UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Gabriel Antonio Camilo Ferrazzo

Construção de back-end de hub de análise de metadados sobre repositórios de dados

Florianópolis
2023

Gabriel Antonio Camilo Ferrazzo

Construção de back-end de hub de análise de metadados sobre repositórios de dados

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Rodrigo Castelan Carlson, Dr.
Supervisor: Marcelo Nogueira Araújo

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Ferrazzo, Gabriel Antonio Camilo
Construção de back-end de hub de análise de metadados sobre repositórios de dados / Gabriel Antonio Camilo Ferrazzo ; orientador, Rodrigo Castelan Carlson, 2024.
47 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. back-end. 3. catálogo de dados. I. Carlson, Rodrigo Castelan . II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Gabriel Antonio Camilo Ferrazzo

Construção de back-end de hub de análise de metadados sobre repositórios de dados

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 26 de fevereiro de 2024.

Prof. Marcelo de Lellis Costa de Oliveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Rodrigo Castelan Carlson, Dr.
Orientador
UFSC/CTC/DAS

Marcelo Nogueira Araújo
Supervisor
Radix Engenharia e Software

Prof. Leandro Becker, Dr.
Avaliador
UFSC/CTC/DAS

Prof. Hector Bessa Silveira, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado à minha mãe, a meu falecido pai e aos amigos Virgílio e George, principais responsáveis pelo caminho que trilhei e de que tanto me orgulho.

AGRADECIMENTOS

Agradeço a toda minha família que me serviu de grande rede de apoio nesta longa jornada acadêmica.

Agradeço, também, a todos membros e ex-membros da Equipe UFSC de Eficiência Energética, que foi de grande papel na minha carreira acadêmica e profissional, além dos inúmeros colegas que a Automação trouxe, e que foram outra grande rede de apoio no final da vida acadêmica.

Por fim, grandes agradecimentos ao Marcelo Araujo e Daniel Franzoni, pelas oportunidades, preocupações e momentos de paciência.

“Em tecnologia, a capacidade de aprender e se adaptar rapidamente é tão importante quanto o conhecimento que você já possui.”
(IMAFIDON, Anne-Marie)

DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 26 de fevereiro de 2024.

Na condição de representante da Radix na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o autor, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.

gov.br
Documento assinado digitalmente
MARCELO NOGUEIRA ARAUJO
Data: 15/02/2024 15:22:48-0300
Verifique em https://validar.iti.gov.br

Marcelo Nogueira Araújo
Radix Engenharia e Software

RESUMO

Este trabalho aborda a implementação de um back-end destinado a um hub de análise de metadados, projetado para integrar informações de diversos repositórios de dados. A iniciativa fundamenta-se na democratização do acesso a esses repositórios por uma empresa de grande porte, esta é uma tendência contemporânea que busca simplificar o acesso a diversas fontes, com isso, são reduzidos duplicidades e otimizado processos. Esse trabalho destaca as ferramentas e tecnologias empregadas no desenvolvimento, com ênfase em boas práticas de codificação, escalabilidade do produto e usabilidade para os usuários. Adicionalmente, o sistema é projetado para expansão, mantendo a flexibilidade para a incorporação de novas funcionalidades.

Palavras-chave: Metadados. Back-end. Democratização de dados.

ABSTRACT

This study addresses the implementation of a backend system for a metadata analysis hub, designed to integrate information from various data repositories. The initiative is based on democratizing access to these repositories for a large company, reflecting a contemporary trend aiming to simplify access to diverse sources, thereby reducing redundancies and optimizing processes. This work highlights the tools and technologies employed in the development, with a focus on coding best practices, product scalability, and user usability. Additionally, the system is designed for expansion, maintaining flexibility for the incorporation of new functionalities.

Keywords: Metadata. Back-end. Data democratization.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A democratização de dados refere-se ao processo de tornar os dados acessíveis e disponíveis para um público mais amplo, possibilitando que indivíduos, técnicos ou não, possam utilizar e beneficiar-se dessas informações, seja em uma instituição privada, uma organização governamental ou qualquer outra instituição que possa se beneficiar de uma estrutura de dados (MICROSOFT, 2024b). Essa abordagem visa superar barreiras tradicionais de acesso aos dados, promovendo a transparência, a igualdade e a participação mais ampla na análise e tomada de decisões. Busca eliminar restrições injustificadas ao acesso, permitindo que uma variedade de usuários, independentemente de sua posição ou habilidades técnicas, possam explorar e utilizar dados para impulsionar a inovação, melhorar processos e contribuir para a resolução de problemas.

Em uma empresa de grande porte, a democratização de dados é uma iniciativa crucial para promover a acessibilidade e a utilização eficaz de informações por todos os membros da organização, independente de suas funções ou departamentos. No entanto, para efetivar essa democratização, é essencial construir um catálogo centralizado de dados.

Um catálogo de dados é uma ferramenta que serve como um repositório centralizado para armazenar e gerenciar informações sobre os dados disponíveis na organização. Esse catálogo funciona como um inventário abrangente, fornecendo metadados detalhados sobre conjuntos de dados, suas origens, estruturas, formatos e outros atributos relevantes.

O objetivo principal de um catálogo de dados é facilitar a descoberta, compreensão e utilização eficiente dos recursos de dados dentro da empresa. Ele pode incluir informações sobre a propriedade dos dados, as políticas de acesso, a qualidade dos dados, as relações entre conjuntos de dados e outras informações contextuais. Além disso, um catálogo de dados pode ser integrado a outras ferramentas de gerenciamento de dados, facilitando a colaboração entre equipes e promovendo a consistência no uso de informações em toda a organização.

Esta ferramenta desempenha um papel fundamental na eliminação de redundâncias e na otimização das operações dos times de software. Isso porque, a falta de governança levou os datalakes a se transformarem em “pântanos de dados”. Isto é, enormes repositórios que armazenam dados cujo conteúdo e origem são desconhecidos por todos, e que ninguém sabe utilizar (OLESEN-BAGNEUX, 2023).

A ausência de um catálogo de dados, ou seja, não existir nenhum tipo de guia ou direcionamento sobre a função, composição e origens dos dados existentes na companhia, de forma centralizada, pode resultar em trabalho duplicado, onde diferentes equipes acabam criando processos de ingestão de dados para fontes similares.

Tal redundância não apenas consome recursos desnecessários, mas também gera inconsistências nos processos, levando a interpretações divergentes dos dados.

Além disso, a falta de uma plataforma centralizada dificulta a padronização e a aplicação de diretrizes consistentes para a ingestão, armazenamento e uso de dados. A inconsistência nos processos pode levar a ambiguidades, comprometendo a confiabilidade das informações disponíveis para os times de software.

Ao centralizar a gestão de dados, um catálogo proporciona eficiência operacional, permitindo que os times de software concentrem seus esforços no desenvolvimento e na inovação, em vez de lidar com tarefas operacionais repetitivas relacionadas à gestão de dados. Essa abordagem também facilita a colaboração entre diferentes equipes, promovendo uma cultura integrada e colaborativa.

O papel central dos metadados é algo que as organizações tradicionais tendem em ter dificuldade em compreender. O gerenciamento de metadados corporativos foi muitas vezes menosprezado, mas é um dos fatores mais importantes para aproveitar completamente o potencial das novas arquiteturas de dados. De acordo com Olesen-Bagneux (2023), não é possível implantar uma estratégia de dados em larga escala sem um catálogo adequadamente estruturado e mantido.

O gerenciamento centralizado de metadados proporciona uma visão mais clara da origem e do contexto dos dados, facilitando a compreensão por parte dos times de software. Isso é essencial para o desenvolvimento de soluções eficazes e para a adaptação rápida a novas tecnologias e fontes de dados, fatores críticos em um ambiente empresarial dinâmico e em constante evolução.

Esse Projeto Final de Curso trata da construção de um back-end de um catálogo de análise de dados sobre repositórios de dados dentro de um cliente da empresa em que o autor realizou estágio. Neste documento serão apresentados uma breve introdução da empresa, as metodologias e tecnologias utilizadas, detalhes sobre como o sistema funciona, e, por fim, seus resultados.

A maior parte do que é apresentado nesse documento foi de responsabilidade do próprio autor do documento, porém o time da construção do back-end era composto por mais um membro, com mais experiência, que serviu de fonte de consultoria e ajudou a solucionar problemas mais complexos, além de agilizar algumas tarefas quando os prazos eram mais curtos.

1.1 OBJETIVOS

O objetivo deste trabalho é construir um back-end para esse catálogo de dados descrito acima. Dessa forma, o trabalho segue diretrizes muito similares as do produto em si, aplicando os requisitos gerais do catálogo de dados às tecnologias utilizadas para a construção do back-end.

A ideia, desde a proposta inicial, é que essa ferramenta começasse como um

Mínimo Produto Viável (do inglês Minimum Viable Product – MVP) ou seja, que fosse construído um produto de menor complexidade, prazo e número de features possível; um produto que demonstre seu potencial com o mínimo esforço necessário.

Por ser concebido desde o início como um MVP, precisou-se levar em conta que seria necessário apresentar um produto que, de fato, pudesse seguir adiante. Porém, ainda foi necessário que se tomasse cuidado com as tecnologias escolhidas para que fossem escaláveis num futuro em que a ferramenta continue a ser desenvolvida.

Um dos objetivos gerais do MVP era que se pudesse visualizar metadados de todas camadas (desconsiderando as camadas sensíveis) do DataLake da companhia. Como será explicado no capítulo das Tecnologias, esse requisito é muito similar ao produto “Purview” da Microsoft.

Entretanto, outro objetivo dessa ferramenta (que foge do escopo do projeto do autor desse documento) é que ela evolua, no futuro. A ideia é que se torne um catálogo central de todos dados da companhia, que sirva para visualização dos dados, a seguir como um ponto de comunicação entre todos times de software que lidem com algum dado do DataLake (ou até de outras fontes), até tornar-se uma central de solicitação de acessos e permissões.

Por isso, um dos objetivos principais desse trabalho é a garantia da fácil escalabilidade da ferramenta, preocupação que permeou a escolha de muitas tecnologias.

1.2 A EMPRESA

A Radix é uma empresa brasileira de engenharia e software que atua em diversos setores, incluindo energia, petróleo e gás, telecomunicações, financeiro e manufatura. A empresa foi fundada em 2010 e tem sua sede no Rio de Janeiro, Brasil.

A Radix é conhecida por fornecer soluções de engenharia e desenvolvimento de software, incluindo serviços de automação industrial, sistemas de controle, integração de sistemas, inteligência artificial, internet das coisas (IoT) e outras tecnologias avançadas. Ela trabalha com clientes no Brasil e no exterior, oferecendo soluções personalizadas para atender às necessidades específicas de cada setor.

A empresa tem uma reputação sólida no mercado de engenharia e tecnologia, e sua atuação abrange desde o projeto e desenvolvimento de sistemas complexos até a implementação de soluções inovadoras para otimizar processos industriais e empresariais (RADIX, 2024).

1.3 ESTRUTURA DO DOCUMENTO

No capítulo 2 serão abordados a Metodologia e as Tecnologias utilizadas no desenvolvimento do trabalho.

No capítulo 3 serão apresentados os requisitos dados pela gerência do projeto e, por sequência, como esses requisitos levaram à escolha das tecnologias empregadas.

No capítulo 4 serão apresentados como foram implementadas as tarefas de automatização necessárias para a construção da ferramenta.

No capítulo 5 serão apresentados a abordagem e construção das execuções em tempo real (majoritariamente a API) da ferramenta.

No capítulo 6 serão apresentados os principais resultados visíveis pelo front-end da ferramenta.

No capítulo 7 será apresentada uma breve conclusão retomando o que foi aqui trabalhado e a opinião do autor sobre o trabalho.

2 METODOLOGIA E TECNOLOGIAS

2.1 METODOLOGIA

Scrum é uma metodologia ágil de gerenciamento de projetos que tem ganhado destaque significativo no cenário empresarial contemporâneo. Desenvolvida inicialmente para a gestão de projetos de software, essa abordagem flexível e iterativa tem sido amplamente adotada em diversas áreas, indo além do desenvolvimento de software. Baseada em princípios de transparência, inspeção e adaptação, o Scrum oferece uma estrutura eficaz para equipes colaborativas atingirem metas complexas e dinâmicas.

No cerne do Scrum está a ideia de dividir o projeto em ciclos curtos, conhecidos como sprints, que geralmente têm uma duração de duas a quatro semanas. Cada sprint é composto por etapas bem definidas, como planejamento, execução e revisão, promovendo a entrega incremental de funcionalidades. O papel crucial do Scrum Master, que atua como facilitador e guardião dos princípios do Scrum, contribui para o ambiente colaborativo, removendo obstáculos e fomentando a auto-organização da equipe.

A comunicação eficaz é um pilar fundamental do Scrum, sendo promovida por meio de reuniões regulares, como as Daily Scrum Meetings, onde os membros da equipe compartilham atualizações e discutem possíveis impedimentos. A utilização de artefatos visuais, como o Product Backlog e o Sprint Backlog, proporciona transparência ao processo, permitindo que todos os stakeholders compreendam o progresso do projeto.

Ao adotar o Scrum, as organizações podem experimentar melhorias significativas na flexibilidade, adaptabilidade e produtividade. A natureza iterativa da metodologia permite uma resposta rápida às mudanças nos requisitos do projeto, resultando em entregas mais alinhadas com as expectativas do cliente. Em resumo, o Scrum destaca-se como uma abordagem dinâmica e colaborativa para gerenciar projetos, promovendo a agilidade e a eficiência na busca por resultados excepcionais (SUTHERLAND, 2014).

2.2 TECNOLOGIAS

2.2.1 Microsoft Azure

A Microsoft Azure é uma plataforma de serviços em nuvem oferecida pela Microsoft, projetada para atender às necessidades de empresas e organizações em suas operações digitais. Essa plataforma abrangente fornece uma ampla gama de serviços, incluindo hospedagem de máquinas virtuais, armazenamento de dados, análise de big data, inteligência artificial e aprendizado de máquina, além de soluções para desenvolvimento e implantação de aplicativos. A Azure permite que empresas dimensionem e

ajustem seus recursos de computação de acordo com as demandas específicas, proporcionando flexibilidade e eficiência operacional. Sua infraestrutura global distribuída garante alta disponibilidade e desempenho, enquanto as rigorosas medidas de segurança proporcionam um ambiente confiável para o armazenamento e processamento de dados sensíveis (MICROSOFT, 2024e).

Ao adotar a Microsoft Azure, as organizações podem se beneficiar da agilidade, escalabilidade e inovação contínua que a computação em nuvem oferece. Essa plataforma facilita a transformação digital, permitindo que as empresas otimizem suas operações, reduzam custos e acelerem o desenvolvimento de soluções tecnológicas. Com uma ampla variedade de ferramentas e serviços integrados, a Azure se destaca como uma escolha abrangente para empresas que buscam uma infraestrutura confiável e avançada na era digital, proporcionando um ambiente propício para a criação e implementação de soluções tecnológicas inovadoras.

2.2.1.1 Synapse Analytics

O Microsoft Azure Synapse Analytics é uma solução avançada de análise de dados que integra armazenamento, big data e análise de dados em uma única plataforma unificada. Projetado para atender às demandas crescentes de organizações em relação ao processamento e análise de grandes volumes de dados, o Synapse Analytics oferece uma abordagem integrada e escalável para lidar com todas as etapas do ciclo de vida dos dados analíticos (MICROSOFT, 2024a).

Essa ferramenta é particularmente valiosa para empresas que buscam insights significativos a partir de seus dados, permitindo a consulta e análise de informações em tempo real. O Synapse Analytics suporta uma variedade de fontes de dados, desde dados estruturados a não estruturados, possibilitando a integração de informações provenientes de diferentes fontes para uma análise mais abrangente. Além disso, sua capacidade de processamento paralelo distribuído e integração com ferramentas familiares, como o Power BI, facilita a criação de dashboards e relatórios interativos.

2.2.1.2 Data Lake Storage

O Microsoft Azure Data Lake Storage Gen2 é uma solução avançada de armazenamento em nuvem projetada para lidar com grandes volumes de dados, proporcionando uma plataforma robusta para armazenamento, processamento e análise de informações. Baseado na arquitetura do Azure Blob Storage, o Gen2 aprimora as funcionalidades do Data Lake Storage original, incorporando recursos como uma hierarquia de diretórios e uma camada de acesso baseada em POSIX, que melhora a eficiência na organização e navegação por grandes conjuntos de dados.

Essa solução oferece um ambiente seguro e altamente escalável para dados estruturados e não estruturados, permitindo a integração fácil com ferramentas analí-

ticas e serviços do ecossistema Azure. Com suporte a protocolos padrão do setor e integração com ferramentas de análise de dados, o Data Lake Storage Gen2 capacita as organizações a gerenciar, processar e extrair insights valiosos de seus dados em larga escala, promovendo uma abordagem integrada e eficiente para o gerenciamento de dados em ambientes de nuvem.

2.2.1.3 Active Directory

O Azure Active Directory (Azure AD) é um serviço de gerenciamento de identidade e acesso baseado em nuvem fornecido pela Microsoft, que desempenha um papel fundamental na administração e controle de acesso a recursos digitais em ambientes corporativos modernos. Ele serve como um serviço de diretório na nuvem, permitindo que organizações controlem e protejam o acesso a aplicativos, dados e outros recursos, tanto na nuvem quanto localmente, de maneira segura e eficiente (MICROSOFT, 2024d).

No âmbito de sua função, o Azure AD oferece uma gama abrangente de recursos e funcionalidades que visam simplificar e aprimorar a gestão de identidades e o controle de acesso em uma variedade de cenários de implantação. Entre esses recursos, destacam-se a autenticação multifator, a federação de identidades, o gerenciamento de dispositivos, a governança de identidade, a autenticação única (SSO), a autorização baseada em políticas, entre outros.

2.2.2 Pomerium

Pomerium é uma ferramenta de proxy de autenticação que pode ser integrada com o Azure Active Directory (Azure AD) para fornecer autenticação e autorização baseadas em políticas em aplicativos e serviços que não são nativamente compatíveis com o Azure AD (POMERIUM, 2024).

Essa integração ocorre por meio do Pomerium atuando como um intermediário entre os usuários e os aplicativos, aplicando políticas de acesso definidas pelo administrador. Quando um usuário tenta acessar um aplicativo protegido pelo Pomerium, o Pomerium intercepta a solicitação de acesso, autentica o usuário por meio do Azure AD e, em seguida, autoriza ou nega o acesso com base nas políticas configuradas.

2.2.3 Purview

O Microsoft Purview é uma solução abrangente de governança de dados que capacita as organizações a descobrir, classificar, mapear e controlar seus ativos de dados em toda a empresa. Integrando-se ao ecossistema de serviços de dados do Azure, o Purview oferece uma visão holística e unificada do panorama de dados, permitindo que as empresas compreendam melhor o que está contido em seus diversos

repositórios, tanto na nuvem quanto em ambientes locais. Além disso, o Purview facilita a conformidade com regulamentações e políticas de privacidade, ao automatizar processos de descoberta e classificação de dados sensíveis, fornecendo ferramentas para rastrear e monitorar o uso desses dados.

Ao proporcionar uma governança eficaz dos dados, o Microsoft Purview contribui para a construção de uma cultura organizacional baseada em dados confiáveis e seguros. Ele oferece funcionalidades avançadas, como a criação de mapas de dados, permitindo uma compreensão visual das relações entre conjuntos de dados, bem como a integração com outras ferramentas e serviços do ecossistema Microsoft e de terceiros. Com o Purview, as empresas podem maximizar o valor de seus dados, ao mesmo tempo em que garantem uma gestão responsável e transparente, essencial em um cenário empresarial cada vez mais orientado por informações (MICROSOFT, 2024c).

2.2.4 Apache

2.2.4.1 Airflow

O Apache Airflow é uma plataforma de gerenciamento de workflow (fluxo de trabalho) de código aberto. Tudo começou no Airbnb em outubro de 2014 como uma solução para gerenciar os workflows cada vez mais complexos da empresa. A criação do Airflow permitiu ao Airbnb criar e desenvolver programaticamente seus workflows e monitorá-los através da interface de usuário incorporada do Airflow. Desde o início, o projeto foi aberto, tornando-se um projeto da Incubadora Apache em março de 2016 e um projeto da Apache Software Foundation de alto nível em janeiro de 2019 (APACHE, 2024).

O Airflow é escrito em Python, e os workflows são criados por meio de scripts Python. O Airflow é projetado sob o princípio de configuração como código. Enquanto outras plataformas de workflow de configuração como código existem usando linguagens de marcação como XML, o uso do Python permite que os desenvolvedores importem bibliotecas e classes para ajudá-los a criar seus workflows.

O Airflow usa gráficos acíclicos direcionados (DAGs) para gerenciar a orquestração dos workflows. Tasks (tarefas) e dependências são definidas em Python e, em seguida, o Airflow gerencia o agendamento e a execução. Os DAGs podem ser executados em um agendamento definido (por exemplo, a cada hora ou diariamente) ou com base em gatilhos de eventos externos (por exemplo, um arquivo que aparece em um banco de dados). No Airflow, os DAGs geralmente podem ser gravados em um único arquivo Python, enquanto em outras ferramentas - também baseadas em DAGs - são necessários múltiplos arquivos de configuração e árvores de sistema de arquivos.

2.2.4.2 Spark

O Apache Spark é um poderoso framework de computação distribuída projetado para processar grandes volumes de dados de forma rápida e eficiente. Desenvolvido para oferecer alto desempenho em tarefas de processamento de dados em larga escala, o Spark fornece uma abstração avançada de programação, permitindo o processamento paralelo e distribuído em clusters de computadores. Sua versatilidade é evidente na capacidade de suportar diversas linguagens de programação, como Scala, Java, Python e SQL, tornando-o acessível a uma ampla gama de desenvolvedores e cientistas de dados.

O Apache Spark é especialmente conhecido por sua arquitetura resiliente e tolerante a falhas, além de oferecer módulos integrados para processamento de dados em lote (Batch), streaming em tempo real, aprendizado de máquina e processamento de gráficos. Isso faz do Spark uma ferramenta central em ecossistemas de big data, permitindo a análise eficiente de dados complexos e a implementação de pipelines de dados completos. Sua popularidade e adoção generalizada são reflexos de sua flexibilidade e desempenho, tornando-o essencial para empresas que buscam extrair valor de grandes conjuntos de dados em ambientes distribuídos.

2.2.5 MongoDB

O MongoDB é um banco de dados NoSQL de código aberto, projetado para lidar com dados não estruturados e semiestruturados, proporcionando flexibilidade e escalabilidade para aplicativos modernos. Diferentemente dos bancos de dados relacionais tradicionais, o MongoDB armazena dados em documentos BSON (formato binário JSON), permitindo a representação de informações complexas de maneira intuitiva. Essa abordagem NoSQL elimina a necessidade de esquemas predefinidos, oferecendo uma resposta dinâmica às necessidades em constante evolução dos desenvolvedores e de suas aplicações.

O MongoDB destaca-se pela sua capacidade de escala horizontal, distribuindo dados em clusters para atender a demandas crescentes de volume e tráfego. Ele suporta operações de leitura e gravação de maneira eficiente, proporcionando desempenho rápido em ambientes de produção. Além disso, sua flexibilidade é evidente na capacidade de consultas complexas e na indexação eficiente, enquanto sua integração com diversas linguagens de programação e frameworks facilita a adoção por desenvolvedores. O MongoDB é uma escolha popular para aplicações que requerem agilidade, escalabilidade e facilidade de adaptação aos requisitos variados de dados em constante evolução.

2.2.6 Docker

O Docker é uma plataforma de código aberto que simplifica o desenvolvimento, implantação e execução de aplicativos em contêineres. Os contêineres são ambientes isolados e leves que encapsulam uma aplicação e suas dependências, permitindo que ela seja executada de maneira consistente em qualquer ambiente que suporte o Docker. Essa abordagem revolucionou a forma como as aplicações são distribuídas, tornando o processo mais eficiente, confiável e fácil de gerenciar.

Ao utilizar o Docker, os desenvolvedores podem empacotar seu código, bibliotecas e configurações em um contêiner, eliminando inconsistências entre ambientes de desenvolvimento, teste e produção. A portabilidade dos contêineres facilita a implantação em diferentes infraestruturas, seja localmente em laptops, em servidores na nuvem ou em ambientes de orquestração como o Kubernetes. Além disso, o Docker simplifica o dimensionamento de aplicativos, permitindo que múltiplos contêineres sejam executados em uma única máquina, otimizando recursos e oferecendo maior eficiência na implantação e gerenciamento de serviços (DOCKER, 2024).

2.2.6.1 Dev Container

Os Dev Containers (Containers de Desenvolvimento) referem-se a uma prática que utiliza tecnologias como o Docker para criar ambientes de desenvolvimento consistentes e reprodutíveis. Esses ambientes são configurados em contêineres, permitindo que os desenvolvedores compartilhem facilmente as configurações do ambiente de desenvolvimento, independente do sistema operacional utilizado. A proposta principal dos Dev Containers é proporcionar um ambiente padronizado, com todas as dependências e configurações necessárias para o desenvolvimento de um projeto, garantindo consistência entre as máquinas dos desenvolvedores e facilitando a colaboração em equipes distribuídas.

Ao adotar Dev Containers, os desenvolvedores podem definir e versionar as configurações do ambiente de desenvolvimento diretamente no repositório do código-fonte, o que contribui para uma integração mais suave e rápida de novos membros na equipe. Ferramentas populares, como o Visual Studio Code, oferecem suporte nativo a Dev Containers, permitindo que os desenvolvedores iniciem rapidamente seu ambiente de desenvolvimento com apenas alguns cliques. Essa abordagem agiliza o processo de desenvolvimento, reduzindo as discrepâncias entre ambientes e facilitando a manutenção e colaboração em projetos complexos.

2.2.7 FastAPI

O FastAPI é um framework moderno para o desenvolvimento de APIs em Python, projetado para ser rápido, fácil de usar e altamente eficiente. Construído sobre as

poderosas funcionalidades do Python 3.7+ e do tipo de anotações, o FastAPI oferece uma sintaxe declarativa que permite aos desenvolvedores criar APIs RESTful com facilidade, sem sacrificar desempenho. Sua combinação única de geração automática de documentação interativa, suporte a tipos de dados e validação automática faz com que seja uma escolha popular para desenvolvedores que buscam rapidez no desenvolvimento sem comprometer a segurança e a robustez (RAMÍREZ, 2024).

O suporte nativo para a especificação OpenAPI e JSON Schema no FastAPI simplifica a criação e manutenção de APIs, enquanto a geração automática de documentação Swagger facilita a compreensão e o teste dos endpoints. Além disso, a eficiência do FastAPI é notável devido ao uso do motor ASGI (*Asynchronous Server Gateway Interface*), que permite a manipulação eficiente de solicitações concorrentes, tornando-o adequado para aplicações em tempo real. Com uma crescente base de usuários e uma comunidade ativa, o FastAPI continua a ganhar popularidade como uma ferramenta ágil e poderosa para o desenvolvimento de APIs em Python.

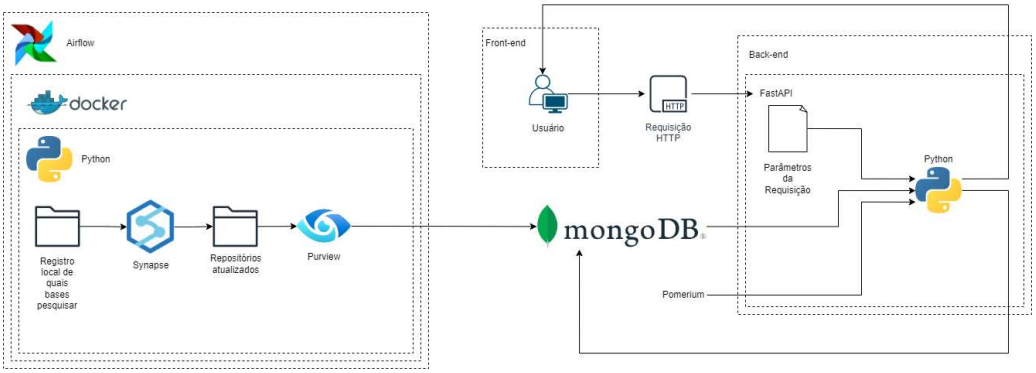


Figura 1 – Diagrama semi-genérico do funcionamento do back-end do catálogo de dados

3 ESTUDOS SOBRE A COMPOSIÇÃO DA FERRAMENTA

3.1 REQUISITOS DA FERRAMENTA

A gerência do projeto indicou os principais requisitos da ferramenta e foi possível, então, estudar quais as tecnologias seriam utilizadas na composição do back-end. Os requisitos apresentados para o time de back-end foram:

Função	Visibilidade	Atributo	Categoria
Autenticação de usuário	Oculto	Segurança	Obrigatório
Registro de acessos de usuários	Oculto	Audiência	Obrigatório
Atualização dos dados diária	Oculto	Periodicidade	Desejável
Tags para cada repositório	Evidente	Classificação	Obrigatório
Favoritar repositórios	Evidente	Usabilidade	Obrigatório
Comentários e sub-comentários	Evidente	Usabilidade	Obrigatório
Repositórios acessíveis	Evidente	Usabilidade	Obrigatório
Endpoint de perfilagem de dados	Oculto	Usabilidade	Desejável

Tabela 1 – Requisitos de back-end do catálogo de dados

3.1.1 Autenticação de usuário

No âmbito da necessidade de implementação de um sistema de autenticação, uma análise detalhada foi conduzida para determinar a abordagem mais adequada. Inicialmente, considerou-se a viabilidade de adotar um sistema de login via back-end, o que implicaria a criação de um banco de dados dedicado aos usuários. Este banco, por sua vez, seria administrado por um responsável designado, incumbido das tarefas de adição e remoção de acessos.

Contudo, em interlocução com a gerência, chegou-se ao entendimento de que todos os usuários da empresa deveriam ter acesso ao sistema. Em virtude dessa compreensão, emergiu a sugestão de empregar o sistema Pomerium integrado à Azure Active Directory (AD). Essa escolha apresentou-se como uma alternativa viável, uma vez que possibilita a utilização das credenciais já existentes dos usuários na Azure AD, eliminando a necessidade de criar e gerenciar um banco de dados separado.

A vantagem crucial dessa abordagem reside na automação da inclusão e exclusão de usuários. Ao integrar-se à Azure AD, o sistema se beneficia da capacidade intrínseca dessa solução de autenticação externa para gerenciar dinamicamente a lista de usuários autorizados, tornando o processo mais eficiente e seguro.

Além disso, ao explorar essa temática em discussões subsequentes, percebeu-se que sua implementação era mais propícia no front-end da aplicação. Essa decisão, prontamente acatada pela gerência, está alinhada com as práticas modernas de desenvolvimento, onde a responsabilidade pela lógica de autenticação e autorização no

front-end é favorecida quando possível, proporcionando uma experiência de usuário mais ágil e responsiva.

3.1.2 Audiência dos acessos

Em resposta à necessidade de manter um registro completo das interações dos usuários com a ferramenta, foi estabelecida a exigência de documentar detalhadamente cada acesso. Esse registro deve abranger informações como a identidade do usuário, timestamp de login, páginas acessadas e outros fatores relevantes. O propósito subjacente a essa medida é atender às exigências de auditoria de acessos, visando garantir uma visão abrangente e rastreável da utilização da ferramenta.

Com o intuito de assegurar a acessibilidade e utilidade desse registro, deliberou-se pelo seu registro em um DataLake, conforme requisitado. Esta decisão foi guiada pela necessidade de permitir a análise da audiência por meio de diversas plataformas, indo além da própria ferramenta aqui descrita. O DataLake, com sua capacidade de armazenar grandes volumes de dados em formatos variados, se configura como o ambiente ideal para a preservação desse registro, possibilitando consultas analíticas e relatórios de acessos de forma eficiente.

A inclusão do DataLake como destino do registro de acessos representa uma escolha estratégica para garantir a integridade, escalabilidade e acessibilidade dos dados de audiência. Essa abordagem não apenas atende às demandas atuais de auditoria, mas também antecipa as necessidades futuras de análise de dados, proporcionando uma solução robusta e flexível para a gestão da audiência da ferramenta.

3.1.3 Período de atualizações

No âmbito da atualização dos dados provenientes do Synapse e do Purview, foi estabelecido que essa operação poderia ser realizada em uma frequência diária. Em outras palavras, a ferramenta não necessitaria apresentar informações em tempo real dessas fontes, considerando que o processo de carga de dados, conforme concebido durante o planejamento da ferramenta, demandaria, possivelmente, mais de 10 minutos.

A decisão de optar por atualizações diárias foi tomada com base em uma ponderação das necessidades do sistema e nas limitações técnicas inerentes ao carregamento de dados. A análise do tempo necessário para a obtenção e integração de dados do Synapse e do Purview revelou que um intervalo diário seria suficiente para manter as informações da ferramenta alinhadas com os dados mais recentes dessas fontes, sem comprometer a eficiência do sistema.

É importante destacar que, ao considerar o panorama técnico e os requisitos de desempenho, a opção por atualizações diárias oferece uma abordagem equilibrada entre a necessidade de dados atualizados e a viabilidade operacional da ferramenta.

Essa escolha estratégica visa otimizar o desempenho do sistema, garantindo a consistência e a integridade dos dados sem comprometer a eficiência operacional.

3.1.4 Tags para classificação dos itens

Cada item pode ser associado a “tags” que definem sua categoria. Foi requisitado que seja viável, por meio da interface front-end para o back-end, adicionar, remover e acessar tags de cada item.

Essa funcionalidade busca proporcionar flexibilidade na categorização de itens, permitindo a manipulação eficiente das tags associadas a cada item por meio da interface da aplicação.

3.1.5 Favoritar itens

Cada usuário pode favoritar ou desfavoritar um item. Será possível acessar os favoritos de um usuário e listar quem favoritou um item.

Essa funcionalidade busca permitir aos usuários gerenciar seus favoritos de forma intuitiva, facilitando a visualização tanto dos itens marcados como favoritos quanto dos usuários que favoreceram um item específico.

3.1.6 Comentários

Na interface de cada item, será disponibilizada uma seção de comentários, incluindo a capacidade de adicionar comentários resposta, que chamaremos de “sub-comentários”. Esta funcionalidade visa proporcionar aos usuários um espaço para realizar anotações pertinentes ou esclarecer dúvidas específicas relacionadas a cada item.

Dessa maneira, torna-se imperativo implementar um sistema de gerenciamento, operando no back-end da aplicação, para administrar os sub-comentários associados a cada comentário. Esta abordagem é essencial para assegurar a ordem, integridade e acessibilidade das interações entre usuários, contribuindo para um ambiente colaborativo e informativo na seção de comentários.

Ao viabilizar o gerenciamento eficiente dos sub-comentários, a plataforma reforça sua utilidade como uma ferramenta de comunicação e compartilhamento de informações, promovendo a interatividade e o engajamento dos usuários na exploração e discussão de conteúdo específico.

3.1.7 Itens acessíveis

Embora a proposta principal da ferramenta seja fornecer uma visão aberta de todos os bancos de dados, é importante ressaltar que existe uma distinção entre a

mera visualização de metadados, origens e aplicações de um item e o efetivo acesso completo a esse item, com permissões de leitura e escrita na íntegra desse repositório.

Isso se deve às políticas internas da companhia, que determinam que cada usuário tenha acesso real (na íntegra) somente aos bancos de dados essenciais para suas atividades diárias. Afinal, o propósito do catálogo é facilitar a descoberta de outros repositórios de dados e, para isso, o usuário não precisa ter acesso aos dados integrais, e o propósito da democratização de dados continua a ser cumprido.

Assim, a ferramenta deve ser capaz de não apenas listar todos os itens disponíveis, mas discriminar quais desses itens são efetivamente acessíveis (na íntegra) por cada usuário. Essa funcionalidade visa atender às políticas internas da empresa, garantindo que a experiência do usuário seja personalizada de acordo com suas reais necessidades e permissões de acesso.

3.1.8 Perfilagem de Dados

Por fim, uma demanda essencial foi apresentada: a implementação de um *endpoint* capaz de disponibilizar a perfilagem de dados para um item selecionado. No entanto, conforme o projeto evoluiu, houve uma revisão desse requisito, transformando-o em uma prova de conceito aplicável apenas a alguns itens específicos, em vez de responder globalmente a todos os itens.

A alteração nesse requisito não foi arbitrária, mas sim uma decisão estratégica tomada adiante no projeto. O próximo capítulo abordará detalhadamente a motivação dessa modificação.

3.2 ESCOLHA DAS TECNOLOGIAS DE CONSTRUÇÃO DA FERRAMENTA

Estudando os requisitos apresentados pela gerência, estudou-se, em time, quais as tecnologias poderiam ser utilizadas na construção da ferramenta. Pelo solicitado, seria necessário alguns bancos de dados, que não precisariam ser necessariamente relacionais, por exemplo:

- Banco de metadados dos itens
- Banco de tags
- Banco de favoritos
- Banco de audiência de acessos

Com base na análise dos bancos de dados disponíveis, concluiu-se que um banco de dados *NoSQL*, com suporte para dados do tipo chave-valor e documentos, seria a escolha mais apropriada para atender ao propósito estabelecido. Entre as opções disponíveis, o MongoDB emergiu como a solução mais adequada, respaldada

pela experiência prévia da equipe e de outros times da companhia. A decisão foi reforçada pela existência de uma biblioteca em Python para comunicação com o banco, acelerando o processo de desenvolvimento. Essa escolha estratégica alinha-se perfeitamente com a natureza da ferramenta, concebida como um MVP (Mínimo Produto Viável, em inglês), visando eficiência e agilidade.

No que tange à automatização de tarefas, embora a empresa tenha adotado uma abordagem de liberdade de escolha, a decisão pelo Apache Airflow foi tomada de forma natural. Isso se deve à profunda expertise consolidada na companhia em relação a essa ferramenta. A familiaridade dos times com o Apache Airflow facilita não apenas a implementação inicial, mas também a manutenção contínua e a sustentabilidade da ferramenta no futuro.

A seleção da FastAPI como comunicador entre o back-end e o front-end foi respaldada por uma vasta experiência acumulada na companhia. Essa escolha se fundamenta, principalmente, na facilidade de desenvolvimento proporcionada pela FastAPI, acelerando todas as fases do processo. Além disso, a decisão é orientada pela experiência de outros times que assumirão a responsabilidade pela sustentação da ferramenta. A integração fácil com bancos MongoDB e a compatibilidade com ambientes de desenvolvimento em Docker também contribuíram para a escolha dessa tecnologia.

Essas decisões estratégicas na seleção de tecnologias refletem não apenas a expertise existente na companhia, mas também uma abordagem pragmática voltada para a eficiência, agilidade e sustentabilidade a longo prazo da ferramenta.

3.3 AMBIENTE DE DESENVOLVIMENTO LOCAL PARA A API

Conforme a padronização da empresa, a implementação da ferramenta necessitou de três ambientes distintos: “dev” (desenvolvimento), “qas” (garantia de qualidade) e “prd” (produção). No entanto, deparou-se com um desafio ao constatar que o desenvolvimento local poderia ser significativamente lento ao utilizar os recursos padrões da FastAPI e do MongoDB. Ambas as tecnologias têm um tempo considerável de implementação, impactando a agilidade do desenvolvimento local.

Para contornar essa questão, optou-se pela tecnologia Docker, que oferece benefícios fundamentais. Em primeiro lugar, proporciona homogeneidade na implementação da ferramenta em ambientes diversos, seja no computador local do desenvolvedor, em nuvem de provedores distintos, ou em outra infraestrutura. Em segundo lugar, viabiliza a utilização de *dev containers*, ambientes locais que permitem a reimplementação do sistema considerando apenas as mudanças realizadas desde a última implementação. Essa abordagem é particularmente eficaz, contanto que não envolva alterações no núcleo da implementação, como inclusão de novas bibliotecas.

Ao adotar *dev containers*, foi possível reduzir significativamente o tempo de

reimplementação, passando de mais de cinco minutos para menos de cinco segundos. Essa redução é crucial para manter a fluidez do trabalho de um desenvolvedor de software, especialmente ao lidar com tecnologias novas que frequentemente exigem tentativa e erro.

4 AUTOMATIZAÇÃO DE PROCESSOS

A ingestão dos metadados do Purview e do Synapse precisariam ser feitas via automatização, dado o volume e demora dessas requisições. Foi necessário, também, automatizar a solicitação dos grupos de acesso de cada página de metadados; o registro do banco de audiências no datalake; e a prova de conceito da perfilagem de dados.

O primeiro passo para poder executar uma Task no Airflow é criar uma imagem para ser utilizada no executor que define o container em que será executada a Task. Para isto então, a partir de um Dockerfile, foi especificado o que se desejava na imagem. No Dockerfile foi configurada a instalação das dependências necessárias para o Airflow, o Spark, o Azure Data Lake Storage Gen2, MongoDB e as dependências que haviam sido definidas pelos desenvolvedores da ferramenta. Com essas definições da imagem seria possível executar os processos em Python no Airflow, pois o ambiente da Task seria similar à máquina local de cada usuário.

E, formadas todas as tasks, elas podem ser unidas e configuradas em uma única DAG, como demonstrado pelas Figuras 2 e 3.

```
60  dag = DAG(
61      dag_id=DAG_ID,
62      default_args=default_args,
63      max_active_runs=1,
64      concurrency=16,
65      schedule_interval="00 06 * * 1-5",
66      doc_md="Extraction of information from Synapse and Purview for DataHub",
67      catchup=False,
68      tags=["DataHub", "Hub"],
69      dagrun_timeout=timedelta(minutes=240),
70  )
```

Figura 2 – Configurações da DAG

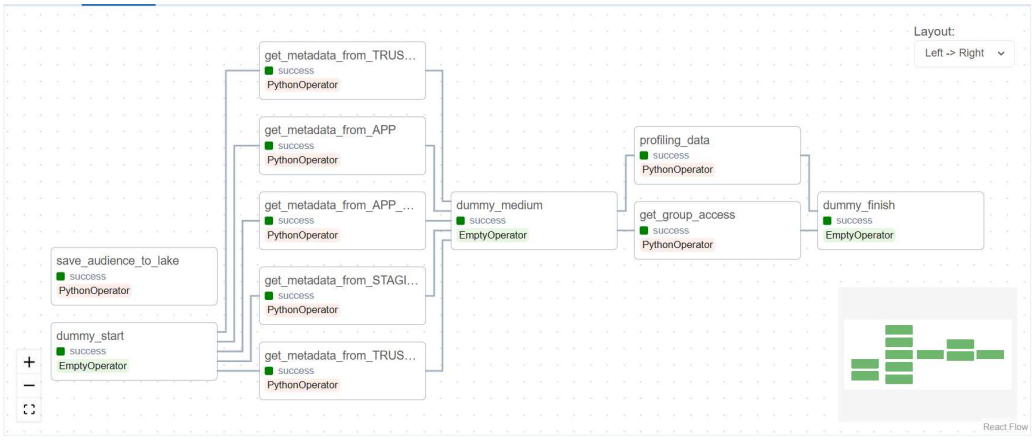


Figura 3 – Recorte do gráfico da DAG do Airflow

4.1 INGESTÃO DOS METADADOS

A obtenção e ingestão dos metadados constituem uma operação de caráter intrincado e abrangente. Esta operação inicia-se com a execução de uma rotina especializada que formula requisições aos serviços do Synapse, visando a obtenção dos metadados pertinentes. Essa requisição, por sua vez, é seguida de um processo de filtragem dos bancos de dados com base em camadas específicas, culminando na elaboração de uma lista de caminhos.

Essa lista de caminhos, cuidadosamente estruturada, torna-se o ponto de partida para uma iteração no Purview. Cada item da lista é submetido a uma busca detalhada, resultando em uma extensa variedade de metadados. Dessa diversidade, uma seleção é realizada, pautada nas diretrizes estipuladas pela gerência do projeto.

Os metadados escolhidos, após filtragem, são então organizados em uma estrutura de dados composta por uma lista de dicionários. Este formato estruturado é concebido para otimizar o armazenamento e manipulação dos metadados. A etapa final desse processo consiste na escrita dessa lista de dicionários no MongoDB, onde os metadados adquiridos são armazenados de forma organizada e acessível para consultas futuras.

Este procedimento envolvendo múltiplas etapas e tecnologias, visa assegurar a precisão e eficiência na gestão dos metadados, permitindo uma integração fluida e uma recuperação eficaz dessas informações essenciais ao projeto.

4.2 GRUPOS DE ACESSO

Para a determinação dos acessos concedidos a diferentes usuários em relação aos bancos de dados, uma abordagem foi adotada, demandando a implementação de uma rotina de requisições direcionadas aos sistemas da Microsoft. O propósito primordial dessas requisições consistia na obtenção de informações relativas aos “grupos de acesso” com permissões específicas para acessar distintos caminhos dentro do sistema.

O retorno fornecido pela Microsoft foi formatado de acordo com um paradigma chave-valor, onde cada “caminho” estava associado a um conjunto correspondente de “grupos de acesso”. Essa estrutura de dados, ao ser recebida, serviu como base para a criação de dois bancos de dados distintos no MongoDB. Um desses bancos refletia a estrutura original, preservando a relação de chave-valor originária da resposta da Microsoft. O segundo banco, por sua vez, apresentava uma inversão dessa relação, associando os “grupos de acesso” aos “caminhos acessíveis”.

A decisão de duplicar os bancos de dados não foi estritamente necessária, todavia, foi deliberadamente tomada para otimizar o tempo de processamento das consultas à API no MongoDB. Essa abordagem visa garantir uma uniformidade no

desempenho, permitindo que consultas referentes a grupos de acesso e caminhos mantenham tempos de processamento semelhantes. Importante ressaltar que tal otimização não se dá sem ônus, sendo acompanhada de um leve aumento no custo computacional associado à execução desta tarefa.

4.3 REGISTRO DA AUDIÊNCIA NO DATALAKE

Essa tarefa era de certa forma bem simples, lendo os dados contidos no MongoDB, a tarefa prosseguia em salvar esses mesmos dados no DataLake, para que pudessem ser acessados independente da ferramenta aqui desenvolvida.

4.4 PERFILAGEM DE DADOS

A fase de construção desta tarefa revelou uma considerável complexidade ao perceber que o tempo de processamento necessário para a perfilagem de dados excedia significativamente as estimativas iniciais. Em resposta a essa constatação, em colaboração com a gerência, deliberou-se pela implementação da perfilagem de dados em uma parcela limitada dos caminhos possíveis.

A opção estratégica recaiu sobre a integração desta rotina no Apache Airflow, configurado como um protótipo representativo do comportamento do hub no cenário em que a perfilagem estivesse disponível para todos os caminhos. Durante os testes, alguns itens apresentaram um tempo de processamento que ultrapassou os 20 minutos, atribuível à considerável quantidade de registros e campos envolvidos.

Diante desse cenário, desencadearam-se discussões com a gerência para avaliar a viabilidade da feature em termos de custo, considerando a possibilidade de ampliar as capacidades de processamento para otimizar a tarefa. Como alternativa, surgiu a proposta de reduzir a abrangência da perfilagem, transformando-a em uma prova de conceito aplicável a apenas alguns itens. Esta alternativa se revelou estratégica para garantir que, durante a apresentação da ferramenta aos stakeholders, ainda fosse possível exibir uma representação funcional da capacidade de perfilagem, mesmo que em uma escala reduzida. A decisão final alinhou-se à implementação da prova de conceito em alguns itens, refletindo uma abordagem pragmática visando a demonstração eficaz da funcionalidade.

Este desdobramento na estratégia de perfilagem de dados destaca a agilidade na identificação e resposta às complexidades inerentes ao desenvolvimento, evidenciando um processo de tomada de decisão flexível que visa otimizar a eficiência operacional e fornecer funcionalidades valiosas aos usuários finais.

5 EXECUÇÃO EM TEMPO REAL DE PROCESSOS

A execução em tempo real trata, majoritariamente, da API, seus *endpoints* e suas rotinas, de acordo com os requisitos estipulados anteriormente, os *endpoints* desenvolvidos foram:

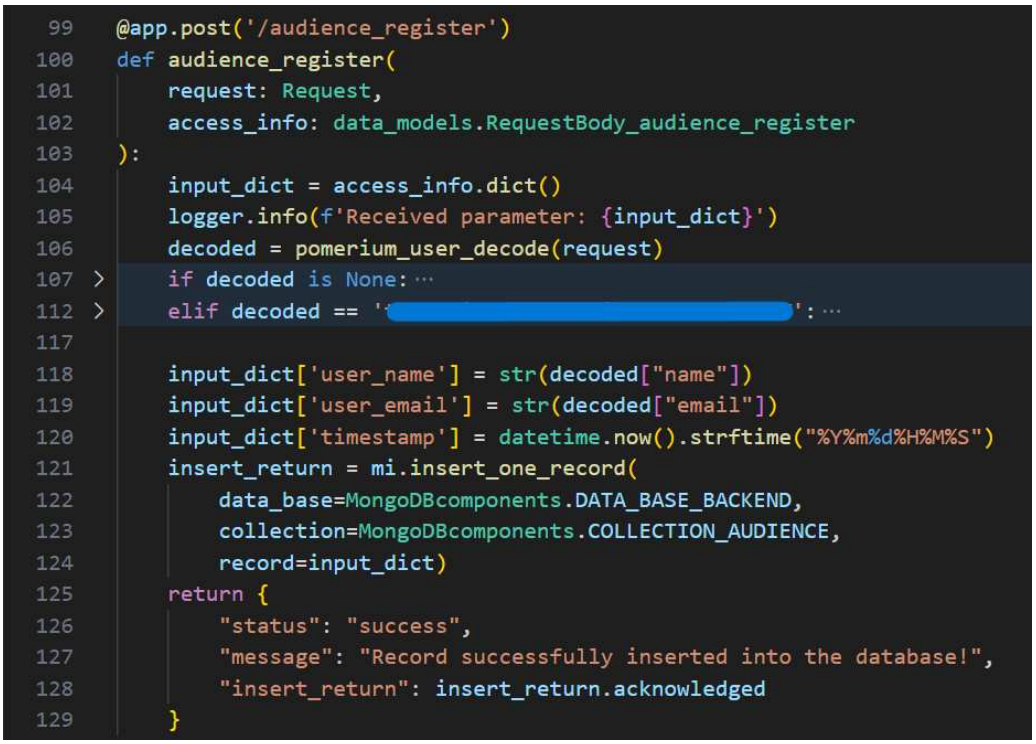
- /audience_register
- /get_all_audience_register
- /tag_insert
- /tag_update
- /get_tags_from_db
- /tag_add_relation_metadata
- /tag_remove_relation_metadata
- /get_metadata_from_db
- /get_filtered_metadata_from_db
- /add_table_to_favorites
- /remove_table_from_favorites
- /get_user_favorites
- /get_favorite_table_info
- /insert_comment
- /insert_SUBcomment
- /get_accessible_paths
- /get_info_from_purview

5.1 *ENDPOINTS* DE AUDIÊNCIA

As rotas `/audience_register` e `/get_all_audience_register` desempenham um papel crucial no manuseio dos registros de audiência da ferramenta.

A rota `/audience_register`, quando invocada, requer dois parâmetros: as informações de acesso, incluindo a página que está sendo acessada, o horário, entre outros; e as informações do usuário. Em seguida, a partir das informações do usuário, são extraídos o nome e o email, os quais são adicionados a um dicionário contendo

as informações de acesso. Por fim, esse dicionário é registrado no MongoDB, mais especificamente na coleção designada para armazenar dados relacionados à audiência. Seu diagrama pode ser conferido em 5.



```
99   @app.post('/audience_register')
100  def audience_register(
101      request: Request,
102      access_info: data_models.RequestBody_audience_register
103  ):
104      input_dict = access_info.dict()
105      logger.info(f'Received parameter: {input_dict}')
106      decoded = pomerium_user_decode(request)
107  >   if decoded is None:···
112  >   elif decoded == '[redacted]':···
117
118      input_dict['user_name'] = str(decoded["name"])
119      input_dict['user_email'] = str(decoded["email"])
120      input_dict['timestamp'] = datetime.now().strftime("%Y%m%d%H%M%S")
121      insert_return = mi.insert_one_record(
122          data_base=MongoDBcomponents.DATA_BASE_BACKEND,
123          collection=MongoDBcomponents.COLLECTION_AUDIENCE,
124          record=input_dict)
125      return {
126          "status": "success",
127          "message": "Record successfully inserted into the database!",
128          "insert_return": insert_return.acknowledged
129      }
```

Figura 4 – Código da rota `/audience_register`

A rota `/get_all_audience_register` realiza uma consulta ao MongoDB e retorna, essencialmente, a coleção completa de registros de audiência. A resposta é formatada como uma lista de dicionários em formato de texto na requisição realizada.

5.2 *ENDPOINTS* DE TAGS

As rotas `/tag_insert`, `/tag_update`, `/get_tags_from_db`, `/tag_add_relation_metadata` e `/tag_remove_relation_metadata` são responsáveis pela gestão de todas as tags da ferramenta.

A rota `/tag_insert` insere uma nova tag na coleção de tags do MongoDB, recebendo como parâmetros um conjunto de informações, como Nome, Logo, Descrição, Tipo, entre outros dados relativos à classe dessa tag.

A rota `/tag_update` é responsável por atualizar uma determinada tag internamente à coleção de tags. Ela recebe como parâmetros os mesmos dados da `/tag_insert`, sendo que todos os campos (exceto o nome de referência) são opcionais. Isso significa que apenas os campos informados serão atualizados.

A rota `/get_tags_from_db` retorna a coleção completa das tags, em formato de texto, via API.

Figura 5 – Diagrama da rota `/audience_register`

Por fim, as rotas `/tag_add_relation_metadata` e `/tag_remove_relation_metadata` são responsáveis por criar ou remover as relações entre as tags e os metadados vindos do Purview. Ambas recebem como parâmetros somente o ID da tag e o ID do caminho do Purview a serem relacionados.

5.3 *ENDPOINTS* DE METADADOS DO PURVIEW

A rota `/get_metadata_from_db` retorna a coleção completa de metadados, dispensando a necessidade de parâmetros na requisição.

Por sua vez, a rota `/get_filtered_metadata_from_db` proporciona uma funcio-

```python
@app.post('/tag_add_relation_metadata')
def tag_add_relation_metadata(
    tag_info: data_models.RequestBody_tag_relation_metadata
):
    input_dict = tag_info.dict()
    logger.info(f'Received tag_info: {input_dict}')
    insert_return = mi.update_array_in_document(
        data_base=MongoDBcomponents.DATA_BASE_BACKEND,
        collection=MongoDBcomponents.COLLECTION_PURVIEW,
        column_to_add='tags',
        elements_to_add=input_dict['tags_id'],
        filter_condition={'id': input_dict['table_id']})
    db = mi.client[MongoDBcomponents.DATA_BASE_BACKEND]
    consulta = db[MongoDBcomponents.COLLECTION_PURVIEW].find(
        {'id': input_dict['table_id']}, {"_id":0})

    return pymongo_cursor_to_dictionary(consulta, 'id')
```

Figura 6 – Código da rota `/tag_add_relation_metadata`

```python
@app.post("/get_filtered_metadata_from_db")
def get_filtered_metadata_from_db(
    filter_list: List[data_models.RequestBody_get_metadata]
):
    input_filter_list = [i.dict() for i in filter_list]
    print('filter_list:',type(input_filter_list),input_filter_list)
    db = mi.client[MongoDBcomponents.DATA_BASE_BACKEND]
    consulta = db[MongoDBcomponents.COLLECTION_PURVIEW].find(
        {'$or': [*input_filter_list]},
        {"_id":0}
    )
    logger.info('Query on METADATA collection successfully executed')
    return pymongo_cursor_to_dictionary(consulta, 'id')
```

Figura 7 – Código da rota `/get_filtered_metadata_from_db`

nalidade mais específica. Ao ser acionada, ela retorna uma fração reduzida da coleção de metadados. Essa redução é determinada pelo parâmetro informado na requisição, indicando a camada pela qual os metadados devem ser filtrados para exibição.

5.4 *ENDPOINTS* DE FAVORITOS

Para gerenciar os favoritos, foram implementadas as rotas `/add_table_to_favorites`, `/remove_table_from_favorites`, `/get_user_favorites` e `/get_favorite_table_info`.

A rota `/add_table_to_favorites` e `/remove_table_from_favorites` têm a finalidade de adicionar ou remover, respectivamente, um caminho do Purview dos favoritos do usuário que realiza a requisição. Como parâmetros, essas rotas aceitam apenas as informações referentes a esses caminhos.

Já a rota `/get_user_favorites` retorna todos os itens que foram marcados

como favoritos pelo usuário que invocou esse *endpoint*.

Por fim, a rota `/get_favorite_table_info` tem como propósito retornar informações sobre quem marcou como favorito um determinado caminho do Purview, o qual é especificado como parâmetro.

5.5 *ENDPOINTS* DE COMENTÁRIOS

A rota `/insert_comment` é responsável por adicionar comentários às páginas de caminhos do Purview. Seus parâmetros incluem o caminho do Purview no qual o comentário será inserido e o texto do comentário.

Por outro lado, a rota `/insert_SUBcomment` tem a função de inserir sub-comentários. Além dos parâmetros de caminho do Purview e texto do sub-comentário, essa rota exige um parâmetro adicional: a referência ao "comentário pai". Essa referência é uma concatenação do email do usuário do comentário de referência com o carimbo da data e hora da criação desse comentário.

Essas rotas fornecem uma maneira robusta de interação e discussão nas páginas do Purview, permitindo a adição de comentários e sub-comentários de forma estruturada.

5.6 *ENDPOINT* DE CAMINHOS ACESSÍVEIS

Por outro lado, a rota `/get_accessible_paths` desempenha uma função crucial ao retornar todos os caminhos aos quais o usuário que a invocou tem acesso. Essa funcionalidade é realizada por meio de um cruzamento entre os grupos nos quais o usuário está inserido e a coleção de dados referentes aos grupos de acesso especificada por caminho do Purview.

Essa rota proporciona uma visão detalhada dos caminhos disponíveis para o usuário, estabelecendo uma relação entre a sua associação a grupos específicos e as permissões associadas a esses grupos sobre os caminhos do Purview.

5.7 *ENDPOINTS* DE INFORMAÇÕES DO PURVIEW

Finalmente, a rota `/get_info_from_purview` desempenha um papel crucial ao retornar metadados adicionais do Purview com base no caminho fornecido. A decisão de manter esta rota foi tomada durante a fase de requisitos, quando ainda não havia certeza sobre quais metadados seriam necessários disponibilizar na coleção de metadados do Purview.

A escolha estratégica de manter uma rota que permitisse acessar todos os metadados foi adotada para viabilizar a obtenção de informações futuras que o projeto poderia demandar. Isso foi feito de maneira a assegurar que o front-end pudesse filtrar

e utilizar todas as informações relevantes, proporcionando uma abordagem flexível para a evolução do sistema em resposta a requisitos desconhecidos durante a fase inicial do desenvolvimento.

Esse endpoint torna-se, assim, uma peça fundamental para a robustez e adaptabilidade do sistema, garantindo a capacidade de obtenção de dados adicionais do Purview conforme as necessidades evoluem ao longo do tempo.

5.8 *ENDPOINTS* DE DEBUG

Adicionalmente, durante o processo de desenvolvimento, foram implementados alguns endpoints com propósitos de debugging. Ao término do projeto, a decisão foi tomada de mantê-los ativos, visto que a equipe reconheceu a utilidade desses endpoints para a fase de sustentação da ferramenta. Estes endpoints específicos são:

- /last_startup_time
- /check_mongodb
- /list_mongodb_collections
- /pomerium_user_decode

A rota `/last_startup_time` representa um endpoint simples cuja função é recuperar da memória o horário do último reinício da API. Essa rota desempenha um papel crucial durante atividades de debugging, oferecendo uma maneira rápida de verificar se a API foi reiniciada conforme necessário, por exemplo.

Figura 8 – Exemplo de uso da rota `/last_startup_time`

Em adição, as rotas `/check_mongodb` e `/list_mongodb_collections` foram implementadas com o propósito de proporcionar uma visualização abrangente de todas as coleções presentes no MongoDB. A primeira permite verificar uma coleção específica do MongoDB, enquanto a segunda oferece uma listagem das coleções existentes, facilitando a análise e monitoramento do banco de dados.

Por fim, a rota `/pomerium_user_decode` executa a função de decodificação responsável por identificar o usuário que está acessando o frontend através do Pomerium.

Figura 9 – Exemplo de uso da rota `/pomerium_user_decode`

A inclusão destes endpoints específicos para debugging reflete a preocupação da equipe em garantir uma operação suave e eficaz da API, proporcionando ferramentas essenciais para a identificação de problemas e manutenção contínua da ferramenta.

6 INTEGRAÇÃO COM FRONT-END E RESULTADOS

Durante o desenvolvimento do back-end, o time responsável pelo front-end acompanhou boa parte dos avanços, afinal, a construção de uma ferramenta com front-end e back-end desenvolvidos por times diferentes necessita, de fato, uma comunicação constante e eficaz.

Durante todo projeto, portanto, houveram muitas iterações em como abordar cada problema, com o resultado saindo depois de várias discussões saudáveis entre os dois times, que acabavam encontrando a melhor solução, do ponto de vista da ferramenta.

Logo, a integração com o front-end foi um processo contínuo e fluido. Findaas as tarefas do back-end, o backlog do front-end era muito reduzido. Entretanto, durante a confecção desse trabalho, a seção de comentários ainda não havia sido desenvolvida, assim como já havia sido implementado a interface para outras funcionalidades futuras que não haviam sido requisitadas a construção em back-end. Esses pontos ficam claros nas imagens de demonstração do produto.

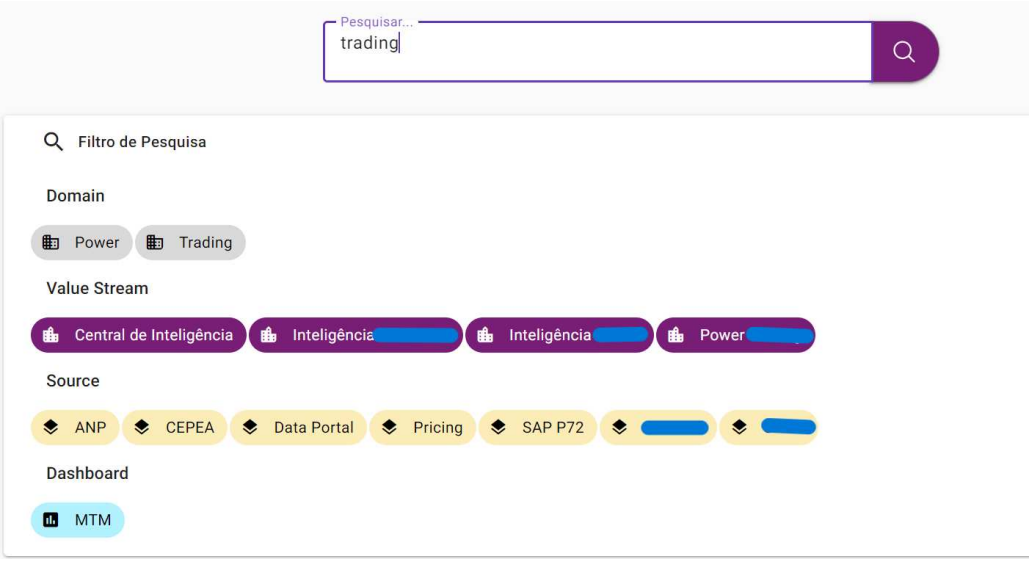


Figura 10 – Exemplo de pesquisa realizada em frontend

Na Figura 10 é possível visualizar uma pesquisa realizada na plataforma pelo termo “trading” e, abaixo, percebe-se opções de filtro. Esses filtros são frutos das “tags” inseridas no back-end que classificam cada repositório de dados.

Já na Figura 11, pode-se notar um caso mais básico de repositório (sem descrição, por exemplo), que foi conferido com os grupos de acesso do autor e a ferramenta reporta que o autor tem acesso real ao repositório, além disso, a ferramenta apresenta usos básicos desses dados, como os caminhos no Data Lake e no Synapse, além de um exemplo de query SQL a ser utilizada no Synapse.

Na Figura 12 nota-se uma descrição mais completa, como por exemplo, o uso

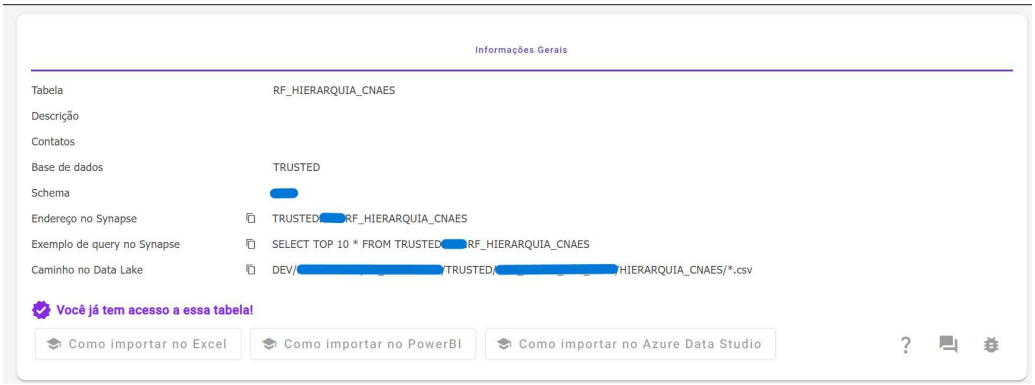


Figura 11 – Demonstração básica de repositório

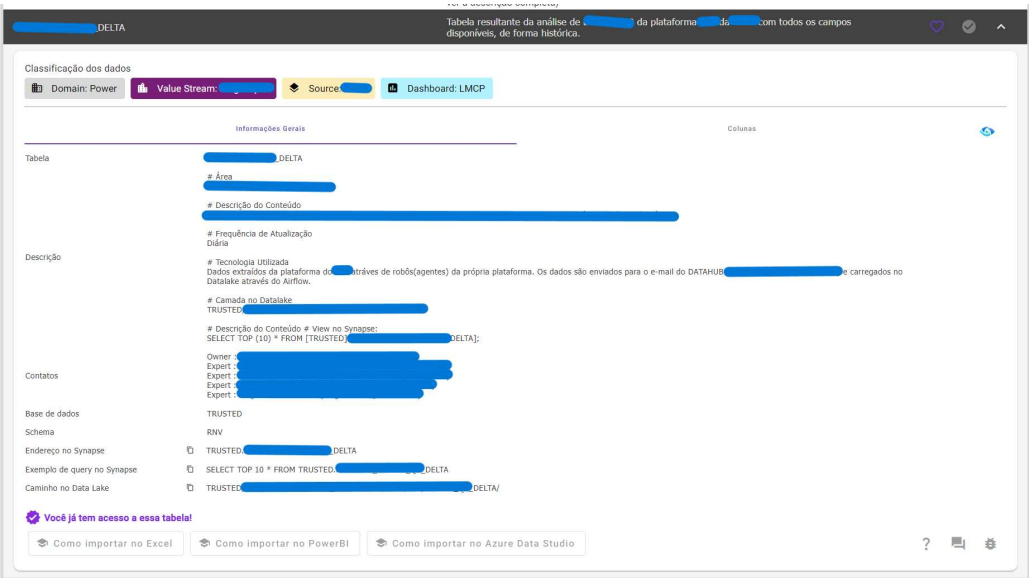


Figura 12 – Demonstração elaborada de repositório - informações gerais

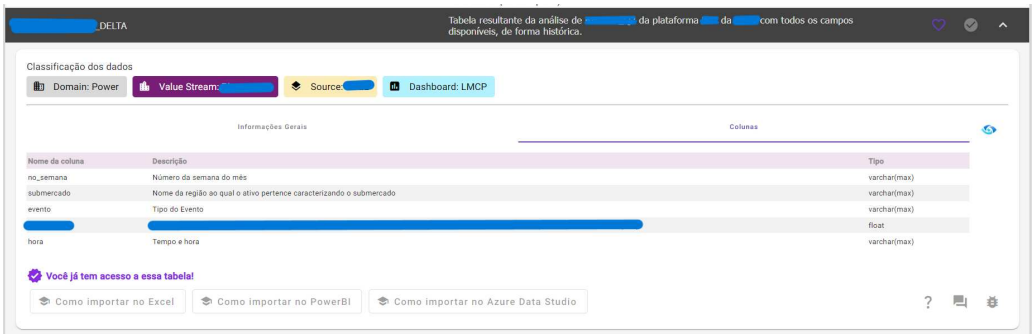


Figura 13 – Demonstração elaborada de repositório - colunas

das “tags”, além de uma descrição textual da origem dos dados desse repositório, a que área pertence, qual a frequência de atualização, como são coletados, além de quem são os funcionários responsáveis por esses dados dentro da companhia. Além disso, na Figura 13, apresenta-se quais campos esse repositório tem, com uma breve descrição de cada um deles.

Ademais, na Figura 14, pode-se notar como são dispostos os repositórios favo-

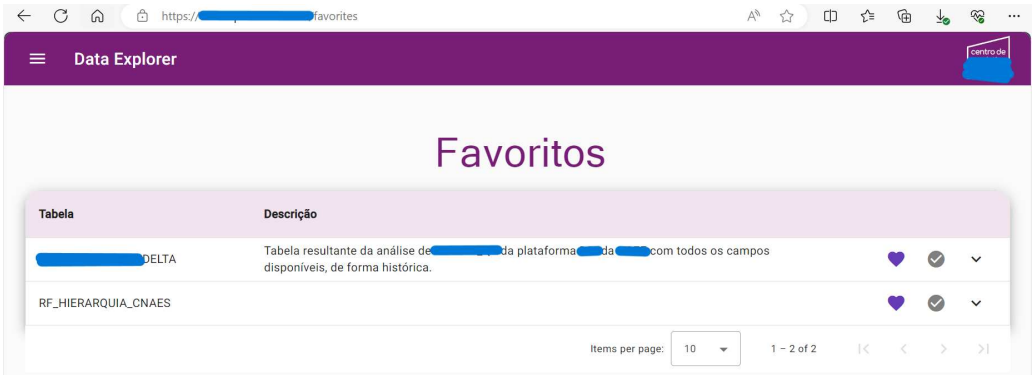


Figura 14 – Exemplo de filtro dos favoritos

ritados pelo usuário.

Vale notar, também, como apresentado na Figura 15 que os tempos de execução das tarefas no Airflow são, à primeira vista, longos. Mas ao se analisar que essas tarefas lidam com metadados de diversas fontes, de vários formatos, é compreensível que leve esse tempo. As camadas “APPLICATION” e “STAGING” que são as que mais possuem repositórios, são as mais demoradas, levando cerca de 30 minutos cada (em paralelo). Já a tarefa de perfilagem de dados é a mais lenta, levando mais de hora (para uma amostra de poucos repositórios), mas não tem efeito sobre a disponibilidade dos metadados das tarefas anteriores, então entendeu-se que não havia problema maior com esse tempo estendido.

Figura 15 – Tempos de execução das tarefas no Airflow

Por fim, os resultados obtidos pelo back-end do catálogo de dados foram completamente satisfatórios. O sistema atendeu a todos os chamados e requisições feitos pelo front-end, com tempos de resposta dentro do esperado para um back-end construído em 2023. Com isso, foi possível, então, implementar o catálogo de dados e disponibilizar para toda companhia a ferramenta de forma a efetivar a democratização de dados que o sistema almejava. Desse modo, houve uma evolução no modo que a companhia percebe os próprios repositórios e como pode utilizá-los com a implementação dessa nova ferramenta, que é tendência nos setores que lidam com dados em todas empresas com grande volume de dados.

7 CONCLUSÃO

Os requisitos foram cumpridos. O único que não foi possível cumprir, a perfilagem de dados, foi bem explicado e compreendido em sua motivação, além de acatadas as sugestões dadas.

Os resultados foram satisfatórios visto que a companhia aprovou a prova de conceito antes do prazo final de entrega, pois já entendia a importância da ferramenta, e a empregou em produção e divulgou para todos os times de software também antes do final do projeto.

A construção de back-end de hub de análise de metadados sobre repositórios de dados foi um grande desafio pessoal e profissional, foi uma tarefa difícil que o autor pôde cumprir com confiança no trabalho que desenvolveu e no amparo de colegas profissionais e de amigos e familiares.

Com certeza a experiência do autor em projetos passados, além da vasta experiência dos colegas de time, foram fundamentais para uma boa fluidez do desenvolvimento.

REFERÊNCIAS

APACHE. **What is Airflow™?** [*S.l.*]. Disponível em: https://airflow.apache.org/docs/apache-airflow/stable/index.html. Acesso em: 26 jan. 2024.

DOCKER. **Docker Overview**. [*S.l.*]. Disponível em: https://docs.docker.com/get-started/overview/. Acesso em: 26 jan. 2024.

MICROSOFT. **Azure Synapse Analytics**. [*S.l.*]. Disponível em: https://azure.microsoft.com/en-us/products/synapse-analytics. Acesso em: 26 jan. 2024.

MICROSOFT. **Ferramentas de democratização de dados**. [*S.l.*]. Disponível em: https://learn.microsoft.com/pt-br/azure/cloud-adoption-framework/innovate/innovation-guide/data?tabs=Catalog. Acesso em: 26 jan. 2024.

MICROSOFT. **Learn about Microsoft Purview**. [*S.l.*]. Disponível em: https://learn.microsoft.com/en-us/purview/purview. Acesso em: 26 jan. 2024.

MICROSOFT. **Microsoft Entra ID (antigo Azure Active Directory)**. [*S.l.*]. Disponível em: https://www.microsoft.com/pt-br/security/business/identity-access/microsoft-entra-id. Acesso em: 26 jan. 2024.

MICROSOFT. **O que é o Azure?** [*S.l.*]. Disponível em: https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-azure/. Acesso em: 26 jan. 2024.

OLESEN-BAGNEUX, Ole. **O Catálogo de Dados Corporativo**. 1. ed. [*S.l.*]: O'Reilly, 2023.

POMERIUM. **About Pomerium**. [*S.l.*]. Disponível em: https://www.pomerium.com/about/. Acesso em: 26 jan. 2024.

RADIX. **Sobre Nós**. [*S.l.*]. Disponível em: https://www.radixeng.com.br/a-radix/sobre-nos. Acesso em: 26 jan. 2024.

RAMÍREZ, Sebastián. **FastAPI**. [*S.l.*]. Disponível em: `https://fastapi.tiangolo.com`. Acesso em: 26 jan. 2024.

SUTHERLAND, Jeff. **Scrum: a arte de fazer o dobro do trabalho na metade do tempo**. [*S.l.*]: Sextamte, 2014.