gera a não priorização em relação a isso (BODE et al., 2023).

Há também, a dificuldade em relação à Qualidade de Metadados, isto é, os descritivos das estruturas de dados não são preenchidos corretamente, o que dificulta o entendimento de como usar e relacionar os dados. Em conjunto há a Falta de Padrão de Modelagem que dificulta integrações e prejudicam o desenvolvimento. Além disso, há a dificuldade de Compreensão de como o *Data Mesh* funciona, fazendo com que haja, por exemplo, a adoção equivocada de um Produto de Dados, às vezes denominando de Produto um dado altamente centralizado (BODE et al., 2023).

Destaca-se também as Limitações de recursos, sejam eles financeiros, técnicos ou humanos. Assim como, Problemas de aceitação e resistência (BODE et al., 2023). De modo geral, é necessário avaliar as transformações necessárias e a adaptabilidade do *Data Mesh* à estrutura organizacional. Para isso, cabe ponderar questões como capacitação, estrutura tecnológica e período de convivência.

3 APLICAÇÃO

Tendo a abordagem apresentada por Dehghani para o *Data Mesh* (DEHGHANI, 2022), esboçou-se como este conceito seria aplicado em um cenário real. Para isso, uma fonte de inspiração foi utilizar os exemplos que Dehghani ilustra através da *Daff, Inc* (DEHGHANI, 2022). A *Daff* é uma companhia ficcional de *streaming* de áudio e música. Dehghani, utiliza os Domínios de negócios desta empresa, assim como, as possíveis interações dos usuários, para apresentar estratégias de aplicabilidade do *Data Mesh*. Uma das características da *Daff* é que ela pode ser segmentada por diversos temas e, logo, produzir uma grande variedade de dados e de Produtos de Dados. Sendo assim, um bom plano de fundo para exercitar o *Data Mesh* (DEHGHANI, 2022).

Devido a abordagem sociotécnica do *Data Mesh*, Dehghani aborda de forma mais conceitual as vertentes do paradigma. Esboçando possíveis estratégias de implementação, sem ter o objetivo de esgotar o tema ou apresentar diretrizes de soluções técnicas, i.e., relacionas a aplicação de alguma tecnologia específica ou serviço de *Cloud Provider* (DEHGHANI, 2022).

A partir disso, nesta seção aplicada deste Projeto, avaliou-se como seguir em uma linha semelhante. Com a diferença de realizar a implementação de partes chaves, utilizando algumas tecnologias *open source* que já são consolidadas no contexto de dados e *Big Data*.

Além da parte implementada, ao final, pretende-se elocubrar, fundamentado no que foi apresentado por Dehghani, sobre possíveis estratégias e melhorias complementares para, de fato, obter-se um *Data Mesh*. Assim como, coletar os aprendizados e pontos de atenção que serão apresentados na Conclusão.

Mediante a isso, implementou-se um site fictício de comércio eletrônico denominado Solarié, o qual é especializado na venda de óculos. Em termos de infraestrutura, o Solarié pode ser dividido em três camadas principais: a *Interface Web* (*Frontend*), o Servidor com Motor de Regras, Rotas e *Interface* de dados (*Backend*) e a Camada de dados composta por bancos de dados e armazenamento de objetos (*Object Storage*).

Como camadas secundárias, isto é, que permeiam estas camadas já citadas. Destacam-se o Catálogo de dados, o Integrador de dados (Query Engine) e o Orquestrador de dados. Este plano de fundo permite, em termos de solução técnica, ilustrar em parte o funcionamento do *Data Mesh*.

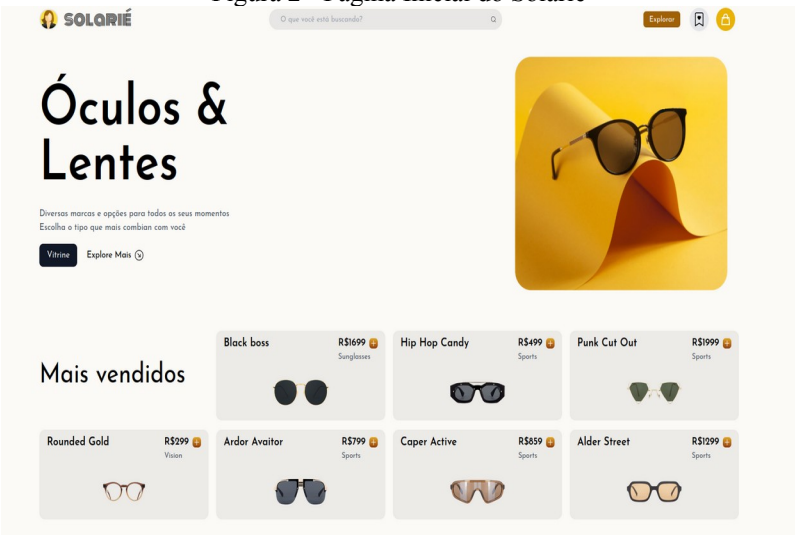
Uma vez que o cerne do Projeto é a arquitetura de dados, mais precisamente a abordagem do *Data Mesh*, utilizou-se *templates* tanto para o *Frontend* quanto para o *Backend*, ambos disponibilizados através de repositórios do *Github*. Tais *templates*, precisaram ser adaptados para integrar-se a Camada de dados, assim como, gerar novos tipos de dados e atender à outras necessidades gerais do Projeto.

O *template* do *Frontend* é o *Eyesome*, o qual foi desenvolvido pela usuária do *Github* Sandhya1007 (SANDHYAR1007, 2023). Este *template* foi refatorado e renomeado de Solarié. Além disso, foi traduzido, em praticamente toda sua totalidade, de Inglês para Português. Assim como, a moeda foi adaptada para o Real brasileiro. O que é ilustrado nas Fig. 1 e Fig. 2 :

Figura 1 - Página Inicial do Template Eyesome

Fonte: Sandhya1007, 2023.

Figura 2 - Página Inicial do Solarié



Fonte: Renan Ferreira Lima, 2023.

Também inclui-se o campo de data de nascimento e gênero na página de cadastro de cliente. A proposta foi de ter mais dados para realizar estudos analíticos posteriores. Cabe ressaltar, que o *template* em sua versão original, possui um emulador de *Backend*. O qual emula os motores de regra e os bancos de dados, entretanto não oferece um exemplo funcional do *site*.

Desta forma, o *template* tinha muitos recursos que não operavam conforme esperado para fins do Projeto ou de ilustração de um *Site* de Comércio Eletrônico. Como exemplo, tem-se o Sistema de Cadastro de Novos Clientes, ou então, a Sacola de Compras, ambos não estavam funcionando corretamente. Isto ocorre, pois como não havia o *Backend*. De modo que, todos os dados e regras da página ficavam armazenados no *Frontend*, o que limita a implantação de recursos.

Para contornar isto, utilizou-se um *template* de *Backend*, o qual está disponível no Github pelo o usuário *collegewap* e é denominado *mern-auth-server* (COLLEGEWAP, 2023). Os detalhes de como funciona este *template* estão disponíveis no *site* *CodingDeft* (“Authentication in React”, 2023). Em linhas gerais, este *template* implanta um sistema de autenticação de usuário, utilizando a estratégia de *JWT* para geração de *token* (“Authentication in React”, 2023).

Assim, não armazena senha de usuário no banco de dados, apenas o *token*. O qual é utilizado na autenticação. Além da autenticação, o *template* também oferece regras para Cadastro de Usuários, definindo o esquema de dados.

Em termos de estrutura, tanto o escopo de autenticação quanto de cadastro, são tratados como rotas. As rotas indicam quais regras devem ser aplicadas quando o usuário executa alguma ação no *Frontend*. Por exemplo, caso o usuário queira efetuar um cadastro, o *Frontend* o direcionará para uma rota com o final */signup*. De modo que, no *Backend* os dados resultantes desta interação serão tratados pelas regras do /signup e, posteriormente, persistidos no banco de dados, retornando uma mensagem de sucesso para o *Frontend* ou uma mensagem de falha caso haja eventuais erros, instabilidade e incompatibilidades.

Como já mencionado, o *template* do *Backend* continha uma rota de autenticação e cadastro. Entretanto, estas não estavam compatíveis com os esquemas de dados definidos para o *Frontend*, assim como, com os padrões de mensagens esperadas em caso de sucesso ou falha. Deste modo, foi feita uma refatoração para adequar o *Backend* aos requisitos do *Frontend*. Durante este processo, também refatorou-se algumas estruturas de dados do *Frontend* para adequá-las as estratégias de autenticação *JWT*. A partir de então, configurou-se o banco de dados para receber os dados gerados no *Frontend* e processados no *Backend*. No caso, este banco recebe os dados de cadastro de usuários/clientes do Solarié.

Seguindo esta estratégia, implementou-se as rotas para os produtos e categorias de produtos exibidos no *Frontend*. Neste caso, além de dados como nome e descrição de produtos, há também as imagens dos produtos. No *template* do *Frontend*, estas imagens estavam armazenadas em pastas e eram carregadas quando o código iniciava. Para o Solarié, essas imagens foram armazenadas em *Object Storage* e os dados dos produtos no banco de dados.

Assim, no *Backend*, criou-se as rotas para retornar estes dados, juntamente com o *endpoint* do diretório de cada imagem associada armazenada no *Object Storage*. De modo que, ao receber este retorno, o *Frontend* consegue apresentar os produtos na tela, em tempo semelhantes ao *template* original.

A proposta de utilizar este *Object Storage*, é que além de armazenar este tipo de dados não estruturados, ele servirá de base para fornecer alguns dados analíticos. Neste cenário, esses dados serão acessados através do *Query Engine* implantado.

Retomando as alterações e implementações no *Backend*, também criou-se rotas para armazenar os dados de Sacola de compras e Lista de Favoritos do cliente. No caso da Sacola, ela é apresentada para o cliente como *Bag* e tratada no *Backend* como C*art*, enquanto que a Lista de Favoritos é tratada no *Backend* como *Wishlist*. Estas terminologias, seguem o padrão do *template* original do *Frontend*, não tendo impactos significativos na demonstração.

Cabe ressaltar, que sempre que é criada uma rota e esta depende de um banco de dados para persistir os dados recebidos, i.e., dados cadastrais, Lista de Favoritos e Sacola de Compras. Então, é necessário criar no *Backend* esquemas de como estes dados serão persistidos no banco de dados. Dessa forma, criou-se esquemas para cada um desses temas.

Além disso, para o caso da descrição dos Produtos e da Categoria de Produtos, onde o *Backend* apenas lê os dados do banco de dados, também foi necessário criar esses esquemas. Com isso, gerou-se a infraestrutura para o correto funcionamento do Solarié, onde o *Frontend* recebe e envia dados para o *Backend* e este, por sua vez, aplica o motor de regras através das rotas criadas, consultando e gravando nos bancos de dados e Object Storage. E por fim, retorna uma mensagem de sucesso ou falha, junto com os respectivos dados necessários, para o *Frontend*.

A segunda etapa do Projeto foi implantar o Catálogo de dados e integrá-lo ao Solarié, de modo a catalogar os dados operacionais. Isto é, conectou-se o Catálogo de dados aos bancos de dados para trazer os metadados de Clientes, Produtos, Categorias, Sacola de Compras e Lista de Favoritos.

Dessa forma, no Catálogo há o esquema dos dados destes bancos, assim como, a descrição de cada campo e o Domínio de negócios. Na seção Tecnologias Utilizadas, há uma figura da *Interface* deste Catálogo.

Nesta etapa, também foram definidos os Domínios de negócios, e parte deles foi adicionada ao Catálogo. Os Domínios definidos foram:

a) Interface Web: Domínio responsável pelo Frontend, isto é, gerenciamento do site Solarié. Nesta demonstração, este Domínio não tem dados para serem catalogados;
b) Plataforma: Domínio responsável pelo Backend, integrando os dados do site Solarié com os demais do Domínios. Nesta demonstração, este Domínio não tem dados para serem catalogados;
c) Vendas: Responsável pelos dados da sacola de compras e da lista de favoritos, além da quantidade e demais dados de vendas. Este domínio pode criar Produtos de dados que indiquem tendências de compras, assim como, produtos mais comprados, produtos mais favoritados, dias e horários mais propícios para campanhas e promoções;
d) Produtos: Responsável por Dados Cadastrais de Produtos, como inclusão e remoção de produtos, assim como, de categorias de produtos. Em termos de Produtos de Dados, estes podem ser criados tendo indicadores como categorias e variedades de produtos, produtos por gênero, produtos por marcas, novos produtos e produtos mais antigos;
e) Cadastro de Clientes: Domínio responsável pela gestão de dados de clientes. Podendo fornecer Produtos de Dados que indiquem qual é a idade média dos clientes, como é a distribuição por gênero e quantidade de novos clientes;
f) Gestão de Acesso: Domínio responsável por criar as regras de segurança para autenticação dos clientes na plataforma. Nesta demonstração, este Domínio não tem dados para serem catalogados;
g) Plataforma de Dados: Domínio responsável por criar as esteiras de automação para catalogar os dados e disponibilizar os Produtos de Dados. Nesta demonstração, este Domínio não tem dados para serem catalogados;
h) Gestão de Pagamentos: Domínio responsável por gerenciar os meios de pagamentos utilizados pelos clientes. Este domínio não foi implementado;
i) Gestão de Pedidos: Domínio que gerencia a logística e dados dos pedidos efetuados. Este domínio não foi implementado.

Após a conclusão desta etapa, criou-se um Produto de Dados a partir dos dados operacionais de Clientes. Isto é, construiu-se uma visão estratificada da população de clientes com agrupamentos por gênero e ano de nascimento, disponibilizando, em seguida, estes dados no diretório do Domínio no *Object Storage*. A opção de utilizar o *Object Storage* foi por ser um armazenamento escalável e de baixo custo. Entretanto, este Produto de Dados poderia estar armazenado em um banco de dados.

O Produto de Dados foi então catalogado, podendo ser descoberto a partir do Catálogo de dados. A disponibilização foi feita a partir do *Query Engine*, que permite realizar consultas SQL em diversos formatos de dados, inclusive em arquivos em um *Object Storage*.

A fim de ter uma massa maior de dados de clientes para o Projeto, gerou-se um código para criar clientes fictícios e randômicos e, em seguida, persisti-los no banco de dados. Este código é executado, no momento de compilação do Projeto e, no caso, foi configurado para gerar 10.000 clientes.

Outro Produto de Dados de dados criado foi para o Domínio de Produtos, onde seguiu-se a mesma estratégia do Domínio de Clientes. Criando alguns indicadores sobre dados de Produtos e disponibilizando no *Object Storage*. Estes indicadores são: Produtos por Categoria, por Marca, por Gênero, entre outros. Este Produto de Dados também foi catalogado e ficou disponível para descoberta.

Por fim, criou-se mais um Produto de Dados, desta vez para o Domínio de Vendas. Para isso pegou-se dados da Lista de Favoritos dos Clientes e criou-se alguns indicadores sobre a categoria, marca e gênero de produto mais frequentes. Neste caso, ao contrário dos demais, o Produto de Dados foi disponibilizado em um banco de dados. Mais uma vez, catalogou-se o Produto para possibilitar sua descoberta. Apesar deste Produto estar em um banco de dados, ele também é consultado através do *Query Engine* pelos consumidores.

Quanto ao processamento, isto é, as agregações para gerar os Produtos de Dados, em todos os cenários é feito através do Orquestrador de dados. Em relação a execução do Projeto, ela é feita no ambiente local de desenvolvimento utilizando contêineres. Mas poderia ser executado em um ambiente em nuvem, como por exemplo, na AWS, GCP ou Azure.

A seção seguinte detalha as tecnologias utilizadas e traz algumas figuras que mostram o Projeto em execução. O material desenvolvido está disponível no repositório do Github e pode ser acessado através do link: ***https://github.com/lima-renan/solarie*** .

3.1 *Tecnologias utilizadas*

Nesta seção, o objetivo é apresentar as tecnologias utilizadas e o que motivou a sua escolha. Inicialmente, será apresentada infraestrutura principal e, em seguida, os componentes que a integram.

3.1.1 Frontend

O *Frontend* foi feito em *React/Javascript* (“React”, 2023), o *framework* e a linguagem aqui não tem implicação direta com o resultado do Projeto. Dessa forma, poderia ter sido feito em outra linguagem.

Entretanto, devido a vasta quantidade de material e documentação disponível, assim como, ser um *framework* frequentemente utilizado para desenvolvimento *Frontend*, optou-se por esta tecnologia. Dessa forma, buscou-se um *template* que atendesse estes requisitos.

3.1.2 Backend

Visando ter um padrão de desenvolvimento, sem ter muitas linguagens de programação distintas, optou-se em utilizar o *Nodejs* (“Node.js”, 2023) no *Backend*. Da mesma forma que para o *Frontend*, a linguagem utilizada aqui não tem impacto nos resultados esperados na demonstração.

3.1.3 Banco de dados

O banco de dados escolhido para suportar o *Backend* foi o MongoDB (“MongoDB”, 2023). A escolha pelo MongoDB foi pelo fato de ser um banco *NoSQL* de documentos, o que traz mais flexibilidade para alterações de esquemas e redefinições de modelagem de dados. Isto facilitou nos momentos de refatoração, uma vez que não foi necessário despender muito enfoque para a modelagem e remodelagem. Na Fig. 3, há um exemplo de documento armazenado no MongoDB. No caso, é referente ao Cadastro de Cliente.

Figura 3 - Documento com Dados de Clientes no MongoDB

Fonte: Renan Ferreira Lima, 2023.

3.1.4 Object Storage

Para armazenar as imagens dos Produtos e da Categoria de Produtos, as quais estão no formato *png*, utilizou-se o Minio (Fig. 4) (“MinIO”, 2023). A escolha foi por ser uma tecnologia compatível com o AWS S3, que é um *Object Storag*e amplamente utilizado em *Big Data*. Logo, é mais propicio para realizar as integrações necessárias.

Figura 4 - Diretório de Produtos do Minio

Fonte: Renan Ferreira Lima, 2023.

Além das imagens, utilizou-se o Minio para disponibilizar os Produtos de Dados no formato *json*. A escolha pelo formato foi pela similaridade com a estrutura do MongoDB, o que deu mais agilidade no desenvolvimento. Entretanto, outro formato poderia ser utilizado, como: Parquet, Avro, ORC e csv.

3.1.5 Banco de Dados – Produto de Vendas

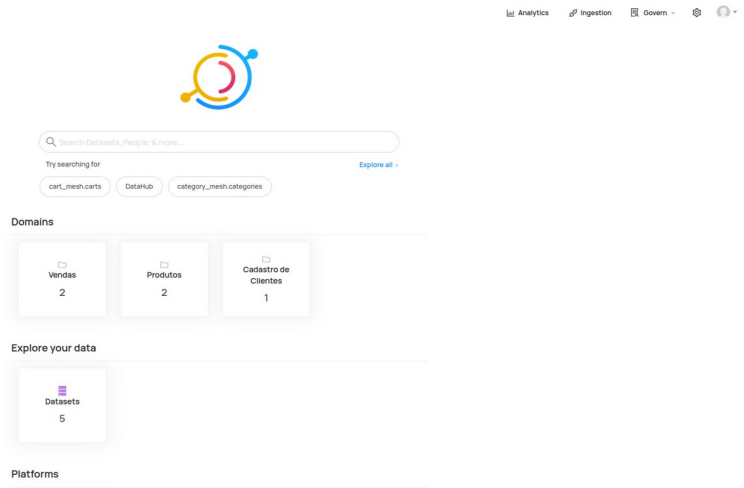
Para ilustrar um outro cenário de uso, utilizou-se o banco SQL PostgreSQL (“PostgreSQL”, 2023). Este é amplamente utilizado no contexto de dados. Aqui também poderiam ser utilizados outros bancos como: MySQL, MariaDB, Oracle ou SQLServer.

3.1.6 Catálogo de dados

Utilizou-se o DataHub (“A Metadata Platform for the Modern Data Stack”, 2023) como catálogo de dados. A escolha foi por ser uma tecnologia que oferece uma boa experiência de uso e tem uma quantidade expressiva de recursos. Entretanto, outras tecnologias também atenderiam bem como: o Amudsen, Atlas ou o OpenMetada.

Para o *deploy*, utilizou-se a opção via arquivo Docker Compose customizado (“Deploying with Docker”, 2023). Mais detalhes sobre o uso de contêineres na seção de Infraestrutura. A Fig. 5, mostra a interface do Catálogo.

Figura 5 - Interface do Catálogo de dados



Fonte: Renan Ferreira Lima, 2023.

3.1.7 Orquestrador de dados

Para simular uma Plataforma de dados, mesmo que de forma superficial, utilizou-se o Apache Airflow (“Apache Airflow”, 2023). A escolha pelo Airflow foi pela grande quantidade de conectores e de materiais disponíveis, assim como, por ser amplamente utilizado em *Big Data*.

3.1.8 Integrador de dados

Como Integrador de dados, ou *Query Engine* (“Distributed SQL query engine for big data”, 2023), utilizou-se o Trino. O qual pode ser integrado as origens de dados utilizadas no Projeto.

3.1.9 Infraestrutura

O Projeto pode ser executado em ambiente local, uma vez que foi desenvolvido utilizando contêineres Docker (“Docker”, 2022). Que são orquestrados utilizando o Docker Compose (“Docker Compose overview”, 2023).

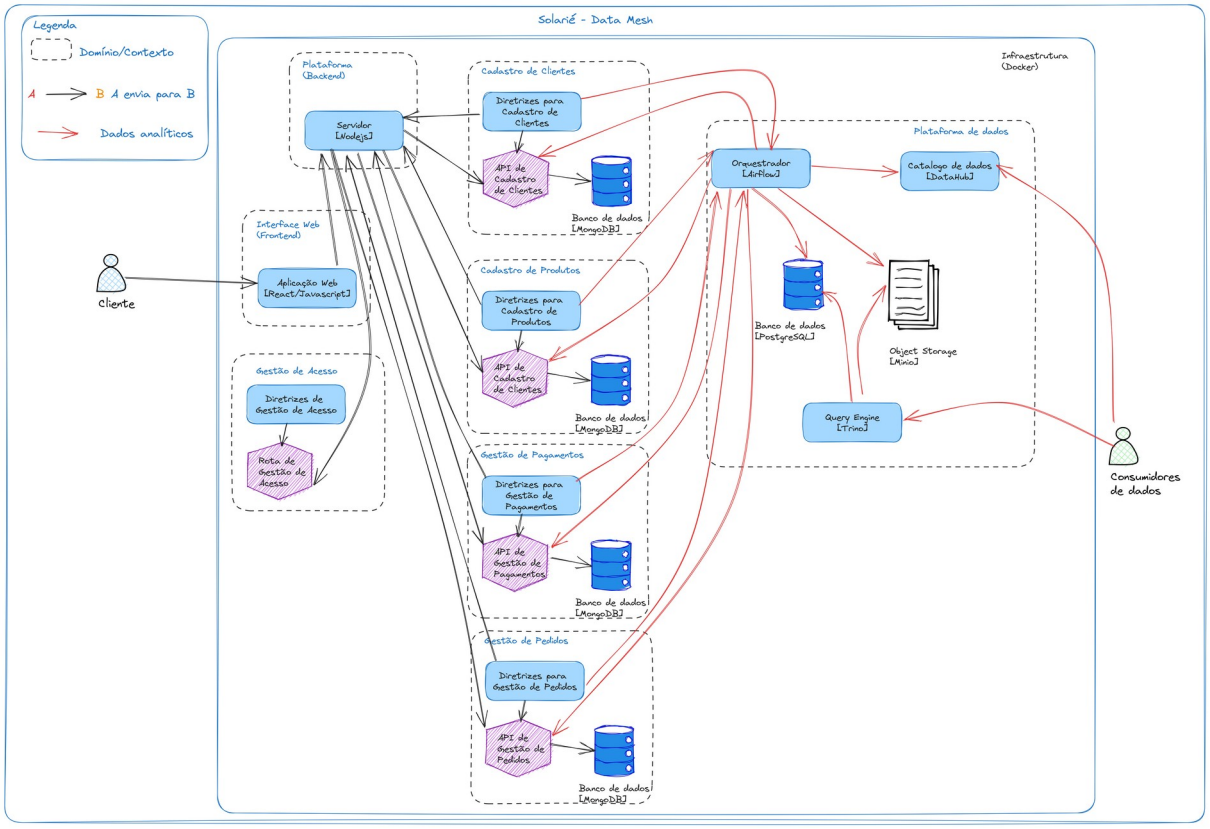
Quanto a rede, criou-se no Docker Compose a rede denominada *datanet* e atribui-se endereços de IP aos contêineres. Isto foi feito, para evitar erros de *deploy* devido a grande quantidade de contêineres.

Cabe ressaltar, que outras tecnologias também poderiam ser utilizadas, podendo ser *open source* ou não. Logo, a proposta não é oferecer um modelo de implementação de *Data Mesh*, mas demonstrar através da tecnologia alguns conceitos abordados. A seguir, será apresentada a Arquitetura utilizada no Projeto.

3.2 ***Arquitetura***

Em relação a arquitetura, a seguir há uma representação de como os Domínios foram separados e como estão interligados através do *Data Mesh* (Fig. 6):

Figura 6 - Arquitetura Data Mesh do Solarié



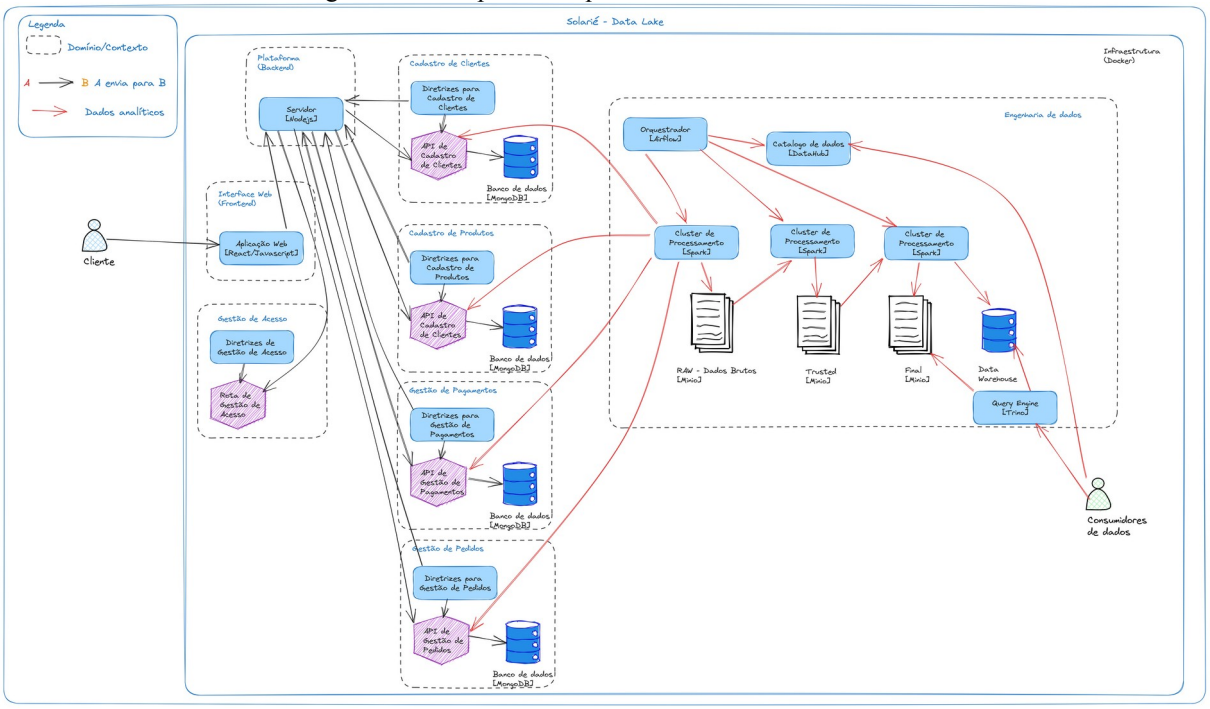
Fonte: Renan Ferreira Lima, 2023.

A exemplificação da arquitetura da figura pode ser feita utilizando o Domínio de Cadastro de Clientes como exemplo. Desta forma, este Domínio passa as diretrizes dos dados operacionais de Cadastro de Clientes para o Backend, este por sua vez segue as diretrizes e faz a integração de dados via API do banco de dados, no caso o MongoDB. Em paralelo, o Domínio de Cadastro de Clientes passa as diretrizes do Produto de dados para o Domínio de Plataforma, o qual acessa os dados operacionais de Cadastro de Clientes, também via API do MongoDB. Em seguida, aplica as diretrizes estabelecidas (no cenário ideal, através de automação) e disponibiliza o Produto de Dados no *Object Storage*. Sem antes, contudo, fazer a catalogação destes dados. Além do Produto de Dados, o Domínio de Plataforma de dados também cataloga os dados operacionais.

Na outra ponta, os Consumidores de dados, que são outros Domínios ou Contexto, podem acessar o Catálogo de Dados e os Produtos de Dados que tiverem acesso.

Já a Fig. 7, apresenta como esta arquitetura seria se fosse utilizado uma Arquitetura de dados com *Data Lake* e *Data Warehouse*:

Figura 7 - Exemplo de Arquitetura Data Lake do Solarié



Fonte: Renan Ferreira Lima, 2023.

Neste outro exemplo de arquitetura, manteve-se a divisão em Domínios. Entretanto, nesta abordagem há uma segregação forte entre dados operacionais e dados analíticos.

Deste modo, os Domínios mantém a propriedade sobre os dados operacionais, enquanto o time de Engenharia de Dados fica com a propriedade dos dados analíticos. Assim, o time de Engenharia de Dados ingere todos esses dados de forma bruta, para isso acessa os dados operacionais via API dos Domínios. Em seguida, cria esteiras para gerar os dados analíticos e disponibilizá-los na camada final do *Data Lake* e no *Data Warehouse*. Além disso, cataloga este dados e disponibiliza acesso ao catálogo para os consumidores.

3.3 *Executando o Projeto*

A etapa inicial para executar o Projeto é clonar o repositório do Github. O que pode ser feito utilizando o comando ***git clone https://github.com/lima-renan/solarie.git***.

Em seguida, acessando a pasta do diretório clonado, deve-se configurar o ambiente conforme o arquivo ***requirements.txt*** e, sem seguida, executar o comando: ***docker-compose -f docker-compose.DataMesh.yml up***. Cabe ressaltar, que é necessário ter o Docker ("Docker", 2022) e o Docker Compose ("Docker Compose overview", 2023) instalados no ambiente.

Em seguida, deve-se executar o Docker Compose do Catálogo de dados, o processo é feito através do comando: ***docker-compose -f catalog/docker-compose.DataHub.yml up***.

Caso não haja erros durante o processo, o Projeto estará em execução e as Funcionalidades poderão ser exploradas, conforme será apresentado na seção seguinte.

3.4 *Funcionalidades*

Quanto as funcionalidades, é possível acessar o Solarié através do endereço ***http://localhost:3000*** ou ***http://172.21.0.6:3000***. Onde é possível criar um novo cadastro, realizar o *login*, adicionar Produtos na Sacola e na Lista de Favoritos.

Também é possível simular uma compra, já que o *Frontend* está conectado com uma API de Teste de Pagamentos. Contudo, isto já fazia parte do *template* e não foi alterado para o Projeto.

Já o *Backend* está no endereço: ***http://localhost:8081 ou http://172.21.0.4:8081***. Onde foram utilizadas as seguintes rotas:

a) ***http://172.21.0.4:8081/user***: Rota principal para chamadas relativas ao Cadastro e *Login/Logout* de Usuários;
b) ***http://172.21.0.4:8081/user/signup***: subrota para Cadastro de Usuário;
c) ***http://172.21.0.4:8081/user/login***: subrota para *Login* de Usuário;

d) ***http://172.21.0.4:8081/user/cart***: subrota para armazenar produtos da Sacola do Usuário;
e) ***http://172.21.0.4:8081/user/wishlist***: subrota para armazenar produtos da Lista de Favoritos do Usuário;
f) ***http://172.21.0.4:8081/product e http://172.21.0.4:8081/products***: subrotas que permitem obter dados de um ou mais Produtos.;
g) ***http://172.21.0.4:8081/categories***: subrota que permite obter dados das Categorias de Produtos.

Quanto as demais configurações, como IP dos bancos, usuários, senhas e variáveis de ambiente. Estas, foram disponibilizadas através do arquivo ***requirements.txt*** do repositório do Github. A seguir, serão analisados os pontos de melhorias e possíveis evoluções.

3.5 *Melhorias e Evoluções*

O Solarié é uma demonstração de conceitos abordados no estudo do *Data Mesh*. No entanto, como esta é uma abordagem sociotécnica, esta demonstração não é capaz de apresentar de forma aplicada como seria a interação entre os Domínios em um cenário real.

Uma vez que, em um cenário real seria preciso entender o Modelo Operacional vigente da Organização e, como consequência, se a Organização está divida em Domínios e como eles interagem entre si. Além disso, seria necessário entender as questões técnicas, regulatórias, riscos, investimentos, entre outras.

Tendo isso em vista, esta análise terá o enfoque apenas na questão técnica. Sendo que, questões de Modelos Operacionais serão colocadas de maneira elocubrada. Assim, em termos técnicos, há grande oportunidade de mudança principalmente na camada de automação.

No *Data Mesh* isto é englobado pelo princípio de Plataforma de Dados com Autosserviço. No Solarié, utilizou-se o Airflow como Plataforma de Dados, contudo ele

poderia passar por uma automação mais profunda, para assim abstrair qualquer questão técnica e operacional que ocorre por trás.

Exemplificando, poderia ter alguma Interface, preferivelmente, amigável ao usuário final. Onde este indicasse qual Produto de Dados deseja consumir e em qual formato, em seguida a Plataforma provisionaria o acesso de acordo com o princípio de Governança Computacional Federada. Por outro lado, os Produtores de dados, poderiam ter uma experiência parecida para disponibilizar seus Produtos de dados ou solicitar uma esteira de processamento.

Outro ponto, seria o de evoluir mais o *Frontend* e o *Backend* para permitirem uma experiência de um *site* de compras real. Assim, seria possível trabalhar com mais Produtos de Dados e exercitar mais profundamente o conceito de *Data Mesh*, avaliando como políticas de Segurança, Governança de dados*, Compliance* e de relacionamento entre Domínios funcionaria. Isto seria um exercício para avaliar o escopo mais voltado ao Modelo Operacional do *Data Mesh.* Além disso, permitiria a criação de novos Domínios como os de Gestão de Pagamentos e Gestão de Pedidos, os quais até foram previstos, mas não foram implementados.

Por fim, a catalogação e qualidade dos metadados poderia ser aprimorada, permitindo, por exemplo, automação para marcação de dados sensíveis, pessoais ou confidenciais. Assim como, um nível de detalhamento maior de indicadores de qualidade e linhagem de dados, o que possibilitaria melhorar os Produtos de dados criados e estaria mais alinhado com o fundamentado por Dehghani.

4 CONCLUSÃO

As Arquiteturas de dados evoluíram rapidamente nas últimas décadas, assim como, a importância dos dados em decisões e posicionamento estratégico das Organizações. Saber utilizar os dados de maneira adequada e eficiente pode representar ter uma vantagem competitiva, ou um produto de melhor qualidade, além de outros benefícios.

Ao mesmo tempo, utilizar uma estratégia de dados equivocada pode implicar em grande ineficiência, perdas financeiras e riscos para o negócio. Em meio a isso, há diversas abordagens como o *Data Warehouse*, o *Data Lake* e, mais recentemente, o *Lakehouse*, o *Data Fabric* e o *Data Mesh*.

O *Data Mesh* se diferencia dos demais, principalmente, por sua abordagem sociotécnica. Traz com isso, uma visão descentralizada e trata os dados como produtos, quebrando a barreira entre dados operacionais e dados analíticos.

Para isso, propõe um novo Modelo Operacional, onde o conhecimento dos Domínio sobre os seus dados passa a ser a chave para se obter melhores análises. Além disso, defende a automação e autonomia, trazendo abordagens já utilizadas na Arquitetura de software, como o *Domain-Driven Design* e a Arquitetura de Microsserviços, como fundo para expressar suas ideias.

No entanto, por ser uma abordagem nova, ainda há poucas referências de implementação. Seja por desafios técnicos ou estratégicos, ou ainda por dificuldades de aplicar o modelo teórico apresentado por Dehghani.

Um ponto em comum, é que os motivadores para a adoção do *Data Mesh* são convidativos. Os Modelos Operacionais que centralizam os dados analíticos já não atendem a demanda de muitas Organizações, fazendo que não se consiga utilizar corretamente ou em tempo hábil os dados coletados.

Contudo, durante a elaboração da demonstração do *Data Mesh*, o que ficou mais evidente é que, talvez, o maior desafio dessa abordagem seja justamente restruturar o Modelo Operacional vigente na Organização que o for implementar. Uma vez que, isso envolve ter

um bom domínio de Arquitetura Corporativa para modelar os Domínios de maneira correta e precisa. De modo que, eles possam ser interconectados de forma clara e possibilite o Data Mesh.

Outro fator, é que nas Organizações onde há essa modelagem seguindo as premissas do *DDD*, ocorre dos Domínios não suportarem, pelo menos em um primeiro momento (BODE et al., 2023), a demanda de ter que construir Produtos de Dados.

Também é um desafio, construir uma Governança Computacional Federada que atenda de forma autônoma e eficiente o Data Mesh. Pois, este processo envolve conciliar políticas de diversas equipes, assim como, questões regulatórias, além de garantir a observabilidade para não incorrer em riscos.

Outro ponto, é que ter esteiras de automação, juntamente com metadados de qualidade e bem catalogados, pode ser um desafio para muitas Organizações. E isso, pode inviabilizar a adoção do *Data Mesh*.

Por fim, é esperado que o *Data Mesh*, assim como *Data Fabric*, ganhem cada vez mais espaço nos próximos anos (“Data Mesh vs Data Fabric”, 2023). Isto vai contribuir para novas abordagens empíricas, o que tende a enriquecer estas abordagens, além de servir como referência para sua implementação.

5 REFERÊNCIAS BIBLIOGRÁFICAS

6 Indexes. Disponível em: <https://docs.oracle.com/cd/A84870_01/doc/server.816/a76994/indexes.htm>. Acesso em: 8 ago. 2023.

A Metadata Platform for the Modern Data Stack | DataHub. Disponível em: <https://datahubproject.io/>. Acesso em: 8 ago. 2023.

About Databricks, founded by the original creators of Apache Spark™. Disponível em: <https://www.databricks.com/company/about-us>. Acesso em: 8 ago. 2023.

ADADI, A. A survey on data-efficient algorithms in big data era. **Journal of Big Data**, v. 8, n. 1, p. 24, 2021.

Amazon Web Services (AWS). Disponível em: <https://aws.amazon.com/pt/>. Acesso em: 8 ago. 2023.

Apache Airflow. Disponível em: <https://airflow.apache.org/>. Acesso em: 8 ago. 2023.

Apache Beam. Disponível em: <https://beam.apache.org/about/>. Acesso em: 8 ago. 2023.

Apache Beam | Google Open Source Projects. Disponível em: <https://opensource.google/projects/apachebeam>. Acesso em: 8 ago. 2023.

Apache Spark™ - Unified Engine for large-scale data analytics. Disponível em: <https://spark.apache.org/>. Acesso em: 9 ago. 2023.

Armazenamento S3 - Simple Storage Service - Amazon Web Services. Disponível em: <https://aws.amazon.com/pt/s3/>. Acesso em: 9 ago. 2023.

ARMBRUST, M. et al. Delta lake: high-performance ACID table storage over cloud object stores. **Proceedings of the VLDB Endowment**, v. 13, n. 12, p. 3411–3424, 2020.

ARMBRUST, M. et al. **Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics**. Proceedings of CIDR. **Anais**...2021.

Authentication in React using Express, Node, Passport and MongoDB | CodingDeft.com. Disponível em: <https://www.codingdeft.com/posts/react-authentication-mern-node-passport-express-mongo/>. Acesso em: 8 ago. 2023.

Azure Blob Storage | Microsoft Azure. Disponível em: <https://azure.microsoft.com/en-us/products/storage/blobs>. Acesso em: 9 ago. 2023.

BODE, J. et al. Data Mesh: Motivational Factors, Challenges, and Best Practices. **arXiv preprint arXiv:2302.01713**, 2023.

BOSWELL, M. et al. **Contêineres e zonas do data lake - Cloud Adoption Framework**. Disponível em: <https://learn.microsoft.com/pt-br/azure/cloud-adoption-framework/scenarios/cloud-scale-analytics/best-practices/data-lake-zones>. Acesso em: 8 ago. 2023.

CHANG, W. L.; GRADY, N. Nist big data interoperability framework: Volume 1, definitions. 2019.

Cloud Storage. Disponível em: <https://cloud.google.com/storage?hl=pt-br>. Acesso em: 9 ago. 2023.

Cloud Storage como um data lake | Centro de arquitetura do Cloud. Disponível em: <https://cloud.google.com/architecture/build-a-data-lake-on-gcp?hl=pt-br>. Acesso em: 8 ago. 2023.

COLLEGEWAP. **collegewap/mern-auth-server**. , 23 maio 2023. Disponível em: <https://github.com/collegewap/mern-auth-server>. Acesso em: 8 ago. 2023

Data Lake | Implementations | AWS Solutions. Disponível em: <https://aws.amazon.com/solutions/implementations/data-lake-solution/>. Acesso em: 8 ago. 2023.

Data Mesh vs Data Fabric: Key Differences & Proven Benefits. Disponível em: <https://www.informatica.com/blogs/data-fabric-vs-data-mesh-3-key-differences-how-they-help-and-proven-benefits.html>. Acesso em: 8 ago. 2023.

DEAN, J.; GHEMAWAT, S. MapReduce: simplified data processing on large clusters. **Communications of the ACM**, v. 51, n. 1, p. 107–113, 2008.

DEHGHANI, Z. **How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh**. Disponível em: <https://martinfowler.com/articles/data-monolith-to-mesh.html>. Acesso em: 8 ago. 2023.

DEHGHANI, Z. **Data Mesh Principles and Logical Architecture**. Disponível em: <https://martinfowler.com/articles/data-mesh-principles.html>. Acesso em: 8 ago. 2023.

DEHGHANI, Z. **Data Mesh: Delivering Data-Driven Value at Scale**. 1ª edição ed. Sebastopol, CA: O'Reilly Media, 2022.

Deploying with Docker | DataHub. Disponível em: <https://datahubproject.io/docs/docker/>. Acesso em: 8 ago. 2023.

Distributed SQL query engine for big data. Disponível em: <https://trino.io/>. Acesso em: 8 ago. 2023.

Docker: Accelerated Container Application Development. Disponível em: <https://www.docker.com/>. Acesso em: 8 ago. 2023.

Docker Compose overview. Disponível em: <https://docs.docker.com/compose/>. Acesso em: 8 ago. 2023.

DOLHOPOLOV, A.; CASTELLTORT, A.; LAURENT, A. Trick or Treat: Centralized Data Lake vs Decentralized Data Mesh. 2023.

Elastic Observability — uma solução aberta e extensível para equipes de DevOps | Elastic. Disponível em: <https://www.elastic.co/pt/observability>. Acesso em: 8 ago. 2023.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 6ª edição ed. [s.l.] Pearson Universidades, 2010.

ESTABROOK, N. et al. **Introdução ao Azure Data Lake Storage Gen2 - Azure Storage**. Disponível em: <https://learn.microsoft.com/pt-br/azure/storage/blobs/data-lake-storage-introduction>. Acesso em: 9 ago. 2023.

FANG, H. **Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem**. 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). **Anais**...IEEE, 2015.

FOWLER, M.; SADALAGE, P. J. **NoSQL Essencial: um Guia Conciso Para o Mundo Emergente da Persistência Poliglota**. 1ª edição ed. [s.l.] Novatec Editora, 2013.

Google Cloud Platform. Disponível em: <https://cloud.google.com/>. Acesso em: 8 ago. 2023.

Hadoop MapReduce: Introdução a Big Data. Disponível em: <https://www.devmedia.com.br/hadoop-mapreduce-introducao-a-big-data/30034>. Acesso em: 8 ago. 2023.

Hadoop vs. Spark | Diferença entre frameworks do Apache | AWS. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-hadoop-vs-spark/>. Acesso em: 8 ago. 2023.

INMON, B. **Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump**. First Edition ed. Basking Ridge, NJ: Technics Publications, 2016.

INMON, W. H. **Building the Data Warehouse**. 4th edition ed. Indianapolis, Ind: Wiley, 2005.

ISEMINGER, D. et al. **Entenda o esquema em estrela e a importância para o Power BI - Power BI**. Disponível em: <https://learn.microsoft.com/pt-br/power-bi/guidance/star-schema>. Acesso em: 8 ago. 2023.

KHATRI, V.; BROWN, C. V. Designing data governance. **Communications of the ACM**, v. 53, n. 1, p. 148–152, 2010.

KHINE, P. P.; WANG, Z. S. **Data lake: a new ideology in big data era**. ITM web of conferences. **Anais**...EDP Sciences, 2018.

LIN, J. The Lambda and the Kappa. **The Lambda and the Kappa**, p. 60–66, set. 2017.

Microsoft Azure. Disponível em: <https://azure.microsoft.com/pt-br/free/search>. Acesso em: 8 ago. 2023.

MILOSLAVSKAYA, N.; TOLSTOY, A. Big data, fast data and data lake concepts. **Procedia Computer Science**, v. 88, p. 300–305, 2016.

MinIO | High Performance, Kubernetes Native Object Storage. Disponível em: <https://min.io>. Acesso em: 8 ago. 2023.

Modelo de programação para o Apache Beam | Cloud Dataflow. Disponível em: <https://cloud.google.com/dataflow/docs/concepts/beam-programming-model?hl=pt-br>. Acesso em: 8 ago. 2023.

MongoDB: A Plataforma De Aplicação De Dados. Disponível em: <https://www.mongodb.com/pt-br>. Acesso em: 8 ago. 2023.

MOURA, C.; SOTTA, E.; RUBINOV, K. **Pros and Cons of Data Mesh**. Disponível em: <https://www.pwc.ch/en/insights/data-analytics/data-mesh-challenges.html>. Acesso em: 8 ago. 2023.

Node.js. Disponível em: <https://nodejs.org/en>. Acesso em: 8 ago. 2023.

O que é armazenamento de objetos? – Explicação sobre armazenamento de objetos – AWS. Disponível em: <https://aws.amazon.com/pt/what-is/object-storage/>. Acesso em: 8 ago. 2023.

O que é um data lake? Disponível em: <https://aws.amazon.com/pt/big-data/datalakes-and-analytics/what-is-a-data-lake/>. Acesso em: 8 ago. 2023.

O que é um Data Lake? Data Lake vs. Warehouse | Microsoft Azure. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-a-data-lake/>. Acesso em: 8 ago. 2023.

PostgreSQL. Disponível em: <https://www.postgresql.org/>. Acesso em: 8 ago. 2023.

PRASAD, A. et al. **Data lakes - Azure Architecture Center**. Disponível em: <https://learn.microsoft.com/en-us/azure/architecture/data-guide/scenarios/data-lake>. Acesso em: 8 ago. 2023.

React. Disponível em: <https://react.dev/>. Acesso em: 8 ago. 2023.

Recommended data layers - AWS Prescriptive Guidance. Disponível em: <https://docs.aws.amazon.com/prescriptive-guidance/latest/defining-bucket-names-data-lakes/data-layer-definitions.html>. Acesso em: 8 ago. 2023.

SALLOUM, S. et al. Big data analytics on Apache Spark. **International Journal of Data Science and Analytics**, v. 1, p. 145–164, 2016.

SANDHYAR1007. **SandhyaR1007/eyesome-react**. , 8 ago. 2023. Disponível em: <https://github.com/SandhyaR1007/eyesome-react>. Acesso em: 8 ago. 2023

Silos de dados: o que são e quais os prejuízos? Disponível em: <https://www.salesforce.com/br/blog/2020/04/silos-de-dados-o-que-sao.html>. Acesso em: 10 ago. 2023.

TEKINER, F. et al. Build a modern, distributed Data Mesh with Google Cloud. 2023.

WATSON, H. J.; WIXOM, B. H. The current state of business intelligence. **Computer**, v. 40, n. 9, p. 96–99, 2007.

What is Apache Spark? | Introduction to Apache Spark and Analytics | AWS. Disponível em: <https://aws.amazon.com/big-data/what-is-spark/>. Acesso em: 8 ago. 2023.