Instituto Federal da Paraiba - IFPB

Bacharelado em Engenharia de Computação

Ialy Cordeiro de Sousa

Arquitetura conceitual para automação de ETL de microdados públicos: um estudo de caso com o ENEM

Campina Grande – PB

2025

Ialy Cordeiro de Sousa

Arquitetura conceitual para automação de ETL de microdados públicos: um estudo de caso com o ENEM

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Instituto Federal da Paraiba - IFPB como requisito parcial para obtenção do grau de Bacharela em Engenharia de Computação.

Orientador: Dr. Paulo Ribeiro Lins Júnior

Campina Grande – PB

2025

Catalogação na fonte:
Ficha catalográfica elaborada por Gustavo César Nogueira da Costa - CRB 15/479

S725a Sousa, Ialy Cordeiro de.

Arquitetura conceitual para automação de ETL de microdados públicos: um estudo de caso com o ENEM / Ialy Cordeiro de Sousa. - Campina Grande, 2025.
25 f.: il.

Trabalho de Conclusão de Curso (Graduação em Tecnologia em Construção de Edifícios) - Instituto Federal da Paraíba, 2025.
Orientador: Prof. Dr. Paulo Ribeiro Lins Júnior

1. Engenharia de dados. 2. ETL (Extração, Transformação e Carga). 3. Ciência aberta. 4. Automação de processos. 5. Microdados públicos. I. Lins Júnior, Paulo Ribeiro. II. Título.
.

CDU 004.6

Sumário

Resumo

Este trabalho tem como objetivo propor uma arquitetura conceitual denominada AutoMicroETL, voltada para a automação do processo de Extração, Transformação e Carga (ETL) de microdados públicos. A solução busca reduzir barreiras técnicas de acesso e promover a democratização da análise de dados governamentais, oferecendo uma estrutura modular, reutilizável e de fácil implantação em ambientes locais. A metodologia consistiu na implementação de um pipeline lógico baseado em ferramentas *open source*, utilizando *Docker* para conteinerização do ambiente, *Python* para extração e padronização dos dados, *MinIO* como repositório de armazenamento em camadas (bronze, prata e ouro) e *Apache Airflow* para orquestração das etapas de execução. O estudo de caso foi conduzido com os microdados do ENEM, disponibilizados pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP). Como resultado, foi desenvolvido um pipeline funcional capaz de automatizar a coleta, descompactação e padronização dos microdados, garantindo reprodutibilidade e portabilidade em qualquer ambiente compatível com Docker. A arquitetura mostrou-se escalável, podendo ser expandida para outras bases públicas além do ENEM. Conclui-se que a AutoMicroETL contribui para a democratização da análise de dados públicos, ao oferecer uma solução acessível e eficiente para pesquisadores, estudantes e gestores, representando um passo relevante no fortalecimento da ciência aberta e na promoção de boas práticas de engenharia de dados no setor público e acadêmico.

Palavras-chave: ETL, Microdados Públicos, Automação, Arquitetura de Dados, Docker, Airflow.

1 Introdução

A grande importância da ciência embasada em dados e da transparência de dados públicos, tem fomentado o acesso à dados governamentais em grande escala. Os microdados brasileiros, educacionais, demográficos, de saúde, trabalhistas, tem sido disponibilizados por instituições como o IBGE, o INEP, os Ministérios da Educação, do Trabalho, da Saúde, entre outros. Esses dados desagregados e anonimizados representam uma grande e importante fonte informação para pesquisadores, gestores públicos jornalistase para a população, principalmente quando utilizados para análises sobre políticas públicas, desigualdades sociais, desempenho educacional e mercado de trabalho. De acordo com Neves, os dados públicos disponíveis, abrem possibilidades para a sociedade se auto conhecer e realizar as mais diversas análises das informações públicas por meio da correlação de diferentes bases de dados, até a criação de aplicações que fazem uma leitura frequente de bases de dados públicas para fornecer soluções que a beneficie ou que geram oportunidades de negócio (NEVES, 2013). Dessa maneira, os microdados públicos permitem que informações públicas sejam utilizadas livremente na tomada de decisão e na geração de conhecimento, garantindo sua utilidade e potencial para reaproveitamento.

Apesar de serem disponibilizados gratuitamente, seu uso não é prático ou funcional, pois esbarra em uma série de limitações técnicas, pois são fornecidos de forma bruta, sem nenhum tratamento, geralmente compactados em arquivos *ZIP*, que contém múltiplos arquivos *CSV*, ou outros formatos, e com estrutura que variam bastante entre os anos e os dados. Ademais, as páginas, portais que contém os dados habitualmente exige que o usuário faça a identificação, extrações e downloads individuais e manuais dos arquivos, necessitando de uma padronização, compreenssão da documentação técnica e tratamento, antes de realizar qualquer análise. Em função dessas dificuldades, diversas soluções têmsido criadas de forma pontual para bases específicas. É o caso da biblioteca *pycaged*, desenvolvida exclusivamente para o tratamento dos dados do CAGED (CAIXETA, 2020). Embora bastante útil em seus respectivos contextos, essas bibliotecas e ferramentas são limitadas por não oferecerem uma estrutura genérica, modular e reutilizável com processo de ETL (Extraction, Transformation, and Load) automatizado, pois não possibilita o mesmo tratamento para outras bases de microdados, como por exemplo, o ENEM, a PNAD, a RAIS, INEP.

Diante disso, este trabalho se propõe a apresentar uma arquitetura de ETL automatizada, que se utilizada de ferramentas de fácil acesso e *open source*, que são largamente utilizadas por especialistas em dados, como o Docker para conteinerização e portabilidade, o Python para extração e tratamento dos dados, além de sua ampla comunidade e bibliotecas, o MinIO para simular um data lake com suas estrutura medalhão (bronze, prata e ouro), e o Apache Airflow que fará toda a orquestração e automação das extrações e tratamentos de acordo com o agendamento definido.

A proposta será validada por meio de um estudo de caso utilizando os microdados do ENEM disponibilizados pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) que realiza um papel crucial na organização e viabilização de dados sobre educação no Brasil. Ele foi fundado em 1937 como Instituto Nacional de Pedagogia e transformado em autarquia federal vinculada ao Ministério da Educação (MEC) em 1997, o INEP é responsável por fornecer informações precisas e confiáveis sobre o sistema educacional do país (FONSECA; NAMEN, 2016). No entanto, o objetivo central é fornecer uma solução genérica e escalável, adaptável a qualquer conjunto de microdados estruturados disponibilizados por órgãos públicos. Com isso, busca-se contribuir para a democratização da análise de dados públicos, reduzindo a barreira técnica de entrada e promovendo boas práticas de engenharia de dados no setor público e acadêmico.

2 Problema

O uso de microdados públicos disponibilizados por órgãos governamentais, como o INEP, o IBGE, e o Ministério do Trabalho, entre outros, exige um grande processo manual por parte da equipe de engenharia e análise de dados. Normalmente, os arquivos estão disponíveis em páginas sem padronização, onde se faz necessária uma busca detalhada, realização de downloads individuais, descompactação desses arquivos, interpretação da documentação técnica e uma posterior padronização e entendimento, para poder utilizá-los. Esse fluxo não estruturado compromete a reprodutibilidade, aumenta o risco de erro humano e dificulta a integração com pipelines automatizados de análise e visualização de dados.

Embora vários estudos e bibliotecas Python venham sendo desenvolvidos ultimamente, para extrações e análises de microdados, existem apenas soluções pontuais e exclusivas

para algumas bases de dados, como a RAIS ou o CAGED, contudo, observa-se uma lacuna importante para se ter uma solução mais genérica e modular que possa ser aplicada e reutilizável para um amplo conjunto de microdados públicos brasileiros.

A ausência de uma arquitetura flexível, automatizada e compatível com múltiplas fontes de dados dificulta a padronização, a escalabilidade e a reprodutibilidade de análises em projetos com dados públicos. Essa limitação compromete não apenas o uso eficiente dos microdados como também o desenvolvimento de políticas públicas baseadas em dados.

Este trabalho, portanto, busca preencher essa lacuna ao propor uma solução automatizada e generalizável, que não se limita a um conjunto específico de dados, mas que pode ser adaptada para diversos formatos e fontes públicas, promovendo maior acessibilidade, modularidade e robustez nos processos de ingestão e organização de dados públicos.

3 Justificativa

Os microdados públicos representam uma fonte riquíssima de informações sobre os diversos assuntos, como a demografia do país, informações educacionais, entre outras, utilizadas amplamente por pesquisadores, instituições públicas, veículos de comunicação e cidadãos interessados em entender o crescimento populacional, os avanços, perdas, desempenho e perfil dos estudantes brasileiros, entre as mais diversas informações. Contudo, o formato em que esses dados são oferecidos atualmente, como arquivos compactados, com múltiplos arquivos CSV, sem padronização entre os anos e documentação limitada, representa um obstáculo significativo à sua utilização prática.

A ausência de um pipeline automatizado e estruturado para ingestão e tratamento dos microdados torna o processo de análise altamente dependente de conhecimento técnico prévio e esforços repetitivos. Isso compromete a reprodutibilidade das análises, aumenta o tempo necessário para extração de informações úteis e limita o acesso aos dados a um grupo restrito de especialistas.

Dessa forma, este trabalho justifica-se pela necessidade de criar uma solução moderna, acessível e automatizada que trate os microdados públicos, de forma eficiente e modular. Ao empregar ferramentas como Docker, Apache Airflow e MinIO, a arquitetura proposta permite:

- Automatizar tarefas repetitivas de verificação, download e organização dos dados;

- Armazenar os dados em camadas (bronze, prata e ouro), seguindo boas práticas de engenharia de dados;
- Reduzir barreiras de acesso e promover a democratização da informação;
- Facilitar a análise exploratória por meio de notebooks interativos na camada ouro.

Ao propor uma estrutura reutilizável, baseada em padrões modernos e tecnologias livres, esta solução se posiciona como uma contribuição prática não apenas para o acesso aos dados do ENEM, que é a base piloto, mas como um modelo replicável para o tratamento de outros microdados públicos. Isso fortalece o compromisso com a ciência aberta, a transparência institucional e a justiça informacional.

4 Objetivos

4.1 Objetivo Geral

Automatizar o processo de extração, transformação e carregamento (ETL) de microdados públicos disponibilizados por instituições governamentais, utilizando tecnologias open source como Docker, Python e MinIO, com foco em padronização, reprodutibilidade e integração com ambientes de análise de dados.

4.2 Objetivos Específicos

- Investigar os desafios associados à coleta e organização de microdados públicos de instituições como IBGE, INEP, e Ministério do Trabalho;
- Projetar e implementar uma arquitetura modular baseada em boas práticas de engenharia de dados para automação do processo de ETL;
- Criar um ambiente containerizado com Docker que integre os serviços necessários à execução do pipeline;
- Automatizar a extração e o armazenamento dos microdados do ENEM em sua forma original (camada bronze), utilizando MinIO;
- Realizar a descompactação, padronização de colunas e estruturação dos dados na camada prata;

- Orquestrar todo o fluxo com Apache Airflow, assegurando agendamento e reexecução automatizada;
- Garantir a reprodutibilidade e escalabilidade do processo de ingestão de dados;
- Validar a arquitetura proposta utilizando uma base real (ENEM) como estudo de caso.

5 Fundamentação Teórica

Este capítulo apresenta os conceitos e fundamentos que embasam a proposta deste trabalho, com foco na utilização de microdados públicos, práticas modernas de engenharia de dados, orquestração de pipeline e uso de ferramentas tecnológicas como Docker, Apache Airflow, MinIO e Jupyter Notebook.

5.1 Microdados públicos e ciência aberta

Microdados são conjuntos de dados desagregados, anonimizados e estruturados que permitem análises em nível individual ou institucional, sendo fundamentais para estudos estatísticos, políticas públicas e pesquisa acadêmica (ALVES; MARTINS; SILVA, 2020). No contexto educacional, microdados como os do Exame Nacional do Ensino Médio (ENEM) oferecem subsídios valiosos para avaliar o desempenho de estudantes, desigualdades regionais, e a eficácia de políticas educacionais.

Entretanto, a forma como esses dados são disponibilizados, geralmente por meio de arquivos compactados, sem padronização entre os anos e com documentação dispersa, representa um obstáculo à sua utilização prática (MUNIZ et al., 2018). Isso compromete tanto a acessibilidade quanto a reprodutibilidade das análises.

Segundo Murieta (MURRIETA, 2024) a ciência aberta é um movimento que propõe alterações estruturais no modo como o conhecimento científico é realizado, sistematizado, compartilhado e reutilizado. Caracteriza uma nova maneira de fazer ciência, mais cooperativo, transparente e sustentável. Instiga grande empenho de pesquisadores, governos, agências de fomento ou da própria comunidade científica para tornar os resultados primários de pesquisas acessíveis ao público em formato digital, sem nenhuma ou com um mínimo de restrição, como meio para acelerar a pesquisa e o conhecimento, vislumbrando

aumentar a transparência e a colaboração e promover a inovação.

A ciência aberta e os dados governamentais abertos vêm sendo incentivados como parte de políticas de transparência e democratização do conhecimento, conforme orientações da LAI (Lei de Acesso à Informação) e iniciativas de dados abertos no Brasil. No entanto, a ausência de mecanismos que transformem esses dados brutos em informações acessíveis ainda limita seu potencial de uso (MACHADO et al., 2019).

5.2 Engenharia de dados e arquitetura medalhão

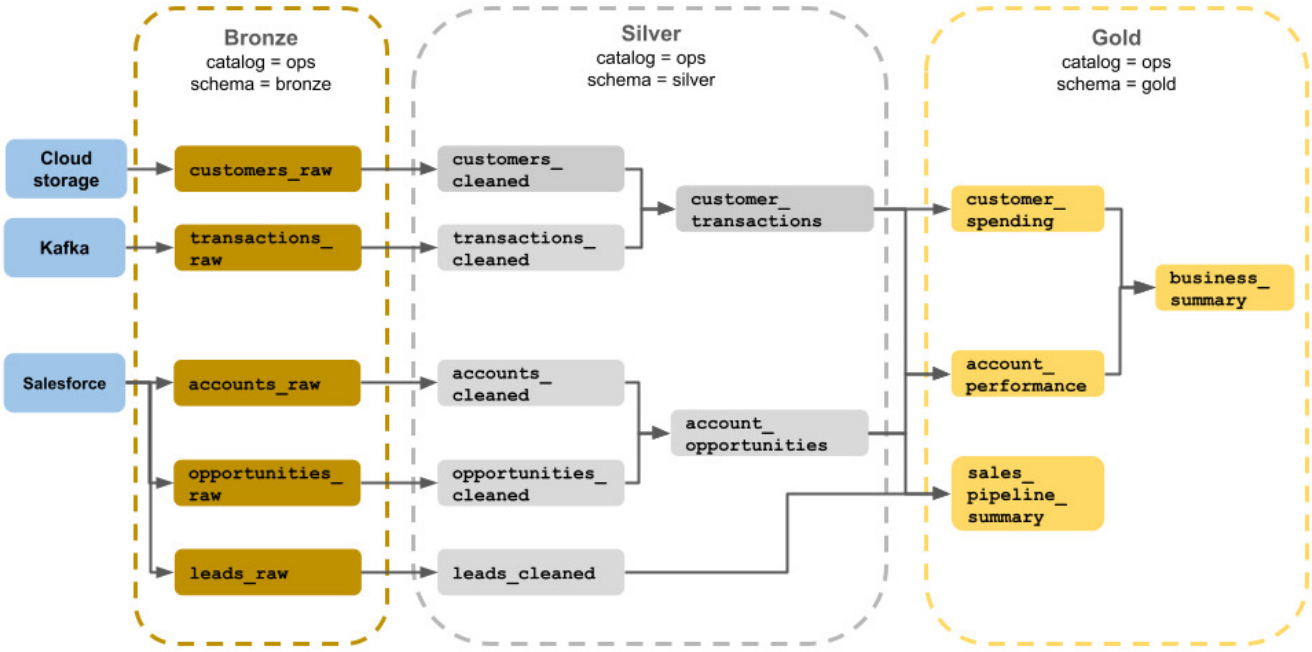
Engenharia de dados é responsável por projetar, construir e manter sistemas de coleta, armazenamento, processamento e entrega de dados. Uma das abordagens mais consolidadas na área é o uso de pipelines de dados com arquitetura em camadas, geralmente chamadas de **bronze**, **prata** e **ouro** (Figura 1), que organizam o fluxo dos dados conforme seu nível de tratamento e preparação para análise (INMON, 2016).

A arquitetura medalhão é um padrão moderno de organização de dados em data lakes, que propõe a divisão em três camadas principais: **bronze**, **prata** e **ouro**. Seu objetivo é organizar os dados em diferentes estágios de qualidade e transformação, favorecendo rastreabilidade, governança e reprocessamento. Na camada **bronze**, os dados são armazenados de forma bruta, sem alterações, preservando as informações como vieram da fonte, o que permite reprocessamento total das camadas seguintes. Já na camada **prata**, os dados passam por processos de limpeza, normalização e transformação, com definição de tipos e exclusão de registros inválidos. Finalmente, na camada **ouro**, os dados são ajustados para atender a casos de uso específicos, com agregações, modelagem em estrela (star schema), enriquecimento com tabelas de dimensão e exclusão de informações sensíveis para garantir a privacidade dos usuários finais. Essa arquitetura geralmente é implementada em soluções baseadas em cloud computing, como o Delta Lake, por facilitar a escalabilidade e o controle incremental da qualidade dos dados (WISELKA, 2024).

- **Camada bronze** armazena os dados brutos exatamente como foram extraídos da fonte, garantindo rastreabilidade.
- **Camada prata** realiza transformações básicas, como padronização de colunas, tipos e normalizações.

- **Camada ouro** contém os dados enriquecidos e prontos para análises exploratórias ou modelagens.

Figura 1: Exemplo de arquitetura medalhão



Fonte: (Microsoft, 2025).

Esse modelo modular permite maior controle de qualidade, auditabilidade e reprocessamento de dados em diferentes fases, sendo especialmente útil em projetos com fontes públicas heterogêneas.

5.3 Automação e orquestração de pipelines

A crescente complexidade dos fluxos de trabalho em projetos de ciência de dados e aprendizado de máquina tem impulsionado a necessidade de automação eficiente dos pipelines de dados. Os métodos tradicionais e manuais de ingestão, transformação e carregamento mostram-se inadequados frente ao volume, variedade e velocidade dos dados em ambientes de larga escala, pois são suscetíveis a erros, retrabalho e gargalos operacionais. A automação desses pipelines permite a integração fluida entre múltiplas fontes de dados, otimizando o desempenho e reduzindo significativamente o esforço operacional.

Com o avanço das soluções em nuvem e das estruturas de processamento distribuído, como Apache Spark e Airflow, os pipelines automatizados tornaram-se pilares na sustentação de aplicações baseadas em inteligência artificial, viabilizando a entrega de dados de alta qualidade para treinamento e inferência. Entretanto, a implementação desses fluxos automatizados ainda enfrenta desafios importantes, como a integração de fontes heterogêneas, a rastreabilidade e a governança dos dados, bem como a conformidade com regulamentações. Esses obstáculos demandam conhecimento aprofundado sobre boas práticas, ferramentas modernas e tendências emergentes, a fim de garantir a escalabilidade, a segurança e a confiabilidade de todo o ciclo de vida dos dados (SINHA, 2024).

5.4 Uso de contêineres e isolamento de ambientes

A tecnologia de conteinerização, com destaque para o Docker, tem revolucionado o desenvolvimento e a implantação de aplicações em ambientes computacionais modernos. O Docker fornece uma plataforma padronizada para empacotamento, distribuição e execução de aplicativos em contêineres, encapsulando o código, as bibliotecas, o tempo de execução e todas as dependências necessárias para garantir a consistência entre diferentes ambientes (MERKEL, 2014). Essa abordagem se baseia em tecnologias de contêineres do Linux, como o LXC, que proporcionam virtualização leve por meio de mecanismos de isolamento em nível de kernel (SOLTESZ et al., 2007).

Desde sua introdução em 2013, o Docker tem sido amplamente adotado em setores como computação em nuvem, DevOps e arquiteturas de microsserviços, sendo reconhecido por otimizar fluxos de trabalho, melhorar a utilização de recursos e reduzir o tempo de entrega de aplicações (BOETTIGER, 2015). Sua popularidade se deve, em grande parte, à comunidade ativa que contribui com ferramentas, integrações e boas práticas, ampliando seu ecossistema e sua versatilidade (VAUGHAN-NICHOLS, 2016).

No contexto acadêmico, o Docker tem se destacado como ferramenta fundamental para experimentos computacionais reprodutíveis, promovendo a transparência e a replicabilidade em pesquisas científicas (BOETTIGER, 2015). Além disso, sua aplicação tem sido explorada no desenvolvimento de sistemas distribuídos escaláveis, facilitando a execução de cargas de trabalho complexas (CHUNG et al., 2019).

A utilização de contêineres proporciona benefícios cruciais para ambientes de dados modernos, como portabilidade, isolamento, escalabilidade e controle de versão. A porta-

bilidade permite que aplicações sejam executadas de forma consistente entre diferentes estágios (desenvolvimento, teste e produção), reduzindo falhas de execução relacionadas a ambientes divergentes (SOLTESZ et al., 2007). O isolamento garante que os serviços operem de forma independente, promovendo segurança e eficiência no uso dos recursos (MERKEL, 2014). A escalabilidade é potencializada pelo suporte a orquestradores como o Kubernetes, que automatizam o gerenciamento e o balanceamento de carga entre contêineres (BURNS et al., 2016). Já o versionamento é simplificado pela estrutura de imagens imutáveis do Docker, permitindo o rastreio e a reversão de versões com segurança (HILL, 2016).

Dessa forma, a conteinerização tem se consolidado como uma prática essencial na construção de plataformas modernas de dados, oferecendo um ambiente flexível, controlado e eficiente para a execução de pipelines de dados, análises e aplicações de machine learning.

5.5 Data Lake como alternativa às arquiteturas tradicionais

Com o avanço da era do big data, cresce o entendimento de que novas arquiteturas de dados são necessárias para lidar com a crescente complexidade, diversidade e volume das informações. Essa nova era, caracterizada por ser computacionalmente intensiva e orientada a dados em larga escala, exige abordagens mais flexíveis e escaláveis para armazenamento e processamento. Segundo Woods (WOODS, 2011), os modelos tradicionais de dados são insuficientes diante do cenário atual, sendo necessário adotar arquiteturas mais robustas e adaptáveis às exigências do big data.

Nesse contexto, o conceito de *data lake* surge como uma alternativa moderna aos tradicionais *data warehouses*, oferecendo um modelo de repositório capaz de armazenar dados em diferentes formatos, sejam eles não estruturados (como imagens e áudios), semiestruturados (como arquivos JSON e CSV), estruturados (dados com esquema definido) ou multiestruturados (combinação de formatos, como páginas da web) (PALMER, 2022).

De acordo com (PALMER, 2022), um *data lake* tem como principal característica, armazenar dados no seu formato original, sem a necessidade de transformações preliminares. Ao contrário do modelo de *data warehouse*, onde o processo de ETL (Extract, Transform, Load) é executado antes da carga dos dados, no *data lake* os dados são carregados primeiro (em sua forma bruta) e transformados posteriormente conforme a necessidade de cada usuário ou setor da organização. Essa abordagem, muitas vezes chamada de ELT

(Extract, Load, Transform), permite maior agilidade e reutilização dos dados, além de reduzir os custos iniciais de ingestão.

Além disso, uma vez armazenados no data lake, os dados tornam-se disponíveis para diversas finalidades analíticas em toda a organização, promovendo compartilhamento, descoberta e reutilização. Isso o torna uma arquitetura atraente para empresas que trabalham com múltiplos tipos de dados e desejam flexibilidade na modelagem e no consumo das informações ao longo do tempo (PALMER, 2022).

Exitem diversas alternativas de código aberto para se configurar um *data lake*, uma delas é o *MinIO* (MINIO,), com sua escalabilidade, resiliente a falhas de servidor, armazena grandes volumes de dados, de fácil configuração e não é necessária muita administração, tornando-o basteante adequado para a configuração de um data lake pessoal (PALMER, 2022).

6 Metodologia

A metodologia adotada baseia-se na construção de um pipeline de dados moderno, distribuído em camadas e automatizado via orquestração de workflows. A arquitetura foi estruturada com o uso de contêineres Docker para isolar os serviços, armazenamento via MinIO para armazenar os arquivos em suas diferentes camadas (bronze, prata e ouro), com organização por ano e tipo de dado, promovendo um padrão de governança dos arquivos e, controle dos fluxos com Apache Airflow.

6.1 Etapas de Desenvolvimento

As etapas descritas a seguir foram implementadas e orquestradas utilizando ferramentas específicas. Todas as ações são executadas automaticamente a partir de uma DAG configurada no Apache Airflow, que organiza as dependências e agendas de execução:

1. **Preparação do ambiente**: Foi criado um ambiente Docker com serviços do MinIO, Apache Airflow e Jupyter Notebook. Isso garantiu isolamento e reprodutibilidade local.

2. **Extração de dados (camada bronze)**: Foi desenvolvido um script em Python responsável por fazer o download dos arquivos ZIP contendo os microdados direta-

mente do portal do INEP, armazenando-os no MinIO.

3. **Descompactação e padronização (camada prata)**: Utilizando Python (com bibliotecas como zipfile e pandas), os arquivos foram descompactados e padronizados (renomeando colunas, padronizando tipos, etc.). O resultado é armazenado no MinIO novamente, agora na camada prata.

4. **Tratamento e estruturação (camada ouro)**: Os arquivos padronizados foram organizados para consumo final. A estrutura resultante foi validada via notebooks Jupyter. Essa camada foi utilizada apenas para verificação da qualidade e integridade dos dados.

5. **Orquestração com Airflow**: A DAG foi configurada para automatizar a verificação de novas versões, executar os scripts em ordem e gerar logs de execução. Todas as etapas estão integradas no Airflow e utilizam volumes Docker compartilhados.

6.2 Arquitetura Proposta

A arquitetura desenvolvida neste trabalho, denominada **AutoMicroETL**, foi concebida para ser reutilizável, modular e de fácil implantação em ambientes locais, mesmo com recursos computacionais limitados. Seu principal objetivo é facilitar o processo de ingestão e organização de microdados públicos de forma automatizada, acessível e reprodutível, permitindo que qualquer pessoa, com o mínimo de conhecimento técnico, possa executar o processo em sua própria máquina.

A solução proposta adota uma abordagem baseada em contêineres *Docker*, garantindo o isolamento dos serviços, a padronização do ambiente e a facilidade de implantação. Todo o ecossistema necessário à execução do pipeline é encapsulado e gerenciado em containers, tornando o sistema portável e independente da infraestrutura do usuário.

A estrutura segue o modelo de *data lake* organizado em três camadas: **bronze**, **prata** e **ouro**. A **camada bronze** armazena os dados brutos conforme foram obtidos da fonte oficial; a **camada prata** contém os dados descompactados e padronizados; e a **camada ouro** guarda os dados tratados e prontos para consumo analítico, ficando disponível para futuras análises por diferentes ferramentas, conforme a necessidade do usuário.

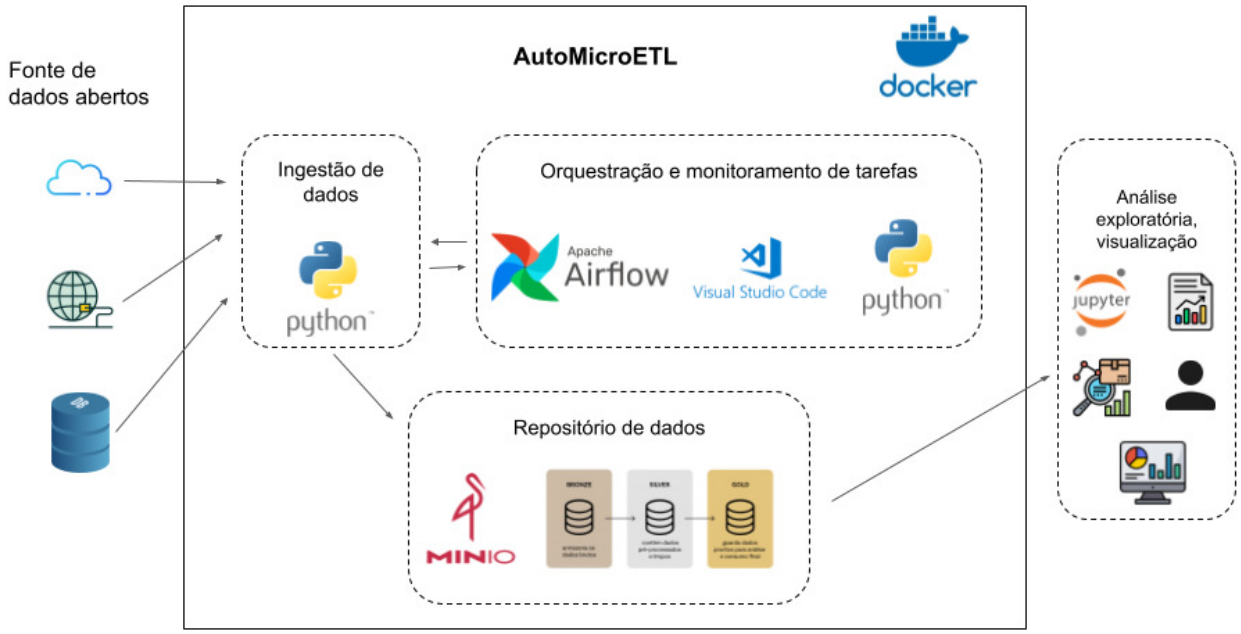
Além disso, a arquitetura conta com a orquestração automatizada por meio do *Apache Airflow*, que coordena as etapas de verificação de novos arquivos, extração, descompacta-

ção, padronização e movimentação entre camadas. A modularidade da solução permite que novos fluxos sejam incorporados com facilidade, viabilizando sua aplicação a diferentes bases de dados públicos além do ENEM.

Foram utilizadas as seguintes ferramentas:

- **Docker**: utilizado para criar um ambiente containerizado, isolado e portátil, com todos os serviços necessários ao pipeline.
- **Python**: linguagem utilizada nos scripts de automação, incluindo a extração dos arquivos, descompactação e padronização dos dados.
- **MinIO**: utilizado como repositório de armazenamento de objetos, simulando um *data lake* local, estruturado em camadas.
- **Apache Airflow**: responsável pela orquestração do fluxo de dados, executando e monitorando automaticamente todas as etapas do pipeline.

Figura 2: Arquitetura AutoMicroETL



Fonte: Elaboração da autora (2025).

A Figura 2 apresenta a arquitetura da solução **AutoMicroETL**, desenvolvida com o objetivo de automatizar a extração, organização e disponibilização de microdados públi-

cos, como os do ENEM. A estrutura é composta por módulos integrados, todos executados em um ambiente containerizado com *Docker*.

À esquerda do diagrama, encontram-se as fontes de dados públicos, como os portais do governo e instituições como o INEP. Esses dados são obtidos por scripts desenvolvidos em Python, responsáveis por realizar a ingestão automática dos arquivos compactados e armazená-los na **camada bronze** do repositório MinIO, que atua como um *data lake* local.

Na sequência, o pipeline realiza a descompactação e padronização dos dados, que são organizados na **camada prata**, também no MinIO. Nessa etapa, ocorre o tratamento inicial dos arquivos CSV, como padronização de colunas e estruturação dos dados.

Posteriormente, os dados são refinados e disponibilizados na **camada ouro**, prontos para análise exploratória, que podem ser feitas por meio de notebooks interativos no *Jupyter Notebook* ou outra ferramenta, onde os dados já tratados podem ser visualizados, filtrados e explorados de forma dinâmica.

O controle de fluxo entre essas etapas é feito pelo *Apache Airflow*, que orquestra automaticamente as tarefas, desde a verificação de novos dados até a organização final nas camadas. Todo esse ambiente é executado de forma isolada e reprodutível com *Docker*, garantindo portabilidade, modularidade e facilidade de implantação.

A arquitetura proposta é modular e reutilizável, podendo ser adaptada para o tratamento automatizado de diferentes bases de microdados públicos, promovendo governança, eficiência e reprodutibilidade no processo de análise de dados.

7 Resultados

Este capítulo apresenta os resultados obtidos com a implementação da arquitetura **AutoMicroETL**, validada com a extração automatizada dos microdados do ENEM 2023 como estudo de caso.

A arquitetura foi executada com sucesso em ambiente local, utilizando Docker Compose para integrar os serviços do Apache Airflow, MinIO e Jupyter. A DAG configurada no Airflow realizou automaticamente a verificação, extração e organização dos dados.

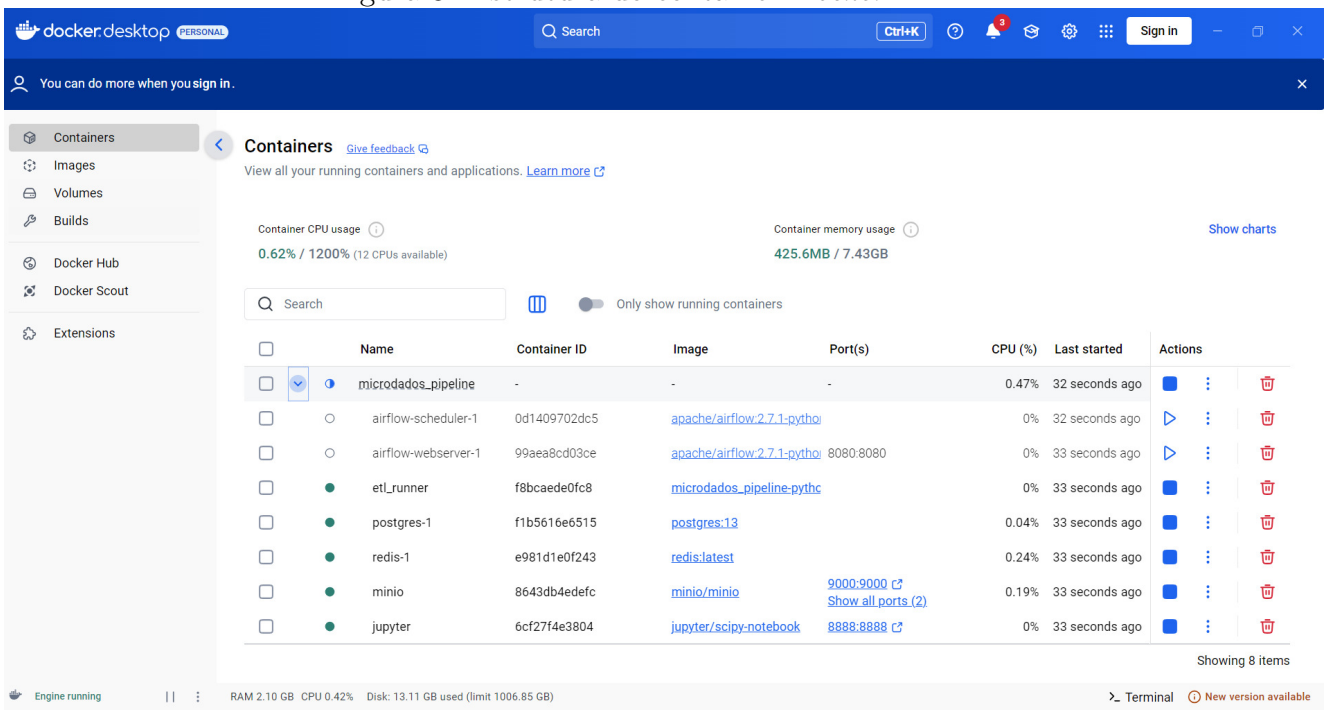
A execução resultou em:

- Download de 1 arquivo ZIP do ENEM 2023;

- Armazenamento na camada bronze do MinIO;
- Descompactação e padronização de todos os arquivos, documentação, entre outros (camada prata);
- Organização final dos dados na camada ouro em CSV.

A Figura 3 apresenta a estrutura no *Docker*, como estão os containers de cada um dos softwares.

Figura 3: Estrutura do container *Docker*

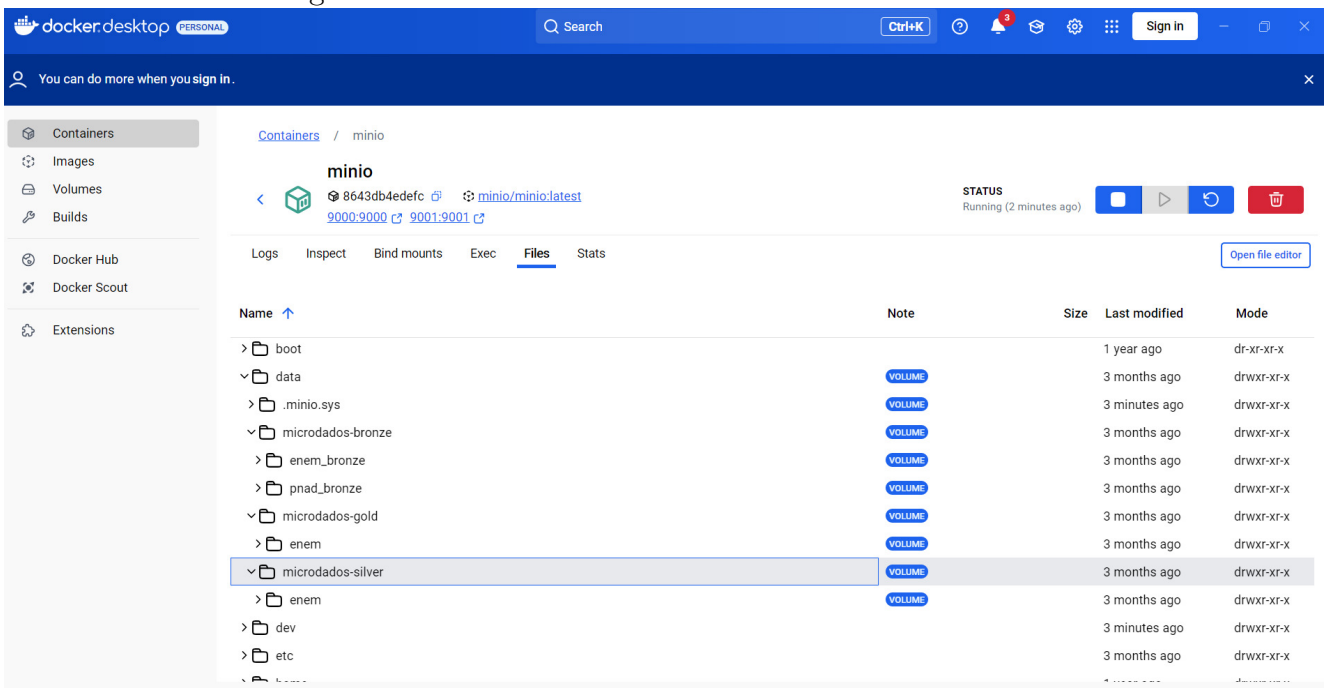


Fonte: Elaboração da autora (2025).

Após a execução do pipeline, os dados foram organizados no MinIO em três camadas distintas, e está ilustrado na Figura 4:

- **Bronze:** contém os arquivos ZIP originais baixados do site do INEP, com os dados crus e metadados como timestamp e nome do arquivo.
- **Prata:** armazena os arquivos CSV descompactados, com colunas padronizadas e estrutura homogênea.
- **Ouro:** armazena os dados tratados, prontos para análises exploratórias e elaboração de relatórios e *dashboards*.

Figura 4: Camadas medalhão no *MinIO Docker*



Fonte: Elaboração da autora (2025).

Todos os scripts desenvolvidos estão disponíveis no repositório público: <https://github.com/ialysousa/arquitetura_microdados_publicos>.

Abaixo, apresenta-se um trecho do código de extração dos dados:

Código – Script para baixar os arquivos do ENEM

```
# 1. URL direta do ENEM 2023
zip_url = 'https://download.inep.gov.br/microdados/
    microdados_enem_2023.zip'
zip_filename = zip_url.split("/")[-1]

# 2. Baixa o ZIP
print(f"Baixando {zip_url} ...")
zip_response = requests.get(zip_url)
zip_response.raise_for_status()
print("Download conclu do!")

# 3. Cria o bucket bronze se nao existir
bucket_name = 'microdados-bronze'
existing = s3.list_buckets()
```

```
if bucket_name not in [b['Name'] for b in existing['Buckets']]:
    s3.create_bucket(Bucket=bucket_name)

# 4. Envia o ZIP para o MinIO
key = f"enem_bronze/{zip_filename}"
print(f"Enviando para MinIO: {bucket_name}/{key}")
s3.upload_fileobj(BytesIO(zip_response.content), bucket_name, key)
print("Envio concluido!")
```

Em seguida, apresenta-se um trecho de código da DAG de orquestração e atualização dos dados.

Código – Script para baixar os arquivos do ENEM

```
# Diretorios
# ========================
DOWNLOAD_DIR = '/app/data/bronze'
SILVER_DIR = '/app/data/silver'
GOLD_DIR = '/app/data/gold'
FILENAME = 'microdados_enem_2023.zip'
URL_BASE = f'https://download.inep.gov.br/microdados/
   microdados_enem_2023.zip'  # substitua se necessario

# ========================
# Fun   es Python
# ========================

def verificar_novos_dados():
    """Verifica se o arquivo ja existe na camada bronze."""
    os.makedirs(DOWNLOAD_DIR, exist_ok=True)
    path = os.path.join(DOWNLOAD_DIR, FILENAME)
    if os.path.exists(path):
        print("Arquivo ja est  presente. Nenhum novo dado
           detectado.")
        return False
    else:
```

```
        print("Novo arquivo detectado.")
        return True

def baixar_dados():
    """Faz o download do arquivo ZIP do ENEM."""
    response = requests.get(URL_BASE, stream=True)
    destino = os.path.join(DOWNLOAD_DIR, FILENAME)

    with open(destino, 'wb') as f:
        for chunk in response.iter_content(chunk_size=8192):
            f.write(chunk)

    print(f"Download concluido: {destino}")

def descompactar_e_padronizar():
    """Extrai os dados da camada bronze e os salva na prata."""
    zip_path = os.path.join(DOWNLOAD_DIR, FILENAME)
    os.makedirs(SILVER_DIR, exist_ok=True)

    with ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(SILVER_DIR)

    print(f"Descompactacao concluida para: {SILVER_DIR}")

def mover_para_ouro():
    """Move os arquivos da camada prata para a camada ouro."""
    os.makedirs(GOLD_DIR, exist_ok=True)
    arquivos = os.listdir(SILVER_DIR)

    for file in arquivos:
        src = os.path.join(SILVER_DIR, file)
        dst = os.path.join(GOLD_DIR, file)
        if os.path.isfile(src):
            os.rename(src, dst)
```

```
    print(f"Arquivos movidos para camada ouro: {GOLD_DIR}")

    with DAG(
    dag_id='automicroetl_enem',
    default_args=default_args,
    description='Pipeline AutoMicroETL para microdados do ENEM',
    schedule_interval='@monthly',
    start_date=days_ago(1),
    catchup=False,
    tags=['enem', 'automicroetl'],
) as dag:

    verificar = PythonOperator(
        task_id='verificar_novos_dados',
        python_callable=verificar_novos_dados
    )

    baixar = PythonOperator(
        task_id='baixar_dados',
        python_callable=baixar_dados
    )

    descompactar = PythonOperator(
        task_id='descompactar_e_padronizar',
        python_callable=descompactar_e_padronizar
    )

    mover = PythonOperator(
        task_id='mover_para_ouro',
        python_callable=mover_para_ouro
    )

    # Orquestracao
```

```
verificar >> baixar >> descompactar >> mover
```

Outros scripts presentes no repositório incluem: descompactação, padronização dos dados, movimentação entre camadas do medalhão.

A arquitetura **AutoMicroETL** demonstrou-se funcional e aderente às boas práticas de engenharia de dados. A modularidade da solução permitiu a separação clara entre as etapas de ingestão, transformação e análise. O uso de *Docker* garantiu a portabilidade e reprodutibilidade do ambiente, enquanto o MinIO permitiu simular um data lake local com estrutura de camadas. O Airflow, por sua vez, possibilitou o agendamento e controle de execução do pipeline de forma automatizada e rastreável.

A arquitetura pode ser facilmente adaptada para outras fontes de microdados públicos, como RAIS, PNAD ou dados do CAGED, validando sua aplicabilidade em diferentes contextos e reforçando sua contribuição para o acesso e organização de dados governamentais.

8 Conclusão

Este trabalho apresentou a **AutoMicroETL**, uma arquitetura modular, automatizada e reutilizável voltada à ingestão e organização de microdados públicos, com ênfase em dados educacionais, como os do ENEM. Diante dos desafios encontrados na manipulação desses dados, como ausência de padronização, formato bruto e necessidade de operações manuais, a solução proposta demonstrou ser eficaz ao integrar ferramentas amplamente utilizadas no ecossistema de dados e ciência aberta, como *Docker*, *Apache Airflow*, *Python* e *MinIO*.

A principal contribuição da arquitetura está na sua flexibilidade e adaptabilidade, pois ela não se limita a um conjunto específico de dados, podendo ser facilmente ajustada para outras fontes públicas, como RAIS, CAGED ou PNAD. Ao adotar o modelo de camadas do data lake (bronze, prata e ouro), a **AutoMicroETL** promove governança de dados, rastreabilidade e reprocessamento controlado, seguindo boas práticas de engenharia de dados.

A automatização das etapas por meio do *Apache Airflow*, aliada à conteinerização via *Docker*, permitiu a criação de um ambiente portátil e escalável, com mínima dependência de recursos computacionais, tornando sua adoção viável tanto em ambientes acadêmicos

quanto em contextos institucionais.

O estudo de caso com os microdados do ENEM 2023 validou a proposta, comprovando a viabilidade técnica e operacional da arquitetura. Como desdobramento futuro, espera-se expandir a aplicação da **AutoMicroETL** para novas bases governamentais, bem como integrar módulos de transformação analítica e visualização interativa dos dados, ampliando ainda mais o acesso e a democratização da informação pública.

Embora a arquitetura **AutoMicroETL** tenha se mostrado funcional e eficiente na automatização da ingestão e organização dos microdados do ENEM 2023, algumas limitações e possibilidades de aprimoramento foram identificadas ao longo do desenvolvimento.

Entre as limitações, destaca-se a ausência de um mecanismo genérico de análise e visualização de dados na camada ouro. Por opção metodológica, essa etapa foi deixada livre para que o usuário final defina as ferramentas e abordagens analíticas mais adequadas ao seu contexto. Contudo, a incorporação de notebooks modelo ou dashboards interativos pode facilitar ainda mais o uso da arquitetura por perfis menos técnicos.

Outra limitação é a dependência de atualizações manuais na definição de URLs de origem dos dados, que podem variar de ano para ano. A criação de um módulo mais robusto de verificação de novas versões, com web scraping ou integração com APIs públicas, ampliaria a autonomia e escalabilidade do pipeline.

Como proposta de trabalho futuro, destaca-se a generalização da arquitetura para abranger múltiplas bases simultaneamente, com configuração parametrizada via arquivos *YAML* ou banco de metadados. Além disso, a integração com ferramentas como *Data Build Tool (DBT)*, *Spark* ou *BigQuery* pode aumentar a eficiência em contextos de maior volume de dados e análises mais sofisticadas.

Finalmente, espera-se que a **AutoMicroETL** possa servir como base para soluções institucionais mais amplas, contribuindo para o fortalecimento da cultura de dados abertos, ciência reprodutível e decisões baseadas em evidências.

Referências

ALVES, H. F.; MARTINS, C.; SILVA, R. Microdados e políticas públicas: desafios de análise em grandes bases de dados educacionais. *Revista Brasileira de Educação*, v. 25, n. e250073, 2020.

BOETTIGER, C. *An introduction to Docker for reproducible research*. 2015. <https://doi.org/10.1145/2723872.2723882>. Publicado na ACM SIGOPS Operating Systems Review, 49(1), pp. 71–79.

BURNS, B. et al. *Borg, Omega, and Kubernetes*. 2016. <https://doi.org/10.1145/2898445.2898447>. Publicado na ACM Queue, 14(1), pp. 70–93.

CAIXETA, H. *Módulo no Python para extrair dados do CAGED*. 2020. Disponível em: <https://www.linkedin.com/pulse/mÃşdulo-python-para-extrair-dados-do-caged-heitor-caixeta/?trackingId=VRTyBuDORqqjAcDadnbzGg%3D%3D>. Acesso em: 10 jun. 2025.

CHUNG, C.-J. et al. *Container-as-a-Service platform with Docker and Kubernetes for cloud-based medical analysis*. 2019. <https://doi.org/10.1007/s10916-018-1106-5>. Publicado na Journal of Medical Systems, 43(1), pp. 1–11.

FONSECA, S. O. D.; NAMEN, A. A. Mineração em bases de dados do inep: uma análise exploratória para nortear melhorias no sistema educacional brasileiro. *Educação em Revista*, SciELO Brasil, v. 32, p. 133–157, 2016.

HILL, K. M. D. *Docker: Up & Running: Shipping Reliable Containers in Production*. Sebastopol, CA: O'Reilly Media, 2016.

INMON, B. *Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump*. 2016. <https://www.technicspub.com/data-lake-architecture/>. 1ª ed. Denville, NJ, USA: Technics Publications, LLC. ISBN 1634621174.

MERKEL, D. *Docker: lightweight Linux containers for consistent development and deployment*. 2014. <https://dl.acm.org/doi/10.5555/2600239.2600241>. Publicado na Linux Journal, 239(2), pp. 2–11.

Microsoft. *O que é arquitetura medallion do Lakehouse?* 2025. <https://learn.microsoft.com/pt-br/azure/databricks/lakehouse/medallion>. Documentação oficial da Microsoft. Acesso em: 20 jun. 2025.

MINIO. *High Performance Object Storage*. Disponível em: <https://min.io/>. Acesso em: 10 jun. 2025.

MURRIETA, M. *Ciência Aberta, tema que integra o 6º Plano de Ação de Governo Aberto é discutido com a sociedade durante a 76ª Reunião Anual da Sociedade Brasileira para o Progresso da Ciência e durante a 5ª Conferência Nacional de Ciência, Tecnologia e Inovação*. 2024. <https://www.gov.br/cgu/pt-br/governo-aberto/artigos/ciencia-aberta-uma-nova-forma-de-fazer-ciencia-mais-colaborativo-transparente-e-sustentavel/ciencia-aberta-um-novo-modo-de-fazer-ciencia-mais-colaborativo-transparente-e-sustentavel>. Publicado pelo Ministério da Ciência, Tecnologia e Inovação em 15 ago. 2024.

NEVES, O. M. D. C. Evolução das políticas de governo aberto no brasil. In: *Anais do VI Congresso Brasileiro de Gestão Pública – CONSAD*. Brasília, Brasil: [s.n.], 2013.

PALMER, P. *Data Lake Pessoal Eficiente para Análise de Dados*. 2022. <https://repozitorij.uni-lj.si/IzpisGradiva.php?id=139064>. Tese de Mestrado — Universidade de Liubliana, Faculdade de Ciência da Computação e Informática. Programa de Estudos de Segundo Nível de Mestrado em Computação e Informática.

SINHA, R. Automation of data pipelines in machine learning workflows: Trends, tools, and challenges. *International Journal of Advanced and Innovative Research*, v. 17, n. 4, p. 219, 2024. ISSN 2278-7844. Department of Information Technology.

SOLTESZ, S. et al. *Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors*. 2007. <https://doi.org/10.1145/1272996.1273028>. Publicado na ACM SIGOPS Operating Systems Review, 41(3), pp. 275–287.

VAUGHAN-NICHOLS, S. J. *The amazing Docker: Bringing containers for Linux to your desktop*. 2016. <https://doi.org/10.1109/MC.2016.113>. Publicado na Computer, 49(4), pp. 70–73.

WISELKA, M. *Development of Modern Data Platform using Medallion Architecture*. 2024. <https://www.theseus.fi/handle/10024/803972>. Bachelor's Thesis — Haaga-Helia University of Applied Sciences, Degree Programme in Business Information Technology.

WOODS, D. *Big Data Requires a Big, New Architecture*. 2011. <https://www.forbes.com/sites/ciocentral/2011/07/21/big-data-requires-a-big-new-architecture/?sh=5557786f1157>. Publicado na Forbes em 21 jul. 2011. Acesso em: 15 jan. 2021.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Restrito

Versão final do TCC

Assunto:	Versão final do TCC
Assinado por:	Ialy Sousa
Tipo do Documento:	Tese
Situação:	Finalizado
Nível de Acesso:	Restrito
Hipótese Legal:	Informação Pessoal (Art. 31 da Lei no 12.527/2011)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Ialy Cordeiro de Sousa**, **ALUNO (201921250043) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 03/09/2025 20:40:09.

Este documento foi armazenado no SUAP em 03/09/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1596709
Código de Autenticação: e4b2f17284