Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Arquitetura de Data Fabric para Armazenamento e Processamento de Imagens de Câncer de Pele

Autor: Abdul Hannan, Heitor Marques Simões Barbosa
Orientador: Dra. Carla Silva Rocha Aguiar

Brasília, DF
2025

Abdul Hannan, Heitor Marques Simões Barbosa

Arquitetura de Data Fabric para Armazenamento e Processamento de Imagens de Câncer de Pele

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Dra. Carla Silva Rocha Aguiar

Brasília, DF

2025

Abdul Hannan, Heitor Marques Simões Barbosa

Arquitetura de Data Fabric para Armazenamento e Processamento de Imagens de Câncer de Pele/ Abdul Hannan, Heitor Marques Simões Barbosa. – Brasília, DF, 2025-

67 p. : il. (algumas color.) ; 30 cm.

Orientador: Dra. Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2025.

1. Data Fabric. 2. Arquitetura de Dados. I. Dra. Carla Silva Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Arquitetura de Data Fabric para Armazenamento e Processamento de Imagens de Câncer de Pele

CDU 02:141:005.6

Abdul Hannan, Heitor Marques Simões Barbosa

Arquitetura de Data Fabric para Armazenamento e Processamento de Imagens de Câncer de Pele

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 01 de junho de 2025 – Data da aprovação do trabalho:

Dra. Carla Silva Rocha Aguiar
Orientador

Titulação e Nome do Professor
Convidado 01
Convidado 1

Titulação e Nome do Professor
Convidado 02
Convidado 2

Brasília, DF
2025

Resumo

Este trabalho propõe uma arquitetura de Data Fabric para armazenamento e processamento de imagens de câncer de pele, visando facilitar a gestão eficiente, segura e escalável desses dados. O estudo se baseia em uma abordagem teórica e conceitual, e utiliza revisão bibliográfica para embasar a viabilidade da proposta. A metodologia inclui um estudo de caso que analisa os desafios, requisitos e possíveis soluções para a adoção de uma infraestrutura de Data Fabric nesse contexto. A arquitetura proposta é estruturada para garantir integração de dados de diversas fontes, segurança e privacidade no armazenamento e compartilhamento das imagens, além de possibilitar a governança centralizada e a escalabilidade do sistema. Com base na revisão bibliográfica e no estudo teórico, observa-se que o Data Fabric apresenta potencial para otimizar o acesso e processamento de imagens médicas, promovendo interoperabilidade e eficiência no tratamento de dados. Como conclusão, o estudo propõe um roadmap para implementação futura da arquitetura, estruturado em etapas sequenciais: configuração do ambiente de desenvolvimento e implantação do cluster Kubernetes, integração do MinIO para armazenamento distribuído, instalação do Delta Lake para gestão transacional de dados, configuração do Apache Airflow para orquestração de pipelines e, por fim, implementação do Delta Sharing para compartilhamento seguro de dados. A conclusão reforça a importância da continuidade da pesquisa, destacando a necessidade de validação prática da arquitetura e a investigação de desafios técnicos e regulatórios para sua implementação em ambientes clínicos reais.

Palavras-chave: Data Fabric. Arquitetura de Dados. Armazenamento de Imagens Médicas. Segurança da Informação. Interoperabilidade.

Abstract

This work proposes a Data Fabric architecture for storing and processing skin cancer images, aiming to facilitate the efficient, secure, and scalable management of such data. The study is based on a theoretical and conceptual approach, using a literature review to support the feasibility of the proposed solution. The methodology includes a case study that analyzes the challenges, requirements, and possible solutions for adopting a Data Fabric infrastructure in this context. The proposed architecture is structured to ensure data integration from multiple sources, security and privacy in image storage and sharing, as well as enabling centralized governance and system scalability. Based on the literature review and theoretical study, it is observed that Data Fabric has the potential to optimize access and processing of medical images, promoting interoperability and efficiency in data management. As a conclusion, the study proposes a roadmap for future implementation of the architecture, structured in sequential stages: setting up the development environment and deploying the Kubernetes cluster, integrating MinIO for distributed data storage, installing Delta Lake for transactional data management, configuring Apache Airflow for pipeline orchestration, and finally, implementing Delta Sharing for secure data sharing. The conclusion reinforces the importance of continuing research, highlighting the need for practical validation of the architecture and the investigation of technical and regulatory challenges for its implementation in real clinical environments.

Key-words: Data Fabric. Data Architecture. Medical Image Storage. Information Security. Interoperability.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

LGPD	Lei Geral de Proteção de Dados
GDPR	General Data Protection Regulation
API	Application Programming Interface
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
CNCF	Cloud Native Computing Foundation
CORS	Cross-Origin Resource Sharing
DAG	Directed Acyclic Graph (Grafo Acíclico Direcionado)
ETL	Extract, Transform, Load (Extração, Transformação e Carga)
HDFS	Hadoop Distributed File System
RDD	Resilient Distributed Datasets
HPA	Horizontal Pod Autoscaler
ELK	Elasticsearch, Logstash e Kibana
IA	Inteligência Artificial
JSON	JavaScript Object Notation
JWT	JSON Web Token
LOD	Linked Open Data
ML	Machine Learning (Aprendizado de Máquina)
CRUD	Create, Read, Update, Delete
HTTP	Hypertext Transfer Protocol
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
RBAC	Role-Based Access Control

Sumário

1 Introdução

1.1 INTRODUÇÃO

Com o avanço da tecnologia, armazenar e processar grandes volumes de dados ficou mais ágil e inteligente. Nesse cenário, o Data Fabric se destaca como uma solução poderosa, especialmente para lidar com imagens dos diferentes fontes. Ele cria uma conexão fluida e integrada entre os dados, permitindo acesso rápido, seguro e escalável.

Este trabalho tem como objetivo propor uma arquitetura de Data Fabric para armazenar e processar imagens de câncer de pele. A metodologia usada inclui uma revisão detalhada da literatura já existente e um estudo de caso que analisa os desafios, requisitos e possíveis soluções para implementar essa infraestrutura. A revisão da literatura fornece a base teórica necessária para a viabilidade da proposta, enquanto o estudo de caso oferece uma análise prática das questões envolvidas.

A arquitetura proposta neste estudo é projetada para integrar dados de várias fontes, garantindo a segurança e privacidade no armazenamento e compartilhamento das imagens. Além disso, ela permite uma governança centralizada e a escalabilidade do sistema, aspectos fundamentais para a gestão eficiente de dados médicos. A integração de ferramentas como Kubernetes, MinIO, Delta Lake, Apache Airflow e Delta Sharing é detalhada, destacando suas funções e benefícios no contexto da arquitetura de Data Fabric.

1.1.1 Justificativa

A demanda por soluções mais inteligentes e seguras para armazenar e processar imagens cresce a cada dia, impulsionando a criação de novas arquiteturas de dados. O Data Fabric surge como uma resposta moderna a esse desafio, oferecendo uma gestão integrada, escalável e eficiente. Com foco na privacidade e na segurança, essa abordagem garante que as informações sejam acessadas e utilizadas de forma confiável, simplificando processos.

1.1.2 Objetivos

Os objetivos deste trabalho são:

- Propor uma arquitetura de Data Fabric para armazenar e processar imagens de câncer de pele.
- Analisar os desafios e requisitos para a implementação desta arquitetura.

- Apresentar um roadmap detalhado para a implementação da arquitetura proposta.
- Unifica dados de várias fontes em um armazenamento MinIO escalável, permitindo análises avançadas com segurança e eficiência.

1.1.3 Estrutura do Documento

Este documento está estruturado da seguinte forma:

- **Capítulo 1** - Introdução: Apresenta o contexto, a justificativa, os objetivos e a estrutura do documento.
- **Capítulo 2** - Referencial Teórico: Revisão da literatura sobre Data Fabric, Engenharia de Dados, Data Warehouse, Data Lake, Data Mesh, Apache Airflow, Apache Spark, Kubernetes, FastAPI, e Pipelines de dados.
- **Capítulo 3** - Metodologia: Detalhamento da metodologia adotada, incluindo o estudo de caso e as perguntas de pesquisa.
- **Capítulo 4** - Projeto: Descrição dos requisitos, arquitetura proposta, tecnologias utilizadas e roadmap de implementação.

2 Referencial Teórico

2.1 Engenharia de Dados

A era da informação, impulsionada pela digitalização crescente, nos coloca diante de uma explosão de dados sem precedentes. Nesse cenário, a Engenharia de Dados emerge como uma área essencial para transformar essa massa de dados brutos em informação valiosa e conhecimento estratégico. A Engenharia de Dados, utilizando princípios de engenharia e ciência da computação, projeta, constrói e gerencia sistemas robustos de dados, abrangendo desde a coleta e armazenamento até o processamento, a análise e a visualização.

A importância da Engenharia de Dados reside na sua capacidade de extrair valor dos dados. Através da conversão de dados brutos em insights acionáveis, as organizações podem otimizar suas operações, identificar tendências de mercado, personalizar a experiência do cliente e tomar decisões estratégicas mais eficazes. Essa capacidade é crucial no contexto do Big Data, com seus desafios de volume, velocidade, variedade e veracidade.

Garantir a qualidade dos dados é um fator crítico. Decisões baseadas em dados imprecisos ou inconsistentes podem acarretar resultados indesejáveis, ineficiência operacional e perda de oportunidades (REDMAN, 1998). A Engenharia de Dados busca assegurar a qualidade dos dados por meio de processos de limpeza, transformação e validação, garantindo que os dados sejam confiáveis e adequados para análise (HAUG; ZACHARIASSEN; LIEMPD, 2011).

2.1.1 Histórico e Evolução

Engenharia de Dados tem suas raízes em áreas como gerenciamento de banco de dados e processamento de dados. O conceito de dados como ativo surgiu em 1994 com o Hawley Committee, que definiu ativos de dados como "Dados que são ou devem ser documentados e que têm valor ou valor potencial". Essa visão impulsionou a importância da governança de dados dentro das organizações (ALHASSAN; SAMMON; DALY, 2016). Essa necessidade já apontava para a importância de tratar os dados como um ativo estratégico e gerenciá-los adequadamente.

A necessidade de lidar com o Big Data impulsionou o desenvolvimento de arquiteturas voltadas para processamento distribuído, como Hadoop e Spark, além de bancos de dados NoSQL, capazes de lidar com grandes volumes e diferentes formatos de dados (VOLK et al., 2019). A ascensão da computação em nuvem teve um impacto significativo na evolução da Engenharia de Dados, fornecendo infraestrutura escalável e flexível para

armazenamento e processamento de dados (KATAL; WAZID; GOUDAR, 2013). Além disso, a integração com o aprendizado de máquina vem ganhando relevância, permitindo a aplicação de algoritmos de aprendizado para análise preditiva, detecção de anomalias e outras tarefas complexas (L'HEUREUX et al., 2017).

2.1.2 Desafios Atuais

A Engenharia de Dados enfrenta desafios complexos e multifacetados, impulsionados pela natureza dinâmica dos dados modernos. Entre os principais desafios, destaca-se a questão da escalabilidade. A necessidade de processar volumes crescentes de dados, provenientes de diversas fontes, exige sistemas e infraestruturas altamente escaláveis. As arquiteturas tradicionais de banco de dados frequentemente se mostram inadequadas para lidar com a escala do Big Data, demandando a adoção de novas tecnologias e abordagens (KATAL; WAZID; GOUDAR, 2013).

Outro desafio relevante é a variedade de dados. A heterogeneidade dos dados, que pode incluir informações estruturadas, semi-estruturadas e não estruturadas, representa um obstáculo significativo para a integração e o processamento. A Engenharia de Dados deve lidar com diferentes formatos, como texto, imagens, áudio e vídeo, além de dados de sensores, logs de eventos e mídias sociais (VOLK et al., 2019). Essa necessidade de integrar e analisar dados de origens e formatos diversos exige o uso de ferramentas e técnicas especializadas, como ETL e processamento de linguagem natural.

A velocidade dos dados é também um aspecto desafiador. O processamento em tempo real, como o streaming de dados, impõe a necessidade de arquiteturas e tecnologias adequadas para lidar com o fluxo constante de informações. A Engenharia de Dados desenvolve soluções capazes de capturar, processar e analisar dados em tempo real, garantindo a eficiência e a agilidade necessárias para responder às demandas modernas (L'HEUREUX et al., 2017).

Garantir a precisão, consistência e confiabilidade das informações é essencial, especialmente no ambiente de Big Data, onde a complexidade e a variedade de dados podem amplificar os problemas relacionados à qualidade. Conforme destacado por Redman (1998), a baixa qualidade dos dados pode resultar em insatisfação dos clientes, aumento de custos, tomada de decisões ineficazes e redução da capacidade de desenvolver e implementar estratégias. Em casos extremos, os custos associados à baixa qualidade dos dados podem representar entre 8% e 12% da receita de uma empresa, além de atingir até 40% a 60% das despesas em organizações de serviços (REDMAN, 1998).

A proteção de informações sensíveis e a conformidade com regulamentações, como a GDPR e a LGPD, requerem medidas de segurança abrangentes. Isso inclui a adoção de práticas como criptografia, controle de acesso e anonimização, que são fundamentais

para garantir a proteção dos dados e a privacidade dos usuários, atendendo às exigências legais e éticas.

2.1.3 Impacto das Dificuldades

As dificuldades enfrentadas na Engenharia de Dados podem ter um impacto significativo nas organizações, afetando diversos aspectos críticos. Uma das consequências mais evidentes é a tomada de decisão ineficaz. Decisões baseadas em dados imprecisos, inconsistentes ou incompletos podem gerar resultados indesejáveis, como investimentos equivocados, perda de clientes e redução da eficiência operacional. A ausência de dados confiáveis compromete a capacidade de uma organização de responder adequadamente às mudanças do mercado e de tomar decisões estratégicas eficazes (STONE et al., 2019).

Outro impacto importante é a perda de competitividade. A incapacidade de utilizar os dados de maneira eficaz prejudica a inovação e limita as vantagens competitivas. Por outro lado, empresas que conseguem extrair insights valiosos de seus dados têm a oportunidade de otimizar preços, desenvolver novos produtos e serviços e personalizar a experiência do cliente, conquistando uma posição de destaque no mercado (REDMAN, 1998).

Além disso, a gestão inadequada de dados pode levar ao aumento de custos operacionais. Problemas como a proliferação de silos de dados, a duplicação de informações e a falta de padronização resultam em custos mais altos de armazenamento, processamento e manutenção. Essas ineficiências podem representar um fardo significativo para as organizações, limitando seu crescimento e eficiência (REDMAN, 1998).

Por fim, os riscos de segurança e conformidade também são um aspecto crítico. Falhas na proteção de dados podem causar violações de segurança, multas significativas e danos irreparáveis à reputação da organização. Empresas que lidam com dados sensíveis, como informações pessoais de clientes, precisam assegurar a conformidade com regulamentações de privacidade, como a GDPR e a LGPD, além de adotar medidas robustas de segurança para prevenir vazamentos de informações e proteger a privacidade dos usuários (BENFELDT; PERSSON; MADSEN, 2020).

2.1.4 Práticas de Engenharia de Dados

A Engenharia de Dados utiliza diversas práticas essenciais para enfrentar os desafios e extrair o máximo valor dos dados. Uma dessas práticas é a modelagem de dados, que define a estrutura, as relações e os tipos de dados que serão armazenados, garantindo garantindo que os dados sejam organizados de forma lógica e eficiente, facilitando a análise e o processamento (VOLK et al., 2019).

Outra prática crucial é a arquitetura de dados, que define a estrutura geral dos

sistemas de dados, incluindo os componentes de hardware e software, os fluxos de dados e os mecanismos de segurança. Uma arquitetura bem projetada deve ser escalável, flexível, segura e atender aos requisitos específicos da organização. Nesse contexto, arquiteturas modernas, como data lakes e data warehouses, são projetadas para lidar com a variedade, o volume e a velocidade dos dados gerados atualmente (VOLK et al., 2019).

A integração de dados desempenha um papel fundamental na combinação de informações provenientes de diversas fontes, com o objetivo de criar uma visão unificada e coesa. O sistema deve ser capaz de integrar os dados extraídos de fontes textuais brutas com outras bases de dados, permitindo uma análise mais abrangente e eficiente. Contudo, um dos grandes desafios para alcançar essa integração é a existência de silos de dados, que atuam como barreiras, fragmentando as informações e dificultando a interoperabilidade entre sistemas. Superar esses silos e interconectar os dados de maneira eficiente é essencial para otimizar processos e garantir que as organizações possam explorar todo o potencial dos dados disponíveis (DOAN et al., 2008).

Além disso, a limpeza e transformação de dados é essencial para corrigir erros, remover inconsistências e tratar valores ausentes, nessa etapa há a necessidade de medidas para detectar e corrigir falhas na qualidade dos dados. A transformação de dados permite convertê-los para o formato desejado, seja por meio de agregação, normalização ou enriquecimento com dados externos. Essas etapas são indispensáveis para assegurar a qualidade dos dados e prepará-los adequadamente para análises (REDMAN, 1998).

Por fim, o gerenciamento de dados engloba a implementação de processos, políticas e ferramentas que garantem a qualidade, segurança, disponibilidade e integridade dos dados. Esse gerenciamento cobre todo o ciclo de vida dos dados, desde sua coleta até o descarte, e inclui atividades como o gerenciamento de metadados, controle de versões, backup e recuperação. Juntas, essas práticas formam a base para uma abordagem sólida e eficaz em Engenharia de Dados, atendendo às demandas crescentes e complexas das organizações modernas (JAHNKE; ASHER; KERALIS, 2012).

2.2 Data Warehouse

Um data warehouse é um repositório central de dados históricos que são usados para análise e tomada de decisão. Ele é projetado para ser **orientado por assunto**, **não volátil**, **integrado** e **variante no tempo**.

Orientado por assunto Um data warehouse é organizado em torno de assuntos específicos de negócios, como clientes, produtos ou vendas, em vez de processos de negócios ou aplicativos.

Não volátil Os dados em um data warehouse são somente leitura e não são atualizados

por transações online.

Integrado Um data warehouse integra dados de várias fontes operacionais, fornecendo uma visão única e consistente dos dados da organização.

Variante no tempo Os dados em um data warehouse incluem um componente de tempo, permitindo aos usuários analisar tendências históricas e fazer previsões.

A necessidade de data warehouses surgiu para suportar análise de dados em larga escala, em contraste com os sistemas OLTP, que são projetados para processar transações simples e frequentes. Enquanto os usuários de sistemas OLTP são principalmente funcionários operacionais que realizam operações diárias e acessam dados detalhados e atualizados, os usuários de data warehouses (OLAP) são geralmente gerentes ou analistas que realizam consultas complexas e análises históricas, acessando dados agregados e resumidos. As diferenças nos objetivos de uso resultam em modelos de dados distintos: OLTP usa esquemas altamente normalizados, enquanto OLAP adota modelos multidimensionais, frequentemente denormalizados, para otimizar a performance de consultas analíticas. Além disso, enquanto OLTP exige alta concorrência e processamento em tempo real, OLAP trabalha com dados lidos periodicamente e com menor volume de acessos simultâneos.

O objetivo de um data warehouse é fornecer aos usuários uma visão completa e precisa dos dados da organização, permitindo que tomem decisões mais informadas. As áreas típicas de aplicação de data warehouses incluem análise de vendas e marketing, gerenciamento de relacionamento com o cliente, planejamento de recursos empresariais e inteligência de negócios (NAMBIAR; MUNDRA, 2022; VAISMAN; ZIMáNYI, 2014).

2.2.1 Definição e Conceito

Um data warehouse pode ser definido como uma combinação de conceitos e tecnologias que facilitam a gestão e manutenção de dados históricos obtidos de aplicações operacionais e transacionais. Ele serve como um ambiente que auxilia os tomadores de decisão, como executivos, gerentes e analistas, a tomar decisões mais rápidas e informadas.

Em essência, um data warehouse não é um produto, mas sim um ambiente onde os usuários podem encontrar informações estratégicas. Ele funciona como um repositório centralizado e integrado de informações, extraídas de dados dispersos em diversos sistemas, com o propósito de auxiliar na tomada de decisões.

O data warehouse se diferencia de um banco de dados operacional por ser um resumo lógico dos dados, separado do banco de dados transacional. Ele permite a integração de diversos tipos de dados de diferentes aplicações ou sistemas, proporcionando um mecanismo de acesso unificado para que a gestão obtenha informações e as analise para a tomada de decisão (SANTOSO; YULIA, 2017).

2.2.2 Arquitetura

A arquitetura de um Data Warehouse é composta por várias camadas interconectadas, que trabalham em conjunto para garantir a eficiência na coleta, transformação e análise de grandes volumes de dados. A figura 1 ilustra as principais camadas de uma arquitetura geral de um Data Warehouse.

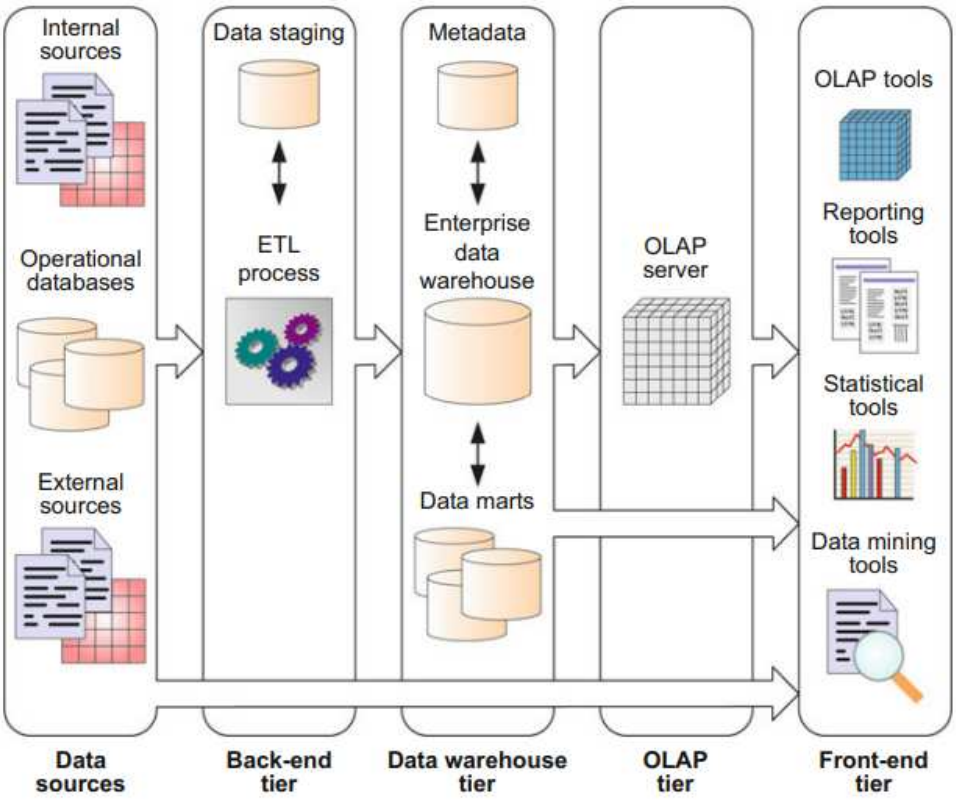

Figura 1 – Arquitetura geral de um Data Warehouse
(VAISMAN; ZIMáNYI, 2014)

A arquitetura apresentada é composta por quatro camadas principais:

Camada de Back-End Responsável pelos processos de extração, transformação e carga (ETL) dos dados. Aqui, os dados são extraídos de fontes heterogêneas, transformados para o formato adequado e carregados no Data Warehouse. Essa camada também pode envolver o uso de uma área de staging, onde os dados extraídos são temporariamente armazenados e processados antes de sua inserção no repositório central de dados.

Camada do Data Warehouse Composta pelo Data Warehouse (armazém de dados centralizado) e por Data Marts (armazéns de dados especializados em áreas específicas). Além disso, essa camada inclui o repositório de metadados, que armazena informações sobre a estrutura e os processos do Data Warehouse.

Camada OLAP (Online Analytical Processing) Esta camada envolve o uso de servidores OLAP, que permitem a exploração multidimensional dos dados armazenados no Data Warehouse. O OLAP facilita a análise de grandes volumes de dados de maneira interativa e flexível, utilizando cubos de dados e outras estruturas analíticas.

Camada Front-End Responsável pelas ferramentas utilizadas para análise e visualização dos dados. Inclui ferramentas OLAP, ferramentas de relatórios, ferramentas estatísticas e de mineração de dados, que permitem a exploração dos dados de forma interativa e a geração de insights valiosos para a tomada de decisões empresariais.

2.2.3 Quando Utilizar

A implementação de um data warehouse é amplamente recomendada em contextos onde as organizações precisam tomar decisões estratégicas baseadas em grandes volumes de dados históricos provenientes de diversas fontes. A necessidade de analisar tendências, identificar padrões e extrair insights estratégicos, que não são acessíveis por meio de sistemas transacionais tradicionais, justifica a adoção dessa tecnologia.

O data warehouse desempenha um papel essencial em cenários específicos. Por exemplo, quando os dados estão dispersos em diferentes sistemas, o que dificulta a obtenção de uma visão unificada da organização, o data warehouse se torna uma solução eficiente. Ele possibilita a integração de dados de múltiplas fontes, fornecendo uma visão única e consistente das informações organizacionais. Além disso, sua capacidade de armazenar dados históricos permite a análise aprofundada de tendências e padrões ao longo do tempo, fator indispensável para decisões baseadas em informações históricas (SANTOSO; YULIA, 2017).

Outro aspecto relevante é que organizações frequentemente necessitam de relatórios e análises complexas, que não podem ser gerados de forma eficaz por sistemas transacionais. Nesse contexto, o data warehouse, projetado especificamente para consultas e análises sofisticadas, oferece suporte à criação de relatórios detalhados e dashboards interativos. Ademais, ao fornecer um ambiente separado para análises, ele evita impactos no desempenho dos sistemas operacionais, cuja principal função é suportar as operações diárias da organização (SANTOSO; YULIA, 2017).

Embora os benefícios de um data warehouse sejam inegáveis, sua implementação exige uma avaliação cuidadosa dos custos e benefícios envolvidos. Trata-se de um processo que pode ser complexo, demorado e custoso, requerendo investimentos significativos em infraestrutura, software e equipes especializadas. Apesar disso, quando bem implementado, o data warehouse pode proporcionar vantagens significativas, como a melhoria na qualidade da tomada de decisões, por meio do fornecimento de informações precisas e oportunas, além de ganhos em eficiência operacional, resultantes da otimização de pro-

cessos baseada nos insights obtidos. Por fim, ele pode oferecer uma vantagem competitiva à organização, ao permitir análises preditivas que ajudam a antecipar tendências e oportunidades de mercado (VAISMAN; ZIMáNYI, 2014; MOUSA; SHIRATUDDIN, 2015).

2.2.4 Quando Não Utilizar

A implementação de um Data Warehouse, embora amplamente vantajosa em diversos contextos, apresenta limitações e desafios que podem torná-lo inadequado para determinados cenários. Um dos principais problemas está relacionado ao tempo e à complexidade do processo de ETL. Muitas vezes, os desenvolvedores não consideram adequadamente o tempo necessário para realizar essas etapas, resultando no esgotamento de grande parte do tempo destinado à construção do sistema. Essa situação pode gerar atrasos e custos inesperados, especialmente em projetos com cronogramas restritos.

Além disso, o modelo de design do Data Warehouse pode ser extremamente complexo e não reflexivo, dificultando sua adaptação às necessidades de negócios em constante mudança. Isso ocorre porque, ao adicionar novas fontes de dados ou realizar alterações nos processos, os designers do Data Warehouse frequentemente precisam redesenhar o ETL, resultando em modificações nas regras ou nas origens dos dados, o que aumenta a complexidade e o custo do sistema.

Outra limitação importante está na atualização do Data Warehouse, que geralmente é feita de forma offline. Isso significa que, enquanto o sistema é atualizado, as aplicações de BI não conseguem acessar os dados ou fornecer informações em tempo real. Esse atraso pode ser prejudicial para empresas que dependem de análises e tomadas de decisão imediatas, como em cenários de operações críticas ou de alta competitividade no mercado.

Ademais, o armazenamento de dados no Data Warehouse implica em custos adicionais. Como o Data Warehouse é uma cópia dos dados originais, ele requer um espaço físico maior para armazenamento, o que pode gerar custos elevados com licenciamento, manutenção e resposta a demandas de infraestrutura (MOUSA; SHIRATUDDIN, 2015).

2.3 Data Fabric

A crescente proliferação de dados em formatos e plataformas diversas apresenta um desafio significativo para as organizações. A necessidade de integrar, gerenciar e analisar esses dados de forma eficiente impulsionou a busca por soluções inovadoras. É nesse contexto que surge o conceito de Data Fabric, uma arquitetura emergente que visa unificar e simplificar o acesso e o uso de dados em todo o ecossistema de uma organização. Data Fabric pode ser definida como uma estrutura unificada e flexível, independente de plataformas e tecnologias, que conecta diferentes fontes de dados, permitindo o acesso e a

análise de forma ágil e eficiente (GADE, 2022). Essa arquitetura utiliza metadados para construir um conhecimento abrangente sobre os dados, facilitando sua descoberta, acesso e governança (BARIK, 2022).

Em sua essência, o Data Fabric visa:

- Superar os desafios dos silos de dados, integrando dados de diferentes fontes e formatos (GADE, 2022).
- Criar uma visão holística e unificada dos dados da organização (BARIK, 2022).
- Simplificar o acesso e a análise de dados, permitindo que usuários de negócios acessem os dados de que precisam de forma autônoma e segura (GADE, 2022).
- Garantir a governança e a conformidade dos dados, aplicando políticas e regras de forma consistente em todo o ambiente de dados (GADE, 2022).

O Data Fabric representa uma mudança de paradigma na gestão de dados, evoluindo de abordagens tradicionais e fragmentadas para um modelo unificado e inteligente.

2.3.1 Definição e Conceito

O Gartner define o Data Fabric como "um conceito de design que atua como uma camada integrada (fabric) de dados e processos de conexão. Essa camada utiliza análises contínuas sobre metadados existentes, detectáveis e inferidos para dar suporte ao design, implantação e utilização de dados integrados e reutilizáveis em todos os ambientes, incluindo plataformas híbridas e multinuvem"(Gartner, 2024). Em outras palavras, o Data Fabric visa conectar e fornecer uma visão holística dos ativos de dados em toda a empresa, utilizando metadados para descoberta, acesso e uso de dados de forma self-service (BARIK, 2022).

O conceito de Data Fabric surgiu com o objetivo de conectar diferentes fontes de dados, em diferentes formatos e plataformas, de forma eficiente (BLOHM et al., 2024). A ideia é criar uma camada de abstração que permita aos usuários acessar e analisar os dados sem se preocupar com a complexidade da infraestrutura subjacente (BARIK, 2022).

O Data Fabric surge como uma evolução das plataformas de dados corporativos, aproveitando capacidades existentes e adicionando funcionalidades para criar uma rede integrada de informações. Ele não exige a consolidação física dos dados, mas utiliza metadados para conectar e permitir acesso a dados em diferentes repositórios, mantendo os sistemas em suas infraestruturas originais. Isso contrasta com plataformas centralizadas, que demandam a movimentação e agregação de dados em um único local (BARIK, 2022).

2.3.2 Arquitetura

A arquitetura do Data Fabric é complexa e não existe um modelo único que atenda a todas as necessidades. No entanto, podemos identificar alguns componentes chave e princípios de design comuns a maioria das implementações.

Componentes da Arquitetura:

- **Camada de Fontes de Dados**: O Data Fabric se conecta a uma variedade de fontes de dados, incluindo bancos de dados relacionais, bancos de dados NoSQL, data lakes, sistemas de arquivos e APIs. O Data Fabric abstrai a complexidade dessas fontes, permitindo que os usuários acessem os dados de forma uniforme, independentemente de sua localização ou formato (BARIK, 2022).
- **Camada de Ingestão e Integração de Dados**: Responsável por coletar, transformar e integrar dados de diferentes fontes. O Data Fabric utiliza uma variedade de técnicas de integração de dados, como virtualização de dados, replicação de dados e ETL (Extract, Transform, Load). A escolha da técnica mais adequada depende das características dos dados, dos requisitos de desempenho e dos custos (BARIK, 2022).
- **Camada de Metadados**: O Data Fabric mantém um catálogo abrangente de metadados que descrevem os dados, incluindo seu significado, relacionamento com outros dados e como podem ser usados. O Data Fabric utiliza inteligência artificial (IA) e machine learning (ML) para automatizar a descoberta, classificação e enriquecimento de metadados. Essa camada é crucial para a descoberta de dados, governança de dados e self-service (BARIK, 2022).
- **Camada de Processamento e Análise de Dados**: O Data Fabric fornece um ambiente para processar e analisar dados, utilizando uma variedade de ferramentas e tecnologias, como Spark, Hadoop e plataformas de machine learning. Essa camada pode ser implementada em diferentes ambientes, como on-premises, cloud ou edge (SHARMA et al., 2023).
- **Camada de Acesso e Consumo de Dados**: O Data Fabric fornece uma interface unificada para acessar e consumir dados, por meio de APIs, ferramentas de visualização de dados e dashboards. Essa camada permite que os usuários de negócios acessem os dados de que precisam de forma self-service, sem a necessidade de envolver a equipe de TI (BARIK, 2022).

Princípios de Design:

- **Descentralização**: O Data Fabric é uma arquitetura descentralizada, o que significa que os dados podem permanecer em seus locais originais, eliminando a necessidade

de criar um repositório centralizado de dados. Isso torna o Data Fabric mais flexível e escalável (SHARMA et al., 2023).

- **Elasticidade e Escalabilidade**: O Data Fabric é projetada para ser elástica e escalável, permitindo que as organizações adicionem ou removam recursos de computação e armazenamento conforme necessário. Isso garante que o Data Fabric possa lidar com o crescimento do volume e da variedade de dados (SHARMA et al., 2023).
- **Segurança e Governança**: O Data Fabric implementa políticas e regras de segurança e governança de dados de forma consistente em todo o ambiente de dados, garantindo a conformidade com as regulamentações (SHARMA et al., 2023).
- **Inteligência Artificial e Automação**: O Data Fabric utiliza IA e ML para automatizar tarefas de gerenciamento de dados, como a descoberta, classificação e enriquecimento de dados. Isso torna o Data Fabric mais eficiente e inteligente, permitindo que as organizações extraiam o máximo valor de seus dados (HECHLER; WEIHRAUCH; 2023).

A arquitetura do Data Fabric é projetada para fornecer uma plataforma unificada e inteligente para gerenciar dados em todo o ecossistema de uma organização. Ao combinar os princípios de descentralização, elasticidade, segurança, governança e inteligência artificial, o Data Fabric capacita as organizações a aproveitar ao máximo o valor de seus dados, impulsionando a tomada de decisões e a inovação.

2.3.3 Quando Utilizar

Após explorar a arquitetura do Data Fabric, é crucial entender em quais cenários e situações essa abordagem se torna especialmente vantajosa. O Data Fabric não é uma solução universal para todos os problemas de dados, mas sim uma ferramenta poderosa que, quando aplicada corretamente, pode gerar grande valor para as organizações.

O Data Fabric é especialmente útil em organizações que enfrentam desafios na gestão de dados, como a unificação de dados altamente inter-relacionados, mas dispersos em diferentes sistemas. Ela atua como uma camada integradora, simplificando o acesso e análise desses dados. Além disso, ajuda a superar barreiras causadas por silos de dados, promovendo uma visão unificada e melhor colaboração entre equipes (BARIK, 2022).

Empresas com múltiplas plataformas de dados, como data lakes e warehouses, podem usá-la para criar um ambiente de dados coeso e gerenciável. Sua arquitetura flexível e escalável permite adaptar-se rapidamente a mudanças nos requisitos de negócios, promovendo agilidade.

O Data Fabric também facilita a democratização dos dados, oferecendo suporte para análises self-service com segurança e governança adequadas. Por fim, possibilita a

aplicação consistente de políticas de governança em todo o ambiente de dados, garantindo qualidade, segurança e conformidade de maneira automatizada. Assim, ela se apresenta como uma solução estratégica para integrar, gerenciar e governar dados em cenários complexos.

2.3.4 Quando Não Utilizar

Embora o Data Fabric apresente um potencial transformador para a gestão de dados em diversos cenários, existem situações em que sua implementação pode não ser a escolha mais adequada. A decisão de adotar uma arquitetura de Data Fabric deve ser baseada em uma análise cuidadosa das necessidades, recursos e desafios da organização.

Um dos cenários em que o Data Fabric pode não ser a melhor opção é quando organizações possui baixo volume de dados com pouca variedade, o Data Fabric podem não justificar o investimento e o esforço de implementação. Soluções de gerenciamento de dados mais simples e tradicionais podem ser suficientes nesses casos (BARIK, 2022).

Outro cenário desafiador é a falta de maturidade em governança de dados. A eficácia do Data Fabric depende de processos bem estabelecidos de qualidade, segurança e conformidade, o que torna sua implementação mais complexa em empresas que ainda não possuem uma base sólida nesse aspecto (GADE, 2022).

Além disso, a adoção dessa arquitetura pode enfrentar desafios importantes. A escassez de profissionais qualificados para projetar, implementar e gerenciar sistemas avançados de Data Fabric pode dificultar sua implementação, levando a atrasos e à dependência de consultores externos. Além disso, a resistência cultural à mudança representa outro obstáculo, especialmente entre colaboradores acostumados a sistemas e processos existentes. Superar esses desafios exige uma gestão de mudanças robusta, que promova a compreensão dos benefícios do Data Fabric e incentive uma cultura de inovação e adaptação dentro das organizações (GADE, 2022).

2.4 Data Lake

Os **data lakes** são **repositórios escaláveis** projetados para armazenar **grandes volumes de dados brutos em seus formatos nativos até que sejam necessários**. Diferentemente de bancos de dados tradicionais, que lidam principalmente com dados estruturados e frequentemente escalam verticalmente, os data lakes foram concebidos para gerenciar o **volume, velocidade e variedade de dados**, incluindo dados estruturados, semiestruturados e não estruturados (FANG, 2021).

Características Principais dos Data Lakes

- **Arquitetura escalável**: Utilizam escalonamento horizontal, distribuindo a carga entre várias máquinas ou nós, o que permite gerenciar grandes volumes de dados difíceis de serem administrados em sistemas tradicionais (GONZALEZ, 2019).

- **Compatibilidade com tecnologias modernas**: São implementados com sistemas de arquivos distribuídos como o **HDFS** ou serviços de armazenamento em nuvem, como **Amazon S3**, **Azure Data Lake Storage Gen2** e **Google Cloud Storage** (VASS; STEWART, 2020).

Vantagens dos Data Lakes

- **Custo-benefício**: Aproveitam hardware comum e armazenamento em nuvem, reduzindo custos em comparação aos data warehouses tradicionais (ZAGAN; DANUBIANU, 2020).

- **Flexibilidade**: Permitem armazenar dados em formatos diversos sem a necessidade de transformações iniciais, facilitando análises futuras (ZAGAN; DANUBIANU, 2020).

- **Escalabilidade**: São projetados para crescer horizontalmente, acomodando volumes crescentes e fontes de dados variadas (ZAGAN; DANUBIANU, 2020).

Desvantagens dos Data Lakes

- **Confiabilidade dos dados**: Sem gestão adequada, podem se transformar em "pântanos de dados"devido à falta de controle de qualidade, gerenciamento de esquemas e proteções transacionais (NUTHALAPATI, 2018).

- **Desempenho**: Consultas em dados brutos podem ser lentas e ineficientes (NUTHALAPATI, 2018).

- **Complexidade de gestão**: Exigem habilidades e ferramentas especializadas para administração e segurança (NUTHALAPATI, 2018).

2.4.1 Tabela Comparativa: Data Lakes vs. Bancos de Dados Tradicionais

Característica	**Data Lakes**	**Bancos de Dados Tradicionais**
Estrutura de Dados	Suporta formatos diversos (estruturados, semiestruturados e não estruturados)	Projetados para dados estruturados
Escalabilidade	Escalabilidade horizontal	Escalabilidade vertical
Custo	Mais econômico com uso de hardware comum e nuvem	Mais caro devido a hardware e software especializados
Flexibilidade	Alta flexibilidade para diferentes formatos e esquemas	Menos flexível, requer esquemas definidos previamente
Confiabilidade	Suscetível a problemas de qualidade sem gestão	Alta confiabilidade com propriedades ACID

2.5 Data Mesh: Um Paradigma para Gestão de Dados Descentralizados

O **Data Mesh** é uma abordagem descentralizada para a arquitetura de dados, introduzida como uma alternativa aos modelos centralizados tradicionais, como **Data Warehouses** e **Data Lakes**. Essa abordagem foca na distribuição de responsabilidades para equipes de domínio, promovendo escalabilidade e autonomia (DEHGHANI, 2020).

2.5.1 O que é Data Mesh?

O conceito de *Data Mesh* se baseia na ideia de tratar dados como produtos gerenciados por equipes de domínio, que são especialistas em seus próprios casos de uso. A arquitetura enfatiza a governança distribuída, onde os nós representam *data products* e as conexões entre eles facilitam a integração (AKHTAR, 2021).

2.5.2 Princípios Fundamentais do Data Mesh

O *Data Mesh* baseia-se em quatro princípios fundamentais:

1. **Propriedade de Dados Descentralizada e Orientada ao Domínio:** As equipes de domínio são responsáveis pelos dados que produzem, garantindo sua qualidade e acessibilidade (DEHGHANI, 2020).

2. **Dados como Produto:** Os dados são tratados como produtos independentes, com seus próprios responsáveis que garantem governança e facilidade de consumo (DEHGHANI, 2020).

3. **Plataforma de Dados Autossuficiente:** Uma infraestrutura centralizada abstrai a complexidade técnica, permitindo que equipes de domínio operem com eficiência (AKHTAR, 2021).

4. **Governança Federada:** Diretrizes centrais são aplicadas localmente pelas equipes, promovendo consistência e autonomia (MARTINEZ, 2022).

2.5.3 Vantagens e Desvantagens do Data Mesh

O *Data Mesh* apresenta vantagens significativas, como escalabilidade horizontal e maior agilidade organizacional. No entanto, também possui desafios, como a necessidade de equipes técnicas autônomas e de uma infraestrutura robusta para suportar a descentralização (JOHNSON, 2021).

2.5.4 Quando Utilizar Data Mesh?

O *Data Mesh* é indicado em:

- **Organizações complexas:** Grandes organizações com múltiplos domínios de dados (DEHGHANI, 2020).
- **Cenários de alta escalabilidade:** Empresas em rápido crescimento de volume de dados (MARTINEZ, 2022).

2.5.5 Quando Não Utilizar Data Mesh?

Evite *Data Mesh* em:

- **Pequenas organizações:** Onde os dados centralizados são mais fáceis de gerenciar (JOHNSON, 2021).
- **Ambientes com baixa maturidade em dados:** A descentralização pode ser excessivamente complexa (AKHTAR, 2021).

2.5.6 Ferramentas Open Source para Data Mesh

Diversas ferramentas podem ser integradas ao *Data Mesh*:

- **Apache Airflow:** Para orquestração de pipelines de dados.
- **Apache Kafka:** Para ingestão e streaming de dados em tempo real.
- **dbt (Data Build Tool):** Para transformação e documentação de dados.
- **Kubernetes:** Para implantação escalável de serviços de dados (SMITH, 2023).

2.5.7 Conclusão

O *Data Mesh* representa uma evolução na arquitetura de dados, oferecendo soluções para os desafios da centralização. No entanto, sua implementação exige uma infraestrutura bem projetada e equipes capacitadas.

2.6 Apache Airflow: Orquestração de Pipelines

O Apache Airflow é uma plataforma de orquestração de workflows de código aberto que permite programar, executar e monitorar pipelines de dados de forma programática. Desenvolvido originalmente pelo Airbnb em 2014 e posteriormente doado à Apache Software Foundation, o Airflow se tornou uma ferramenta essencial para a automação e gerenciamento de fluxos de trabalho complexos em ambientes de processamento de dados (CHANG; SHARMA, 2021).

2.6.1 Conceitos Fundamentais

O Airflow utiliza Directed Acyclic Graphs (DAGs) para representar workflows, onde cada nó é uma tarefa e as arestas definem as dependências entre elas. Esta estrutura permite uma visualização clara do fluxo de trabalho e suas dependências, facilitando o gerenciamento e a manutenção dos pipelines (GARCIA; LUMBRERAS, 2021). Os principais componentes do Airflow incluem os DAGs, que são a representação dos workflows como grafos direcionados acíclicos; os Operators, que definem o que realmente será executado em cada tarefa; as Tasks, que são instâncias parametrizadas de operadores; e as Dependencies, que são as relações entre tarefas que determinam a ordem de execução.

2.6.2 Características Principais

O Apache Airflow oferece diversas características que o tornam uma escolha popular para orquestração de pipelines (KUMAR; SINGH, 2022). Ele suporta a execução distribuída de tarefas através de workers, permite a criação de operadores e hooks personalizados, fornece uma interface intuitiva para monitoramento e gerenciamento de DAGs, integra-se facilmente com sistemas de controle de versão e oferece mecanismos robustos para retry e recuperação de erros.

2.6.3 Arquitetura

A arquitetura do Airflow é composta por vários componentes que trabalham em conjunto (SHARMA; GUPTA, 2021). O Scheduler é responsável por agendar e disparar a execução das tarefas, enquanto o Executor determina como as tarefas serão executadas. O Web Server fornece a interface gráfica para interação com o sistema, e o Metadata

Database armazena metadados sobre DAGs, execuções e configurações. Os Workers são os responsáveis por executar as tarefas em ambientes distribuídos.

2.6.4 Benefícios e Limitações

O Apache Airflow possui diversos benefícios, incluindo a configuração como código (Configuration as Code), monitoramento e logging abrangentes, uma comunidade ativa e um grande ecossistema de plugins, além do suporte a múltiplos executores e provedores de nuvem. No entanto, ele também apresenta algumas limitações, como a complexidade inicial de configuração, o consumo significativo de recursos em ambientes grandes e a curva de aprendizado íngreme para recursos avançados.

2.6.5 Melhores Práticas

Para uma utilização efetiva do Airflow, algumas práticas são recomendadas (FOUNDATION, 2023). É importante criar tarefas atômicas e idempotentes, garantir que múltiplas execuções produzam o mesmo resultado, manter documentação clara e atualizada dos DAGs, implementar estratégias robustas de retry e fallback, e configurar alertas e métricas adequadas.

2.7 Apache Spark

Apache Spark é um framework de computação distribuída de código aberto, projetado para processamento de dados em larga escala. Desenvolvido originalmente na UC Berkeley, o Spark se destaca por sua velocidade e capacidade de processar grandes volumes de dados, oferecendo uma interface unificada para diferentes tipos de processamento (ZAHARIA; XIN; WENDELL, 2016).

2.7.1 Arquitetura e Componentes

A arquitetura do Spark é construída em torno do conceito de Resilient Distributed Datasets (RDDs), que são coleções distribuídas e tolerantes a falhas de objetos (KARAU; WARREN, 2023b). Os principais componentes incluem:

- **Spark Core**: Engine de processamento distribuído que gerencia a distribuição de tarefas e coordenação de processos.
- **Spark SQL**: Módulo para processamento de dados estruturados, oferecendo interface SQL e programática.
- **Spark Streaming**: Permite o processamento de dados em tempo real.

- **MLlib**: Biblioteca de machine learning distribuída.
- **GraphX**: API para computação de grafos distribuídos.

2.7.2 Processamento de Dados

O Spark utiliza uma abordagem de computação em memória que o torna significativamente mais rápido que paradigmas anteriores como MapReduce (CHAMBERS; ZAHARIA, 2023). As principais características incluem:

- Processamento em memória para maior velocidade
- Lazy evaluation para otimização de operações
- DAG (Directed Acyclic Graph) para planejamento de execução
- Tolerância a falhas através de linhagem de RDDs

2.7.3 Spark SQL e DataFrames

Spark SQL introduz abstrações modernas como DataFrames e Datasets, que oferecem uma interface mais intuitiva para manipulação de dados estruturados (ARMBRUST; XIN, 2023). Exemplo de código:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("ExemploSparkSQL") \
    .getOrCreate()

df = spark.read.csv("dados.csv")
df.createOrReplaceTempView("tabela")

resultado = spark.sql("""
    SELECT coluna1, COUNT(*)
    FROM tabela
    GROUP BY coluna1
""")
```

2.7.4 Integração com Ecossistema Big Data

O Spark trabalha bem com várias outras ferramentas importantes de big data (WENDELL; ZAHARIA, 2023). Ele usa o Hadoop YARN para controlar melhor os re-

cursos do sistema, e se conecta ao Apache Kafka para processar dados em tempo real. O Spark também funciona junto com o Delta Lake para garantir que os dados sejam guardados de forma segura e organizada. Por fim, sua integração com Kubernetes permite que as aplicações rodem em qualquer ambiente de nuvem de forma mais fácil e flexível.

2.7.5 Performance e Otimização

Para fazer o Spark funcionar melhor e mais rápido, existem algumas técnicas importantes (KARAU; WARREN, 2023a). É preciso distribuir os dados de forma inteligente entre os computadores do sistema e configurar corretamente a memória disponível. Também é importante guardar na memória os dados que são usados com mais frequência, e organizar bem as operações que juntam diferentes conjuntos de dados. Dessa forma, o sistema consegue processar grandes volumes de informação de maneira mais eficiente.

2.8 Kubernetes: Orquestração de Containers para Escalabilidade

O Kubernetes é uma plataforma de código aberto para orquestração de containers, originalmente desenvolvida pelo Google e posteriormente doada à Cloud Native Computing Foundation (CNCF). A plataforma automatiza o deploy, a escalabilidade e o gerenciamento de aplicações containerizadas, consolidando-se como um padrão de facto na indústria para orquestração de containers em larga escala (BURNS; BEDA; HIGHTOWER, 2019).

2.8.1 Arquitetura do Kubernetes

A arquitetura do Kubernetes é baseada em um modelo master-node, também conhecido como control plane e worker nodes, onde cada componente desempenha funções específicas (VERMA; PEDROSA; KORUPOLU, 2020).

O **Control Plane (Master)** é composto por quatro componentes principais. O **kube-apiserver** expõe a API do Kubernetes e atua como frontend do cluster. O **etcd** é um armazenamento consistente e distribuído para todos os dados do cluster. O **kube-scheduler** é responsável por agendar pods nos nodes do cluster, enquanto o **kube-controller-manager** executa os processos de controle do cluster.

Nos **Worker Nodes**, o **kubelet** atua como agente, garantindo que os containers estejam rodando em um pod. O **kube-proxy** mantém as regras de rede nos nodes, e o **Container Runtime** é o software responsável por executar os containers.

2.8.2 Conceitos Fundamentais

O Kubernetes opera com conceitos fundamentais que formam a base de sua funcionalidade (HIGHTOWER; BURNS; BEDA, 2019). O **Pod** é a menor unidade implantável no Kubernetes, podendo conter um ou mais containers. O **Service** é uma abstração que define um conjunto lógico de pods e uma política de acesso. O **Deployment** gerencia a criação e atualização de réplicas de pods, enquanto o **StatefulSet** é utilizado para gerenciar aplicações stateful com identidades persistentes. Além disso, **ConfigMaps** e **Secrets** são utilizados para gerenciar configurações e dados sensíveis, respectivamente.

2.8.3 Recursos de Escalabilidade

O Kubernetes oferece mecanismos robustos para garantir a escalabilidade das aplicações (LUKSA, 2021). A **Escalabilidade Horizontal** é alcançada por meio do **Horizontal Pod Autoscaling (HPA)**, que ajusta automaticamente o número de pods com base em métricas de utilização, e do **Cluster Autoscaling**, que adiciona ou remove nodes automaticamente conforme a demanda. O **Load Balancing** distribui o tráfego entre múltiplas instâncias de aplicação, garantindo uma distribuição equilibrada da carga.

Para **Alta Disponibilidade**, o Kubernetes implementa mecanismos como **Self-healing**, que reinicia containers que falham e substitui pods em caso de indisponibilidade, e **Rolling Updates**, que permitem atualizações sem downtime. Além disso, o **Multi-zone Deployment** distribui a carga entre diferentes zonas de disponibilidade, aumentando a resiliência do sistema.

2.8.4 Benefícios e Desafios

O Kubernetes oferece diversos benefícios, como **Portabilidade**, funcionando em qualquer ambiente (on-premise, nuvem ou híbrido), **Escalabilidade** com gerenciamento automático de recursos, **Resiliência** com recuperação automática de falhas, e uma abordagem **Declarativa**, onde a configuração é tratada como código (Infrastructure as Code).

No entanto, também apresenta desafios significativos. A **Complexidade** da plataforma resulta em uma curva de aprendizado íngreme. O **Overhead Operacional** exige uma equipe especializada para manutenção, e os **Custos** podem ser elevados em ambientes pequenos. Além disso, a **Segurança** requer atenção especial, com a necessidade de configurações robustas para proteger o cluster.

2.8.5 Melhores Práticas

Para uma implementação efetiva do Kubernetes, é recomendado seguir boas práticas (ARUNDEL; DOMINGUS, 2021). A **Segurança** deve ser priorizada, com a im-

plementação de políticas de segurança e network policies. O **Monitoramento** pode ser realizado com ferramentas como Prometheus e Grafana, enquanto o **Logging** deve ser centralizado com soluções como o ELK Stack. O **Gerenciamento de Recursos** deve incluir a definição de limites e requisitos de recursos, e estratégias de **Backup** e disaster recovery devem ser implementadas para garantir a continuidade dos serviços.

2.9 FastAPI: Desenvolvimento de APIs para Integração de Dados

FastAPI é um framework moderno e de alto desempenho para construção de APIs com Python, baseado em padrões como OpenAPI (anteriormente conhecido como Swagger) e JSON Schema. Desenvolvido por Sebastián Ramírez, o FastAPI se destaca por sua velocidade de execução, facilidade de uso e geração automática de documentação interativa (RAMíREZ, 2023).

2.9.1 Características Fundamentais

O FastAPI possui características que o tornam particularmente adequado para integração de dados (RAMíREZ; GONZáLEZ, 2022). A performance do FastAPI é notável porque ele é baseado no ASGI (Asynchronous Server Gateway Interface) e utiliza o Starlette. Além disso, ele aproveita a tipagem estática do Python para fazer validação automática, gera documentação interativa OpenAPI e ReDoc automaticamente, oferece suporte nativo a programação assíncrona através do async/await, e utiliza o Pydantic para validação e serialização de dados.

2.9.2 Arquitetura e Componentes

A arquitetura do FastAPI é construída sobre componentes modernos e eficientes (KUMAR; SINGH, 2023b). Entre os componentes principais estão o Starlette, que é um framework ASGI leve para aplicações assíncronas, o Pydantic, uma biblioteca para validação de dados usando type annotations, o Uvicorn, um servidor ASGI de alto desempenho, e o OpenAPI, uma especificação para documentação de APIs RESTful.

2.9.3 Integração de Dados com FastAPI

O FastAPI oferece recursos robustos para integração de dados (WILSON, 2023). Para a modelagem de dados, por exemplo, é possível usar o Pydantic para criar modelos de dados de forma simples. Aqui está um exemplo de modelo de dados:

```
from pydantic import BaseModel

class DataModel(BaseModel):
```

```
id: int
name: str
value: float
```

Em relação aos endpoints para manipulação de dados, o FastAPI facilita a implementação de operações CRUD (Create, Read, Update, Delete), o processamento eficiente de grandes conjuntos de dados, suporte a streaming de dados em tempo real, e o processamento assíncrono de tarefas pesadas.

2.9.4 Segurança e Autenticação

O FastAPI oferece diversos mecanismos de segurança (BROWN; JOHNSON, 2023). Ele suporta OAuth2 com diferentes fluxos, integra-se facilmente com JSON Web Tokens (JWT), permite autenticação via chaves de API, e tem configuração flexível de Cross-Origin Resource Sharing (CORS).

2.9.5 Performance e Escalabilidade

O FastAPI se destaca em termos de performance (PETERSON; LEE, 2023). Ele oferece baixa latência em requisições HTTP, alta capacidade de processamento concorrente, uso eficiente de recursos computacionais e excelente desempenho em operações que dependem de I/O.

2.9.6 Melhores Práticas

Para um desenvolvimento eficiente com FastAPI, algumas melhores práticas são recomendadas (MARTINEZ; THOMPSON, 2023). É importante estruturar bem o projeto, com organização modular do código e separação clara de responsabilidades. Uso de dependency injection também é recomendado. No tratamento de erros, é crucial implementar exception handlers, fornecer respostas de erro padronizadas e realizar logging adequado. A documentação também deve ser clara, com exemplos de uso e descrições detalhadas dos modelos de dados.

2.9.7 Limitações e Considerações

Apesar de suas muitas vantagens, o FastAPI tem algumas limitações (KUMAR; SINGH, 2023b). Ele exige conhecimento em programação assíncrona, o que pode aumentar a curva de aprendizado para recursos avançados. Além disso, pode adicionar complexidade extra em projetos muito grandes e ainda depende do ecossistema Python.

2.10 Pipelines de Dados no Diagnóstico de Câncer de Pele

Criar pipelines de dados para o diagnóstico de câncer de pele é um ponto muito importante onde a engenharia de dados e a medicina se encontram. Esses pipelines são feitos para processar, analisar e gerenciar muitos dados de imagens dermatológicas, ajudando a diagnosticar diferentes tipos de câncer de pele de maneira rápida e precisa (ZHANG; WANG; SINGH, 2021).

2.10.1 Arquitetura do Pipeline

A arquitetura de um pipeline de dados para diagnóstico de câncer de pele geralmente tem várias etapas conectadas entre si (ESTEVA; TOPOL, 2022). Primeiro, na aquisição de dados, são capturadas imagens dermatológicas, digitalizados os prontuários médicos e coletados os metadados clínicos. Depois, no pré-processamento, as imagens são normalizadas, os ruídos removidos, as lesões segmentadas e os dados aumentados. No processamento, são extraídas características das imagens, analisadas texturas e cores, e classificadas as lesões. Por fim, no armazenamento e recuperação, as imagens são guardadas em um banco de dados, utilizando sistemas de cache e realizando backup e redundância.

2.10.2 Componentes Essenciais

Os componentes principais do pipeline incluem uma infraestrutura de dados robusta e modelos de processamento avançados (KUMAR; PATEL, 2023). A infraestrutura de dados tem uma camada de armazenamento distribuído para grandes volumes de imagens, recursos computacionais para processar essas imagens, e ferramentas para análise e visualização. Quanto aos modelos de processamento, são usadas redes neurais convolucionais, algoritmos de segmentação e modelos de classificação para analisar e interpretar as imagens dermatológicas.

2.10.3 Fluxo de Dados e Processamento

O fluxo de dados em um pipeline de diagnóstico de câncer de pele segue um processo bem organizado (WANG; CHEN, 2023). Primeiro, na ingestão de dados, as imagens dermatoscópicas são coletadas, verificadas quanto à qualidade e padronizadas no formato. No pré-processamento das imagens, o contraste e o brilho são normalizados, os artefatos removidos e a região de interesse segmentada. Na análise e classificação, são extraídas características, aplicados modelos de machine learning e deep learning, e geradas predições.

2.10.4 Garantia de Qualidade

A qualidade dos dados é crucial para o diagnóstico preciso (SMITH; BROWN, 2023a). Para garantir a qualidade, é realizado um controle rigoroso sobre a qualidade das imagens, incluindo verificação de resolução, análise de contraste e validação de formato. A validação de dados assegura a completude, consistência dos metadados e integridade dos dados, assegurando que as informações utilizadas no diagnóstico sejam precisas e confiáveis.

2.10.5 Desafios e Considerações

O desenvolvimento de pipelines para diagnóstico de câncer de pele enfrenta diversos desafios (CHEN; MARTINEZ, 2023). Entre os desafios técnicos, destacam-se a variabilidade na qualidade das imagens, necessidade de processamento em tempo real e requisitos de armazenamento. Os desafios clínicos incluem a variabilidade das lesões, necessidade de alta precisão e interpretabilidade dos resultados. Adicionalmente, considerações éticas envolvem a privacidade dos pacientes, segurança dos dados e conformidade regulatória. Manter a qualidade dos dados é muito importante para um diagnóstico preciso (SMITH; BROWN, 2023a). Por isso, um controle rigoroso é feito sobre a qualidade das imagens, verificando resolução, contraste e formato. A validação dos dados assegura que estão completos, consistentes nos metadados e íntegros, garantindo que as informações usadas no diagnóstico sejam precisas e confiáveis.

2.10.6 Desafios e Considerações

Desenvolver pipelines para o diagnóstico de câncer de pele tem vários desafios (CHEN; MARTINEZ, 2023). Entre os desafios técnicos, estão a variabilidade na qualidade das imagens, a necessidade de processamento em tempo real e os requisitos de armazenamento. Os desafios clínicos incluem a variabilidade das lesões, a necessidade de alta precisão e a interpretabilidade dos resultados. Além disso, as considerações éticas envolvem a privacidade dos pacientes, a segurança dos dados e a conformidade com as regulamentações.

2.11 Fluxo de Dados em Arquitetura de Data Fabric

Em uma arquitetura de Data Fabric, o fluxo de dados é projetado para proporcionar uma integração contínua e inteligente de dados através de múltiplos ambientes, garantindo consistência, governança e acessibilidade (SHARMA et al., 2023). Este modelo representa uma evolução significativa na forma como as organizações gerenciam e utilizam seus dados, especialmente em ambientes híbridos e distribuídos. A arquitetura

de Data Fabric permite que dados de diversas fontes sejam integrados, processados e disponibilizados de maneira eficiente, facilitando a tomada de decisões baseada em dados e aumentando a agilidade organizacional.

2.11.1 Componentes do Fluxo de Dados

O fluxo de dados em uma arquitetura Data Fabric é composto por diversos componentes interconectados (HECHLER; WEIHRAUCH; WU, 2023). A camada de ingestão é responsável por coletar dados de diversas fontes e integrá-los ao sistema, utilizando conectores de fonte de dados, mecanismos de streaming, APIs de integração e protocolos de comunicação. Em seguida, a camada de processamento realiza transformações, validações e enriquecimento dos dados para torná-los úteis e consistentes. Este processo inclui a transformação de dados, validação e limpeza, enriquecimento de dados e normalização. A camada de armazenamento é onde os dados processados são armazenados de forma estruturada e acessível. Esta camada inclui data lakes, data warehouses, bancos de dados distribuídos e sistemas de cache, que armazenam grandes volumes de dados brutos, dados estruturados prontos para análise e dados distribuídos em múltiplos locais, além de otimizar o acesso rápido a dados frequentemente utilizados.

2.11.2 Padrões de Fluxo de Dados

Os padrões de fluxo de dados em uma arquitetura Data Fabric podem ser categorizados em diferentes tipos, cada um servindo a propósitos específicos na gestão e análise de dados (BLOHM et al., 2024). O fluxo batch envolve o processamento em lotes programados, onde grandes volumes de dados são executados em intervalos regulares, realizando operações de transformação de dados elaboradas, agregações de grande escala e cargas incrementais. Por outro lado, o fluxo real-time se concentra na ingestão contínua de dados em tempo real, analisando e processando dados imediatamente após a ingestão, permitindo a análise contínua e a geração de alertas e notificações. O fluxo híbrido combina elementos de ambos os tipos de fluxo, utilizando processamento em tempo real e em lotes conforme necessário. Essa abordagem permite a integração de processamento em tempo real e em lotes, análise preditiva utilizando dados históricos e em tempo real, e a detecção de anomalias.

2.11.3 Gestão e Controle do Fluxo

A gestão eficiente do fluxo de dados é crucial para o sucesso da arquitetura de Data Fabric (GADE, 2022). A orquestração coordena os diversos processos envolvidos no fluxo de dados, incluindo o agendamento de jobs, gerenciamento de dependências, controle de concorrência e recuperação de falhas. O monitoramento acompanha a performance e

o estado dos processos de dados, avaliando a eficiência dos processos através de métricas de performance, logs de execução, rastreamento de dados e alertas de anomalias. A governança define políticas e práticas para garantir a qualidade e segurança dos dados, controlando o acesso aos dados, assegurando que os dados atendam às regulamentações, mantendo a integridade e precisão dos dados e rastreando a origem e transformação dos dados.

2.11.4 Integração e Interoperabilidade

A integração eficiente é fundamental para o sucesso do Data Fabric, permitindo que diferentes sistemas e fontes de dados trabalhem juntos de maneira harmoniosa (BARIK, 2022). A conectividade estabelece conexões entre diversos sistemas e fontes de dados, utilizando APIs padronizadas, protocolos de comunicação e adaptadores de fonte de dados. A transformação ajusta e converte dados para torná-los compatíveis entre sistemas, realizando mapeamento de dados, conversão de formatos e normalização de esquemas. A sincronização assegura que os dados estejam atualizados e consistentes, garantindo consistência de dados, replicação de dados entre sistemas para redundância e resolução de conflitos entre diferentes versões de dados.

2.11.5 Desafios e Considerações

A implementação do fluxo de dados em Data Fabric apresenta diversos desafios que devem ser considerados para garantir seu sucesso (ADDAGADA, 2022). Os desafios técnicos envolvem questões relacionadas à infraestrutura e tecnologia, como a complexidade da infraestrutura, latência em ambientes distribuídos e escalabilidade. Os desafios operacionais estão relacionados à operação contínua e manutenção do sistema, incluindo manutenção contínua, gerenciamento de recursos e resolução de problemas. Além disso, há desafios organizacionais que envolvem a adaptação da organização à nova arquitetura, como adaptação cultural, capacitação de equipes e gestão de mudanças.

2.12 Ingestão de Dados e Preprocessamento de Imagens

A ingestão de dados e o preprocessamento de imagens são etapas cruciais em sistemas de análise de imagens médicas, especialmente no contexto do diagnóstico de câncer de pele. Essas etapas são fundamentais para estabelecer a base para análises posteriores e influenciam diretamente a qualidade dos resultados obtidos (GARCIA; RODRIGUEZ, 2023).

2.12.1 Ingestão de Dados

O processo de ingestão de dados envolve a coleta e integração de diferentes tipos de dados médicos provenientes de várias fontes (KUMAR; SINGH, 2023a). Entre essas fontes, destacam-se as imagens dermatoscópicas, que podem incluir fotografias clínicas de alta resolução, imagens microscópicas digitalizadas e imagens termográficas. Além das imagens, dados clínicos associados também desempenham um papel essencial nesse processo. Informações como o histórico médico do paciente, diagnósticos anteriores e metadados das imagens fornecem o contexto necessário para análises mais robustas e acuradas.

2.12.2 Preprocessamento de Imagens

O preprocessamento é uma etapa indispensável para garantir a qualidade e a padronização das imagens utilizadas (SMITH; BROWN, 2023b). Nesse contexto, a normalização de imagens é realizada para ajustar contraste, corrigir brilho, padronizar tamanhos e normalizar cores, assegurando consistência nos dados analisados. Além disso, técnicas de aprimoramento como redução de ruído, realce de bordas, correção de iluminação e aplicação de filtros de suavização contribuem para melhorar a qualidade visual das imagens, facilitando a identificação de características importantes. Outro aspecto fundamental do preprocessamento é a segmentação, que visa identificar regiões de interesse nas imagens. Essa etapa é crucial para separar a lesão da pele saudável e remover artefatos que poderiam interferir na análise subsequente.

2.12.3 Pipeline de Processamento

O pipeline de processamento segue uma sequência estruturada de operações que asseguram a eficiência e a qualidade do fluxo de trabalho (WANG; CHEN, 2023). Inicialmente, é realizada uma validação inicial para verificar o formato dos arquivos, controlar a qualidade das imagens e validar os metadados associados. Em seguida, ocorre o preprocessamento básico, que abrange operações como redimensionamento, normalização e conversão de formato das imagens. Por fim, o preprocessamento avançado inclui a aplicação de técnicas mais sofisticadas, como segmentação, extração de características e augmentação de dados, que enriquecem o conjunto de informações disponível para análise.

2.12.4 Técnicas de Augmentação de Dados

A augmentação de dados é uma estratégia amplamente utilizada para melhorar o treinamento de modelos, especialmente quando o volume de dados disponíveis é limitado (ZHANG; LI, 2023). Diversas transformações geométricas, como rotação, reflexão, escalonamento e translação, são aplicadas para ampliar a variedade do conjunto de imagens. Além disso, transformações de intensidade, como ajustes de contraste, alterações

de brilho, aplicação de ruído gaussiano e mudanças na saturação, adicionam variações relevantes às imagens, promovendo a robustez do modelo em diferentes cenários.

2.12.5 Controle de Qualidade

O controle de qualidade desempenha um papel crucial em todas as etapas do processo de ingestão e preprocessamento (CHEN; WANG, 2023). Métricas como resolução da imagem, relação sinal-ruído, nitidez e contraste são avaliadas regularmente para garantir a consistência e a qualidade dos dados. O processo de validação é composto por verificações automáticas, revisões manuais e testes de consistência, que asseguram que as imagens processadas atendam aos critérios exigidos para análises confiáveis.

2.13 MinIO Object Storage

O MinIO é uma solução de armazenamento em objetos projetada para alta performance, escalabilidade e segurança. Licenciado como software open source, ele é compatível com a API do Amazon S3 e é amplamente utilizado para implementar plataformas de Armazenamento como Serviço (STaaS). A crescente demanda por aplicações como Big Data, IoT e Inteligência Artificial impulsiona a necessidade de soluções de armazenamento escaláveis e econômicas, área em que o MinIO se destaca (BUILD. . . , 2019).

2.13.1 Arquitetura e Benefícios

O MinIO adota uma abordagem simplificada para fornecer um sistema de armazenamento eficiente, robusto e de alta disponibilidade. Sua arquitetura em camadas é construída para suportar desde pequenas implementações até clusters globais federados. Entre suas principais características estão (BUILD. . . , 2019):

- Desempenho: Utilizando hardware otimizado, como processadores Intel Xeon e unidades SSD NVMe, o MinIO entrega altas taxas de transferência de dados, permitindo atender às demandas de aplicações nativas em nuvem com baixa latência e alta taxa de transferência.

- Escalabilidade Linear: Clusters do MinIO podem ser ampliados de forma linear, adicionando novos nós sem perda de desempenho, o que é essencial para cenários de armazenamento em larga escala.

- Alta Disponibilidade e Tolerância a Falhas: Com código de apagamento e verificações de proteção contra bitrot, o sistema é projetado para suportar falhas de hardware sem interrupção dos serviços.

- Segurança: Oferece criptografia por objeto com chaves únicas, além de suporte a gerenciamento externo de chaves e modos WORM (Write Once, Read Many).
- Simplicidade Operacional: O design minimalista do MinIO facilita a instalação, configuração e manutenção, reduzindo a necessidade de administração contínua.
- Compatibilidade com Kubernetes: A integração com orquestradores de containers permite a gestão eficiente de recursos, consolidando o MinIO como uma solução nativa para ambientes de nuvem.

2.13.2 Relevância no Cenário Atual de Armazenamento em Nuvem

O MinIO Object Storage reflete a evolução do armazenamento de dados, alinhando-se às necessidades modernas de computação em nuvem, escalabilidade e gestão eficiente de recursos. A literatura enfatiza que soluções como o MinIO não apenas preenchem lacunas tecnológicas, mas também democratizam o acesso ao armazenamento avançado, oferecendo uma alternativa competitiva às grandes provedoras de serviços em nuvem. Seu papel no ecossistema de armazenamento é crucial para suportar o crescimento exponencial de dados, previsto para ultrapassar 163 Zettabytes até 2025 (BUILD. . . , 2019).

2.14 Delta Lake

Delta Lake é uma camada de armazenamento de código aberto que combina as melhores características dos data warehouses e data lakes, formando a base para a arquitetura Lakehouse. Criado para solucionar problemas de confiabilidade e escalabilidade de dados, o Delta Lake suporta transações ACID, manipulação escalável de metadados e a unificação de processamento de dados em lote e streaming. Ele é amplamente utilizado em ambientes de big data e inteligência artificial devido à sua capacidade de gerenciar grandes volumes de dados com consistência e performance (LEE et al., 2024).

2.14.1 Arquitetura e Benefícios

O Delta Lake foi concebido para superar limitações de arquiteturas tradicionais de data lakes, que muitas vezes careciam de garantias transacionais e governança de dados. A arquitetura Lakehouse, da qual o Delta Lake faz parte, une a escalabilidade e flexibilidade dos data lakes com a governança e otimização de performance dos data warehouses. Entre suas principais características estão (LEE et al., 2024):

- Transações ACID: Garantem integridade de dados em operações concorrentes.
- Time Travel: Permite consultar versões anteriores dos dados para auditorias ou recuperação.

- Unificação de Processamento: Suporte simultâneo a cargas de trabalho em lote e streaming.
- Evolução e Imposição de Esquemas: Garante consistência estrutural nos dados.
- Fonte Única de Verdade: Mantém um log transacional que organiza todas as alterações de dados.

2.14.2 Relevância no Cenário Atual de Big Data

No cenário atual, caracterizado pela explosão de dados e pela crescente demanda por análise em tempo real, o Delta Lake desempenha um papel central ao permitir que organizações adotem arquiteturas de dados modernas, como o Lakehouse 2. Essa abordagem resolve limitações de sistemas tradicionais, que frequentemente lidam com silos de dados, altos custos de manutenção e falta de flexibilidade para suportar aplicações avançadas como inteligência artificial e aprendizado de máquina.

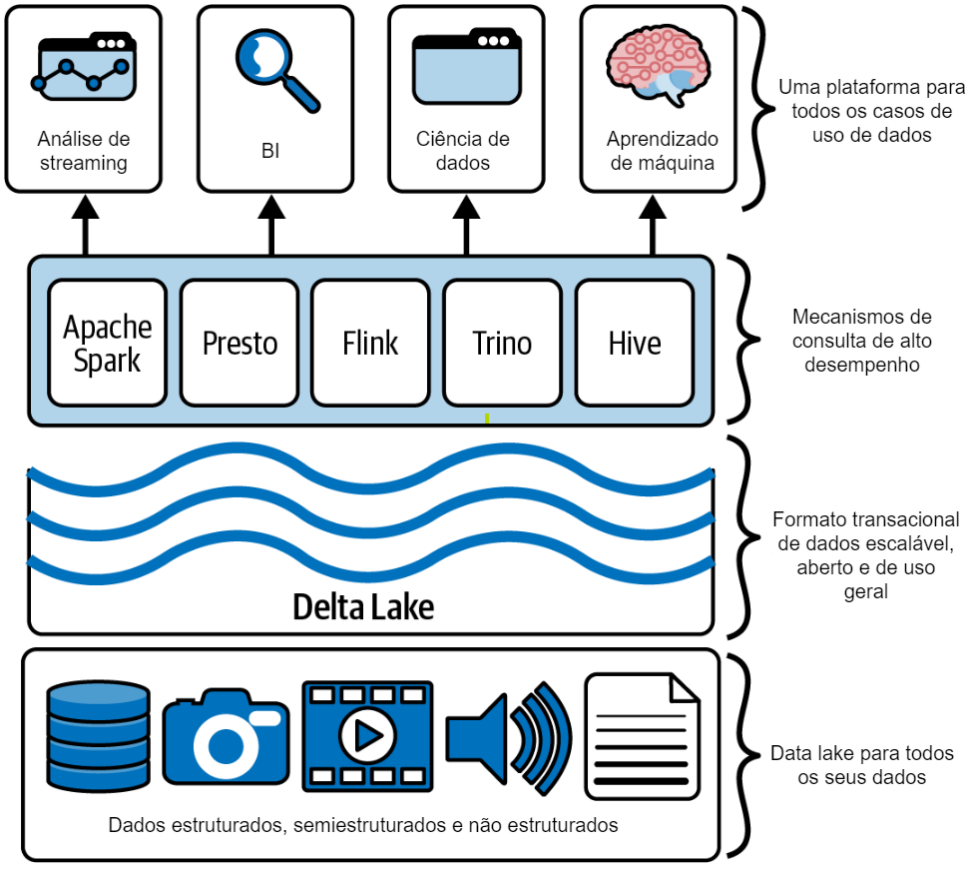

Figura 2 – Delta Lake: Formato transacional escalável e de uso geral para Lakehouse (LEE et al., 2024)

2.15 Related Works

2.15.1 In Search of Big Medical Data Integration Solutions - A Comprehensive Survey

A integração de dados médicos, especialmente em oncologia, enfrenta desafios devido à heterogeneidade e complexidade dos dados, incluindo imagens médicas, informações genômicas e registros eletrônicos de saúde (EHRs). Abordagens como data lakes e data warehouses modernos têm se destacado na consolidação de grandes volumes de dados estruturados e não estruturados, possibilitando o armazenamento e processamento eficiente de informações diversas, como imagens de ressonância magnética e dados genômicos. Ferramentas baseadas em Hadoop, Apache Hive e frameworks semelhantes são amplamente utilizadas para criar plataformas escaláveis e integradas, promovendo análises mais eficazes, como na previsão de doenças crônicas e prevenção de condições como obesidade. (DHAYNE et al., 2019)

Adicionalmente, soluções baseadas em web semântica e aprendizado de máquina vêm contribuindo significativamente para melhorar a interoperabilidade e a análise de dados médicos. Tecnologias como Linked Open Data (LOD) e ontologias médicas (SNOMED CT e UMLS) aprimoram a semântica dos dados, enquanto modelos de aprendizado profundo (deep learning) avançam na análise de imagens médicas, auxiliando na detecção precoce de doenças e personalização de tratamentos. Iniciativas como o iManageCancer exemplificam o uso de data warehouses semânticos para integrar e analisar dados de forma avançada, reforçando o potencial dessas abordagens no apoio à saúde pública e à medicina personalizada. (DHAYNE et al., 2019)

2.15.2 What Clinics Are Expecting From Data Scientists? A Review on Data Oriented Studies Through Qualitative and Quantitative Approaches

A integração de análise de dados em ambientes clínicos, sob o paradigma de "Connected Health"(CH), está transformando a forma como dados médicos são utilizados para melhorar a tomada de decisão e o manejo de doenças. A "Translational Medicine"(TM) atua como um elo essencial entre cientistas de dados e profissionais de saúde, empregando análises preditivas de prontuários eletrônicos e resultados laboratoriais para identificar riscos e aprimorar diagnósticos. Embora abordagens quantitativas sejam predominantes, a combinação com técnicas qualitativas tem demonstrado maior eficácia em áreas como a interpretação de narrativas de pacientes e análise de padrões complexos, especialmente em contextos de doenças como câncer, onde fontes de dados variados, como imagens médicas e genômicas, precisam ser integradas. (XU et al., 2019)

Apesar dos avanços, desafios éticos e técnicos, como interoperabilidade e privaci-

dade, permanecem obstáculos significativos. Padrões como HL7 e iniciativas como "Big Data to Knowledge"(BD2K) têm promovido o uso de big data biomédico para superar essas barreiras. A integração de abordagens qualitativas e quantitativas na análise de dados de câncer, aliada à interoperabilidade, representa uma oportunidade crucial para melhorar diagnósticos precoces e tratamentos personalizados. O desenvolvimento de arquiteturas de dados robustas e interdisciplinares continua sendo um foco promissor para avanços em sistemas de saúde conectados. (XU et al., 2019)

3 Metodologia

3.1 Estudo de Caso

O estudo de caso é uma metodologia de pesquisa que permite uma análise profunda de um fenômeno dentro de seu contexto real. Este método é especialmente útil para explorar e entender questões complexas e contextuais que não podem ser facilmente quantificadas. No presente trabalho, optamos por um estudo de caso devido à necessidade de uma compreensão detalhada e contextualizada do uso de arquiteturas de Data Fabric para salvar imagens de câncer de pele.

3.2 Perguntas de Pesquisa

As perguntas de pesquisa deste estudo de caso foram formuladas para investigar como a arquitetura de Data Fabric pode ser usada no armazenamento e processamento de imagens de câncer de pele. As perguntas específicas são:

1. **Como a arquitetura de Data Fabric pode ser implementada para melhorar o armazenamento e o processamento de imagens de câncer de pele?**
    - Exemplo: Quais são os componentes essenciais para uma arquitetura de Data Fabric eficaz nesse caso?
2. **Quais são os desafios enfrentados na implementação de uma arquitetura de Data Fabric para imagens médicas?**
    - Exemplo: Quais são as principais dificuldades técnicas e operacionais encontradas durante a implementação?
3. **De que maneira a arquitetura de Data Fabric impacta a eficiência do processamento de imagens de câncer de pele?**
    - Exemplo: Como a utilização de Data Fabric pode influenciar a velocidade e a precisão do processamento dessas imagens?
4. **Quais são as percepções dos profissionais de saúde sobre a eficácia da arquitetura de Data Fabric no armazenamento de imagens de câncer de pele?**
    - Exemplo: Como os profissionais avaliam a usabilidade e os benefícios dessa tecnologia?

5. **Quais são as melhores práticas para garantir a segurança e a privacidade dos dados armazenados em uma arquitetura de Data Fabric?**

    - Exemplo: Quais medidas são recomendadas para proteger as informações sensíveis dos pacientes?

Ao responder essas perguntas, o estudo de caso pretende fornecer uma compreensão detalhada e contextualizada do tema, contribuindo para a literatura existente e oferecendo insights práticos para os profissionais da área.

4 Projeto

4.1 Requisitos

Para o desenvolvimento do projeto de uma arquitetura de Data Fabric para salvar imagens de câncer de pele, foram definidos os seguintes requisitos:

4.1.1 Requisitos Funcionais

ID	Requisito
RFSO01	O sistema deve ser capaz de armazenar grandes volumes de imagens de câncer de pele.
RFSO02	O sistema deve permitir o processamento eficiente das imagens armazenadas.
RFSO03	O sistema deve garantir a segurança e a privacidade dos dados dos pacientes.
RFSO04	Os dados devem estar disponíveis para acesso por pesquisadores e profissionais de saúde de forma segura.
RFSO05	O sistema deve ser escalável para acomodar o aumento no volume de dados ao longo do tempo.
RFSO06	O sistema deve permitir o compartilhamento seguro de dados utilizando Delta Sharing.
RFSO07	O sistema deve permitir a integração de dados de diferentes fontes de forma unificada.
RFSO08	O sistema deve oferecer uma camada de virtualização para consulta de dados distribuídos.
RFSO09	O sistema deve permitir a governança centralizada dos dados, com controle de acesso baseado em papéis.
RFSO10	O sistema deve suportar análises em tempo real para dados armazenados e em trânsito.
RFSO11	O sistema deve possuir uma camada de metadados para catalogação de imagens e dados relacionados.
RFSO12	O sistema deve fornecer APIs para facilitar a integração com ferramentas de visualização de dados.

Tabela 1 – Tabela de Requisitos Funcionais

4.1.2 Requisitos Não Funcionais

ID	Requisito
RNFSO01	O sistema deve ser capaz de processar e recuperar dados rapidamente.
RNFSO02	O sistema deve garantir a integridade e a consistência dos dados armazenados.
RNFSO03	A interface do sistema deve ser intuitiva e fácil de usar para os profissionais de saúde.
RNFSO04	O sistema deve ser de fácil manutenção e atualização.
RNFSO05	A arquitetura deve ser modular para facilitar a expansão e substituição de componentes.
RNFSO06	O sistema deve oferecer suporte a padrões de interoperabilidade, como FHIR.
RNFSO07	O sistema deve garantir compatibilidade com regulamentações de proteção de dados, como LGPD.

Tabela 2 – Tabela de Requisitos Não Funcionais

4.1.3 Histórias de Usuário

Requisitos	Descrição	MoSCoW	Pontuação
RFSO01	Como pesquisador, quero armazenar grandes volumes de imagens de câncer de pele para análise.	Must Have	8
RFSO03	Como administrador, quero garantir a privacidade dos dados dos pacientes armazenados.	Must Have	9
RFSO04	Como médico, quero acessar dados de forma segura para apoiar o diagnóstico.	Must Have	8
RFSO06	Como pesquisador, quero compartilhar dados de forma segura com outras organizações ou contribuidores.	Should Have	7
RFSO07	Como administrador, quero integrar dados de diferentes bases de dados para análise centralizada.	Must Have	9
RFSO08	Como pesquisador, quero consultar dados distribuídos sem me preocupar com a origem física dos dados.	Should Have	7
RFSO09	Como administrador, quero controlar acessos de forma centralizada para garantir segurança.	Must Have	8
RFSO10	Como pesquisador, quero realizar análises em tempo real para melhorar a precisão dos resultados.	Could Have	6
RFSO11	Como desenvolvedor, quero acessar um catálogo centralizado de metadados para identificar rapidamente os dados necessários.	Should Have	7
RNFSO07	Como administrador, quero garantir que os dados do sistema estejam em conformidade com LGPD.	Must Have	10

Tabela 3 – Tabela de Histórias de Usuário

4.2 Arquitetura

A arquitetura do projeto será baseada na abordagem de Data Fabric, que permite a integração e gestão de dados de forma unificada e eficiente. A seguir, a descrição dos principais componentes da arquitetura:

4.2.1 Diagrama de Arquitetura

A figura 3 ilustra a arquitetura proposta, cujo núcleo é o Data Fabric, responsável por integrar fontes de dados internas e externas, incluindo Google Cloud Storage, MongoDB e bancos de dados relacionais. O MinIO funciona como a fonte interna de armazenamento, dividida nas camadas Bronze, Silver e Gold, que suportam, respectivamente,

a inserção inicial, o pré-processamento e o enriquecimento dos dados. A ingestão se inicia por meio de uma DAG de inserção gerenciada pelo Apache Airflow, que realiza coleta (scraping) de imagens de câncer de pele em diferentes repositórios, armazenando-as na camada Bronze do Data Lake (MinIO) junto a seus metadados, os quais são registrados no Delta Lake. Em seguida, ocorre uma nova etapa coordenada por outra DAG do Apache Airflow, na qual o pré-processamento e a análise das imagens são executados, resultando em dados enriquecidos que passam a compor as camadas Silver ou Gold no MinIO, ao mesmo tempo em que o Delta Lake mantém a organização e o controle de versões dos metadados. O Spark atua como motor de processamento que aproveita a camada transacional do Delta Lake para garantir integridade (ACID) e versionamento, além de oferecer as ferramentas necessárias para manipular ou analisar grandes volumes de informações. Por fim, o Delta Sharing possibilita a disponibilização desses dados de maneira segura a pesquisadores e organizações, preservando a governança e a privacidade das informações.

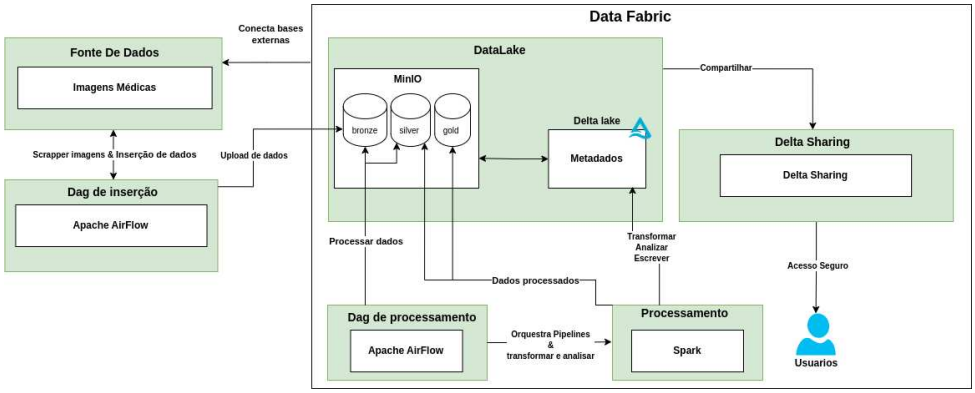

Figura 3 – Diagrama da Arquitetura do Projeto

4.2.2 Diagrama de Fluxo de Dados

A figura 4 ilustra um fluxo de dados em que múltiplas fontes (como hospitais, pesquisas colaborativas e repositórios de imagens médicas) alimentam o sistema por meio de uma DAG de inserção, orquestrada pelo Apache Airflow. Essa etapa controla a ingestão de arquivos e seus metadados, direcionando-os ao DataLake, no qual o MinIO provê armazenamento distribuído. Paralelamente, o Delta Lake é responsável por gerenciar as informações de metadados e garantir versionamento e rastreabilidade das atualizações. Na sequência, uma segunda fase de orquestração ocorre na etapa de Processamento de Dados, na qual o próprio Apache Airflow ativa as rotinas de pré-processamento e de análise, empregando o Spark especificamente para transformações nos metadados e extração de insights. Com os dados processados, o sistema conta com um módulo de Governança e Segurança que adota políticas de controle de acesso (RBAC) e está em conformidade com a LGPD, assegurando integridade e privacidade das informações médicas. Por fim, o módulo de Compartilhamento de Dados, apoiado no Delta Sharing, facilita o acesso seguro

a pesquisadores e outras organizações, permitindo consultas autorizadas e promovendo a colaboração em larga escala.

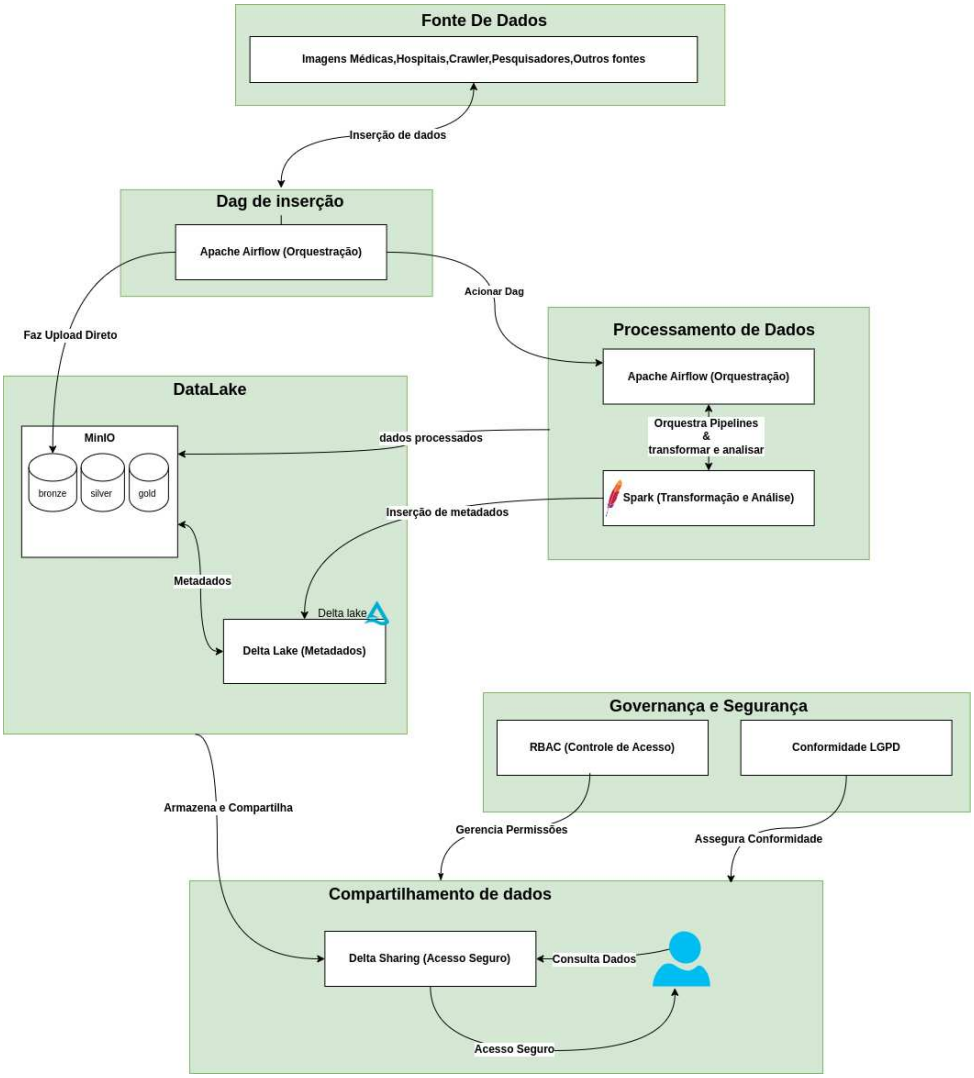

Figura 4 – Diagrama da Fluxo de Dados

4.2.3 Componentes da Arquitetura

- **Ingestão de Dados**: Utilização de pipelines de ingestão para coletar e armazenar imagens de câncer de pele de diversas fontes (hospitais, clínicas, pesquisas).

- **Armazenamento de Dados**: Implementação de um sistema de armazenamento distribuído, como MinIO, para garantir a escalabilidade e a disponibilidade das imagens.

- **Processamento de Dados**: Utilização de ferramentas como Apache Airflow para a orquestração de pipelines de processamento das imagens.

- **Gestão de Dados**: Aplicação de Data Fabric para fornecer uma camada de abstração que facilita a integração e gestão dos dados distribuídos.
- **Compartilhamento de Dados**: Utilização de Delta Sharing para permitir o compartilhamento seguro de dados entre diferentes organizações.
- **Interface de Usuário**: Desenvolvimento de uma interface web utilizando FastAPI para permitir o acesso e a manipulação dos dados por pesquisadores e profissionais de saúde.

4.2.4 Fluxo de Dados

- **Coleta e Ingestão**: As imagens são coletadas de diversas fontes e armazenadas no sistema de armazenamento distribuído.
- **Processamento**: As imagens armazenadas são processadas através de pipelines orquestrados pelo Apache Airflow, que podem incluir etapas de pré-processamento, análise e extração de características.
- **Armazenamento e Gestão**: Os dados processados são armazenados e geridos pela arquitetura de Data Fabric, que garante a integração e a consistência dos dados.
- **Compartilhamento de Dados**: Utilizando Delta Sharing, os dados podem ser compartilhados de forma segura com outras organizações e pesquisadores.
- **Acesso aos Dados**: A interface web desenvolvida com FastAPI permite que os usuários acessem e manipulem os dados de forma segura e eficiente.

4.3 Tecnologias Utilizadas

Com base no referencial teórico, as seguintes tecnologias serão utilizadas no projeto:

- **MinIO**: Para o armazenamento distribuído das imagens, garantindo escalabilidade e disponibilidade.
- **Apache Airflow**: Para a orquestração de pipelines de processamento de dados.
- **Data Fabric**: Para a integração e gestão de dados de forma unificada.
- **Delta Sharing**: Para o compartilhamento seguro de dados entre diferentes organizações.
- **FastAPI**: Para o desenvolvimento da interface web, permitindo o acesso e a manipulação dos dados.

- **Kubernetes**: Para a orquestração de containers, garantindo a escalabilidade e a alta disponibilidade do sistema.
- **Spark**: Fornece um alto nível de paralelismo, processando grandes quantidades de dados eficientimente em vários nós em um cluster.

4.4 Roadmap de Implementação

A implementação dessa arquitetura será conduzida em etapas sequenciais, cada qual associada a um intervalo de tempo estimado, de modo a garantir uma implantação organizada e confiável. Inicialmente, dedica-se um período de aproximadamente duas semanas à configuração do ambiente de desenvolvimento e à implantação do cluster Kubernetes, estabelecendo a infraestrutura necessária para a orquestração e o provisionamento de contêineres. Na sequência, com o Kubernetes já operacional, realiza-se a configuração do MinIO para armazenamento distribuído de dados brutos.; essas tarefas, que incluem testes iniciais de integração, tendem a demandar mais uma ou duas semanas.

Em seguida, procede-se à instalação e integração do Delta Lake com MiniO, fundamental para o versionamento e a gestão transacional dos dados, o que costuma levar cerca de uma semana, considerando a adequação às políticas de governança que regem os metadados. Logo após, configura-se o Apache Airflow, que assume o papel de orquestração das DAGs de inserção e de processamento. A etapa de implantação e ajuste das pipelines, contemplando definição de dependências, verificação de logs e testes de escalabilidade, durando de duas a três semanas.

Por fim, entra-se na fase de implementação do Delta Sharing, que inclui ajustes de segurança (por exemplo, RBAC para controle de acesso e conformidade com a LGPD) e finaliza o ciclo de entrega, em mais duas semanas. Nessa etapa, os recursos de compartilhamento seguro de dados são configurados e submetidos a validações de desempenho e de governança, assegurando que apenas usuários e organizações devidamente autorizados tenham acesso à informação. Após esse período, será realizadas verificações integradas em todo o ecossistema, incluindo testes de carga e monitoramento contínuo, assegurando uma solução robusta, escalável e pronta para suportar o processamento e a análise de grandes volumes de dados de maneira confiável.

Referências

ADDAGADA, T. Best practices for managing a data fabric. *DATAVERSITY*, 2022. Citado na página 46.

AKHTAR, R. Decentralized data mesh architectures: Benefits and challenges. *Journal of Data Science*, p. 123–145, 2021. Citado 2 vezes nas páginas 34 e 35.

ALHASSAN, I.; SAMMON, D.; DALY, M. Data governance activities: An analysis of the literature. *Journal of Decision Systems*, p. 64–75, 2016. Citado na página 21.

ARMBRUST, M.; XIN, R. *Spark SQL: Relational Data Processing in Spark*. [S.l.]: ACM SIGMOD, 2023. Citado na página 38.

ARUNDEL, J.; DOMINGUS, J. *Cloud Native DevOps with Kubernetes*. [S.l.]: O'Reilly Media, 2021. Citado na página 40.

BARIK, R. *Data Fabric Primer*. [S.l.], 2022. Disponível em: <https://www.globallogic.com/in/wp-content/uploads/sites/21/2023/12/Paper-Data-Fabric-primer.pdf>. Citado 5 vezes nas páginas 29, 30, 31, 32 e 45.

BENFELDT, O.; PERSSON, J. S.; MADSEN, S. Data governance as a collective action problem. *Information Systems Frontiers*, p. 299–313, 2020. Citado na página 23.

BLOHM, I. et al. Data products, data mesh, and data fabric: New paradigm(s) for data and analytics? *Business & Information Systems Engineering*, p. 643–652, 2024. Citado 2 vezes nas páginas 29 e 45.

BROWN, S.; JOHNSON, M. Securing fastapi applications: Best practices and patterns. *Security Engineering Review*, v. 8, p. 45–67, 2023. Citado na página 42.

BUILD a High-Performance Object Storage-as-a-Service Platform with MinIO. [S.l.], 2019. Acessado em: 5 jan. 2025. Disponível em: <https://min.io/resources/docs/CPG-MinIO-reference-architecture.pdf>. Citado 2 vezes nas páginas 47 e 48.

BURNS, B.; BEDA, J.; HIGHTOWER, K. *Kubernetes: Up and Running: Dive into the Future of Infrastructure*. [S.l.]: O'Reilly Media, 2019. Citado na página 39.

CHAMBERS, B.; ZAHARIA, M. Spark: The definitive guide. *O'Reilly Media*, 2023. Citado na página 38.

CHANG, J.; SHARMA, R. Apache airflow: Platform for programmatically authoring, scheduling and monitoring workflows. *Journal of Big Data Processing*, p. 45–67, 2021. Citado na página 36.

CHEN, D.; MARTINEZ, A. Challenges in implementing ai-powered dermatological diagnosis systems. *Journal of Healthcare Engineering*, v. 8, p. 145–167, 2023. Citado 2 vezes nas páginas 43 e 44.

CHEN, D.; WANG, L. Quality control in medical image processing pipelines. *Biomedical Engineering Review*, v. 10, p. 112–134, 2023. Citado na página 47.

DEHGHANI, Z. *Data Mesh: Delivering Data-Driven Value at Scale*. [S.l.: s.n.], 2020. Citado 2 vezes nas páginas 34 e 35.

DHAYNE, H. et al. In search of big medical data integration solutions - a comprehensive survey. *IEEE Access*, 2019. Citado 2 vezes nas páginas 50 e 51.

DOAN, A. et al. Information extraction challenges in managing unstructured data. *SIGMOD Record*, p. 14–20, 2008. Citado na página 24.

ESTEVA, A.; TOPOL, E. Deep learning-enabled medical computer vision. *Nature Digital Medicine*, v. 5, p. 1–9, 2022. Citado na página 43.

FANG, T. *Data Lakes: Foundations and Innovations*. [S.l.: s.n.], 2021. Citado na página 32.

FOUNDATION, A. S. *Best Practices for Apache Airflow Implementation*. [S.l.], 2023. Disponível em: <https://airflow.apache.org/docs/>. Citado na página 37.

GADE, K. R. Data analytics: Data fabric architecture and its benefits for data management. *MZ Computing Journal*, 2022. Citado 4 vezes nas páginas 28, 29, 32 e 45.

GARCIA, M.; LUMBRERAS, J. *Data Pipelines with Apache Airflow*. [S.l.]: Manning Publications, 2021. Citado na página 36.

GARCIA, M.; RODRIGUEZ, J. Medical image processing: From acquisition to analysis. *Journal of Medical Imaging*, v. 12, p. 145–167, 2023. Citado na página 46.

Gartner. *What is Data Fabric? Uses, Definition & Trends*. 2024. Accessed: 2024-12-19. Disponível em: <https://www.gartner.com/en/data-analytics/topics/data-fabric>. Citado na página 29.

GONZALEZ, M. Scalable systems for big data management. *Journal of Big Data*, 2019. Citado na página 33.

HAUG, A.; ZACHARIASSEN, F.; LIEMPD, D. van. The costs of poor data quality. *Journal of Industrial Engineering and Management*, p. 168–193, 2011. Citado na página 21.

HECHLER, E.; WEIHRAUCH, M.; WU, Y. C. *Data Fabric and Data Mesh Approaches with AI: A Guide to AI-based Data Cataloging, Governance, Integration, Orchestration, and Consumption*. [S.l.]: Apress, 2023. 38–42 p. Citado 2 vezes nas páginas 31 e 44.

HIGHTOWER, K.; BURNS, B.; BEDA, J. *Kubernetes: The Complete Guide*. [S.l.]: CNCF, 2019. Citado na página 39.

JAHNKE, L.; ASHER, A.; KERALIS, S. D. *The Problem of Data*. [S.l.], 2012. Disponível em: <https://www.clir.org/pubs/reports/pub154/>. Citado na página 24.

JOHNSON, P. Scaling data architectures with mesh principles. *Data Engineering Quarterly*, p. 34–50, 2021. Citado na página 35.

KARAU, H.; WARREN, R. High performance spark. *O'Reilly Media*, 2023. Citado na página 39.

KARAU, H.; WARREN, R. *Learning Spark: Lightning-Fast Data Analytics*. [S.l.]: O'Reilly Media, 2023. Citado na página 37.

KATAL, A.; WAZID, M.; GOUDAR, R. H. Big data: Issues, challenges, tools and good practices. In: *Proceedings of the Sixth International Conference on Contemporary Computing (IC3)*. [S.l.: s.n.], 2013. p. 404–409. Citado 2 vezes nas páginas 21 e 22.

KUMAR, A.; PATEL, S. *Medical Image Processing Pipelines: Design and Implementation*. [S.l.]: Springer, 2023. Citado na página 43.

KUMAR, R.; SINGH, A. Data ingestion strategies for medical imaging systems. *Healthcare Informatics Journal*, v. 8, p. 234–256, 2023. Citado na página 46.

KUMAR, R.; SINGH, A. Modern api development with fastapi and python. *Journal of Software Engineering*, v. 15, p. 123–145, 2023. Citado 2 vezes nas páginas 41 e 42.

KUMAR, V.; SINGH, R. Apache airflow: A comprehensive study of workflow management. *International Journal of Data Engineering*, v. 13, p. 78–92, 2022. Citado na página 36.

LEE, D. et al. *Delta Lake: The Definitive Guide: Modern Data Lakehouse Architectures with Data Lakes*. [S.l.]: O'Reilly Media, 2024. Citado 2 vezes nas páginas 49 e 50.

L'HEUREUX, A. et al. Machine learning with big data: Challenges and approaches. *IEEE Access*, p. 7776–7797, 2017. Citado na página 22.

LUKSA, M. *Kubernetes in Action*. [S.l.]: Manning Publications, 2021. Citado na página 40.

MARTINEZ, C.; THOMPSON, E. Best practices for large-scale fastapi applications. *Web Engineering Quarterly*, v. 4, p. 78–95, 2023. Citado na página 42.

MARTINEZ, L. Federated governance in data mesh implementations. *International Journal of Data Management*, v. 15, p. 56–78, 2022. Citado na página 35.

MOUSA, A. H.; SHIRATUDDIN, N. Data warehouse and data virtualization comparative study. In: *2015 International Conference on Developments of E-Systems Engineering (DeSE)*. [S.l.: s.n.], 2015. p. 369–372. Citado na página 28.

NAMBIAR, A.; MUNDRA, D. An overview of data warehouse and data lake in modern enterprise data management. *Big Data and Cognitive Computing*, p. 132, 2022. Citado na página 25.

NUTHALAPATI, A. *Architecting data lake-houses in the cloud: Best practices and future direction*. [S.l.: s.n.], 2018. 1903 p. Citado na página 33.

PETERSON, E.; LEE, J. *FastAPI Performance Analysis in Production Environments*. [S.l.], 2023. Citado na página 42.

RAMíREZ, S. Fastapi: A modern framework for building apis with python. *Python Software Foundation*, 2023. Citado na página 41.

RAMíREZ, S.; GONZáLEZ, M. *Building Data-Intensive Applications with FastAPI*. [S.l.]: O'Reilly Media, 2022. Citado na página 41.

REDMAN, T. C. The impact of poor data quality on thetypical enterprise. *ACM*, p. 79–82, 1998. Citado 4 vezes nas páginas 21, 22, 23 e 24.

SANTOSO, L. W.; YULIA. Data warehouse with big data technology for higher education. In: *Proceedings of the 4th Information Systems International Conference (ISICO 2017)*. [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 25 e 27.

SHARMA, P.; GUPTA, R. Scalable data pipeline orchestration with apache airflow. In: *Proceedings of the International Conference on Big Data Engineering*. [S.l.: s.n.], 2021. p. 234–245. Citado na página 36.

SHARMA, V. et al. (Ed.). *Data Fabric Architectures: Web-Driven Applications*. [S.l.]: De Gruyter, 2023. Citado 3 vezes nas páginas 30, 31 e 44.

SMITH, J.; BROWN, R. Quality assurance in medical image processing. *Healthcare Data Management*, v. 15, p. 78–95, 2023. Citado 2 vezes nas páginas 43 e 44.

SMITH, J.; BROWN, S. *Image Preprocessing Techniques in Medical Analysis*. [S.l.]: Academic Press, 2023. Citado na página 46.

SMITH, K. Open source tools for data mesh implementations. *Open Data Review*, p. 90–105, 2023. Citado na página 35.

STONE, M. et al. From information mismanagement to misinformation - the dark side of information management. *The Bottom Line*, 2019. Citado na página 23.

VAISMAN, A.; ZIMáNYI, E. *Data Warehouse Systems: Design and Implementation*. [S.l.]: Springer, 2014. 72–80 p. (Data-Centric Systems and Applications). Citado 3 vezes nas páginas 25, 26 e 28.

VASS, J.; STEWART, M. *Big Data Management in the Cloud Era*. [S.l.: s.n.], 2020. Citado na página 33.

VERMA, A.; PEDROSA, L.; KORUPOLU, M. Large-scale cluster management at google with borg. *Proceedings of the European Conference on Computer Systems*, p. 18–33, 2020. Citado na página 39.

VOLK, M. et al. Challenging big data engineering: Positioning of current and future development. In: *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS)*. [S.l.: s.n.], 2019. p. 351–358. Citado 4 vezes nas páginas 21, 22, 23 e 24.

WANG, M.; CHEN, L. Automated skin lesion analysis using deep learning pipelines. *Journal of Medical Imaging*, v. 10, p. 234–256, 2023. Citado 2 vezes nas páginas 43 e 47.

WENDELL, P.; ZAHARIA, M. *The Definitive Guide to Apache Spark*. [S.l.]: Databricks, 2023. Citado na página 38.

WILSON, J. *Data Integration Patterns with FastAPI*. [S.l.]: Apress, 2023. Citado na página 41.

XU, L. et al. What clinics are expecting from data scientists? a review on data oriented studies through qualitative and quantitative approaches. *IEEE Access*, 2019. Citado na página 51.

ZAGAN, E.; DANUBIANU, M. Data lake approaches: A survey. p. 189–191, 2020. Citado na página 33.

ZAHARIA, M.; XIN, R.; WENDELL, P. *Apache Spark: A Unified Engine for Big Data Processing*. [S.l.]: Communications of the ACM, 2016. Citado na página 37.

ZHANG, J.; WANG, L.; SINGH, R. Deep learning applications in dermatological image analysis. *Medical Image Analysis*, v. 28, p. 187–203, 2021. Citado na página 42.

ZHANG, W.; LI, X. Data augmentation techniques for medical image analysis. *Pattern Recognition in Medicine*, v. 15, p. 78–95, 2023. Citado na página 47.