Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Trabalho de Conclusão de Curso – Artigo

Implementação de uma ferramenta de análise de testes automatizados para aumento de produtividade em um ambiente de desenvolvimento

Matheus Medeiros Campos[1]
Centro Universitário Academia, Juiz de Fora, MG

Evaldo de Oliveira da Silva[2]
Centro Universitário Academia, Juiz de Fora, MG

Linha de Pesquisa: Engenharia de Software

RESUMO

No cenário atual de desenvolvimento de *software*, o teste de *software* desempenha um papel crucial na busca pela qualidade dos produtos. A automação de testes tem se mostrado uma solução eficiente para esse desafio, e a integração contínua com ferramentas como o *Cypress* tem possibilitado alcançar métricas de produtividade e eficiência. Neste contexto, o presente trabalho tem como objetivo explorar a geração de métricas a partir da ferramenta *Cypress*, que é amplamente utilizada para a criação de testes funcionais. Por meio da extração e análise de dados provenientes do *Cypress*, é possível obter indicadores valiosos para a avaliação da qualidade do *software* em diferentes projetos. A utilização de *dashboards* permite a visualização e organização dessas métricas de forma clara e intuitiva. Além disso, a integração com outras ferramentas, como o Jenkins, que foi selecionado por ser gratuito e de fácil integração com o processo e controle de builds, amplia as possibilidades de análise e monitoramento do processo de desenvolvimento. Neste trabalho, propomos um processo de extração, transformação e carga de dados gerados pelo *Cypress*, visando a geração de indicadores em *dashboards*. Os *dashboards* oferecem aos profissionais de TI uma visão abrangente e consolidada de vários projetos simultaneamente, facilitando a identificação de padrões, tendências e possíveis problemas. Ao abordar a utilização de dados e métricas extraídas das atividades de teste de *software*, este trabalho busca viabilizar a prática da integração contínua, que é fundamental para agilizar e aprimorar o processo de desenvolvimento. Por meio de um estudo de caso prático, demonstra-se como essa abordagem pode beneficiar equipes de teste, proporcionando uma visão mais ampla e embasada para a tomada de decisões e o aprimoramento contínuo dos testes de *software*. Com base nesse contexto, este trabalho contribui para a compreensão e aplicação de métricas de teste de *software*, visando melhorar a qualidade do produto final e otimizar o trabalho das equipes de TI.

Palavras-chave: Teste *end-to-end*, Teste de software, *Cypress*.

ABSTRACT

[1] Discente do Curso de Engenharia de Software do Centro Universitário Academia – UniAcademia. Endereço: Rua Olegário Maciel 297 ap 1001 - Centro. Celular: (21) 97174-0440. E-mail: matheusmedeiroscampos@gmail.com
[2] Docente do Curso de Engenharia de Software do Centro Universitário Academia. Orientador.

In the current software development landscape, software testing plays a crucial role in pursuing product quality. Test automation has proven to be an efficient solution for this challenge, and continuous integration with tools like Cypress has made it possible to achieve productivity and efficiency metrics. In this context, this study aims to explore the generation of metrics using the Cypress tool, widely used for creating functional tests. By extracting and analyzing data from Cypress, valuable indicators can be obtained to assess software quality across different projects. Dashboards provide a clear and intuitive visualization and organization of these metrics. Furthermore, integration with other tools such as Jenkins, which was chosen for its free availability and ease of integration with the build process and version control, expands the possibilities for analysis and monitoring of the development process. In this study, we propose a process of data extraction, transformation, and loading generated by Cypress, aiming to generate indicators in dashboards. Dashboards offer IT professionals a comprehensive and consolidated view of multiple projects simultaneously, facilitating the identification of patterns, trends, and potential issues. By addressing the use of data and metrics extracted from software testing activities, this study seeks to enable the practice of continuous integration, which is fundamental to streamline and enhance the development process. Through a practical case study, we will demonstrate how this approach can benefit testing teams by providing a broader and well-founded view for decision-making and continuous improvement of software testing. Based on this context, this study contributes to the understanding and application of software testing metrics, aiming to improve the quality of the final product and optimize the work of IT teams.
Key-words: *End-to-end testing, Software test, Cypress*

1 INTRODUÇÃO

No contexto do desenvolvimento de *software*, os testes desempenham um papel crucial na garantia da qualidade e confiabilidade dos sistemas. Segundo Pressman (2014), eles permitem verificar se as funcionalidades do *software* estão de acordo com os requisitos estabelecidos, identificar possíveis falhas e garantir a estabilidade e segurança do sistema em produção. Os testes de *software* são fundamentais para a entrega de um produto de qualidade aos usuários finais. De acordo com Beizer (1990), eles são responsáveis por validar se o *software* executa corretamente as tarefas para as quais foi projetado, verificando se as entradas fornecidas produzem as saídas esperadas. Eles contribuem para a detecção precoce de erros e falhas, reduzem os riscos associados a problemas não identificados e garantem que o *software* atenda às expectativas e necessidades dos usuários. Além disso, os testes auxiliam na validação das funcionalidades, na identificação de comportamentos inesperados e no aprimoramento contínuo do sistema, permitindo a correção de possíveis defeitos e a otimização do desempenho, conforme apontado por Myers et al. (2011).

No entanto, em ambientes com diferentes projetos de *software* em andamento, a visualização e análise dos resultados dos testes podem se tornar um desafio. Diante desta dificuldade em visualizar e analisar os resultados dos testes de diferentes projetos de forma centralizada e unificada. A ausência de uma solução que facilite essa visualização pode levar a

atrasos na identificação de problemas, dificuldades na priorização de ações corretivas e menor eficiência no processo de desenvolvimento de *software*. De acordo com Myers et al. (2011), a gestão e monitoramento eficaz dos testes em ambientes complexos podem ser facilitados pela utilização de ferramentas de gerenciamento de testes, que oferecem recursos para centralizar e visualizar os resultados dos testes em diferentes projetos.

Tendo em vista este problema foi pensada a criação de uma forma de visualização dos resultados dos testes de diferentes projetos de *software* em um único painel de visualização é de grande relevância.

A criação de uma forma de visualização dos resultados dos testes de diferentes projetos de *software* em um único painel de visualização é de grande relevância. Uma referência relevante sobre a importância da visualização dos resultados dos testes de *software* é o artigo "*Visualizing Software Test Results*" (Khan, A. M., 2016) aborda a importância da visualização dos resultados dos testes como uma maneira eficaz de entender a qualidade do *software* e tomar decisões informadas.

Essa abordagem destaca a importância de se ter uma forma de visualização consolidada e integrada dos resultados dos testes, facilitando a identificação de tendências, padrões e problemas comuns entre os projetos, permitindo uma análise mais abrangente e facilitando a identificação de problemas e a tomada de decisões estratégicas.

O que levou ao estudo deste trabalho foi a necessidade de facilitar a visualização dos resultados dos testes de diferentes projetos de *software*. Para isso foi escolhida a utilização do *framework Cypress*[3] para a automação dos testes de *software*, devido à sua capacidade de criar testes *end-to-end* eficientes e complexos além dos seus recursos avançados que permitem a automação de testes em aplicações web (CYPRESS DOCUMENTATION, 2023).

Para o armazenamento dos resultados desses testes em um formato de arquivo de texto (txt) foi desenvolvido um *script* utilizando o próprio *Cypress*, essa abordagem permite a captura dos dados dos testes de forma estruturada e reutilizável conforme apontado FEWSTER & GRAHAM (2012). Por meio do Pentaho[4], uma ferramenta de *Business Intelligence* (BI) de código aberto que oferece um conjunto abrangente de ferramentas e recursos para auxiliar na análise, visualização e transformação de dados. Ele permite a integração de diferentes fontes de dados, como bancos de dados, arquivos, serviços da web e aplicativos, facilitando a

[3] Cypress. Disponível em: <https://www.cypress.io/>. Acessado em 2 de março de 2023.
[4] Pentaho. Disponível em: <https://www.hitachivantara.com/en-us/products/pentaho-platform/data-integration-analytics.html/>. Acessado em 3 de junho de 2023.

extração, transformação e carga (ETL) de dados para uso em análises e relatórios (PENTAHO DOCUMENTATION, 2023).

No caso deste projeto houve a possibilidades para a solução que permitiu a conversão desses dados, para o formato CSV (*Comma-Separated Values*), um formato de arquivo amplamente usado para dados que se inserem em tabelas, os valores em um arquivo CSV são separados por vírgulas, permitindo organizar e estruturar os dados de forma simples. Com o uso do Jenkins[5], uma ferramenta de automação de integração contínua extremamente utilizada por empresas de desenvolvimento de software, esses resultados poderão ser atualizados continuamente em um banco de dados PostgreSQL[6], que está conectado ao *Metabase*[7], uma plataforma de *BI* capaz de se conectar a várias fontes de dados, como bancos de dados SQL, serviços da web e arquivos CSV, para extrair informações relevantes para seu manejo e montagem de gráficos. Isso possibilitará a exibição dos resultados dos testes em um *dashboard* personalizado, criado em um painel de visualização (METABASE DOCUMENTATION, 2023).

A estrutura deste trabalho está organizada da seguinte maneira: inicialmente, serão apresentados os conceitos fundamentais relacionados ao teste de *software*, ao *framework Cypress* e à plataforma *Metabase*. Em seguida, serão discutidas as abordagens existentes para a integração entre essas ferramentas, bem como os desafios e benefícios envolvidos nesse processo. Na sequência, serão detalhadas as etapas do desenvolvimento e implementação do teste de *software* com *Cypress* e a integração até chegar no *Metabase*. Por fim, serão apresentadas as conclusões e considerações finais, destacando os resultados obtidos e as possíveis contribuições para a área de teste de *software*.

Com essa estrutura, busca-se contribuir para a melhoria dos processos de teste de *software*, oferecendo uma abordagem inovadora e integrada que permita a análise eficaz dos resultados dos testes e a tomada de decisões embasadas em dados concretos.

2 REFERENCIAL TEÓRICO

Para o desenvolvimento do projeto em questão é importante destacar os seguintes pontos.

[5] Jenkins. Disponível em: <https://www.jenkins.io/>. Acessado em 2 de março de 2023.
[6] PostgreSQL. Disponível em: <https://www.postgresql.org/>. Acessado em 2 de março de 2023.
[7] Metabase. Disponível em: <https://www.metabase.com/>. Acessado em 2 de março de 2023.

2.1 Teste de *Software*

O teste de *software* é uma atividade crucial para verificar se o sistema atende aos requisitos estabelecidos, identificar falhas e garantir sua qualidade (Pressman, 2014). Existem vários tipos de teste, como teste de unidade, teste de integração, teste de sistema, teste de aceitação e teste de regressão, cada um com sua finalidade específica (Myers et al., 2011).

Neste trabalho, optou-se pelo teste *end-to-end*, que verifica o funcionamento do sistema como um todo, reproduzindo cenários reais de uso e garantindo a integração adequada (Bertolino, 2007). Esse tipo de teste avalia a usabilidade, eficiência e confiabilidade do sistema, verificando se todas as funcionalidades operam conforme o esperado (Fewster & Graham, 2012).

No entanto, o teste *end-to-end* apresenta desafios, como a complexidade dos cenários de teste, dependência de sistemas externos e necessidade de dados reais (Pressman, 2014). É essencial planejar os testes adequadamente, identificar os pontos críticos do sistema e garantir uma cobertura adequada para uma validação completa.

2.2 *Cypress*

O *framework Cypress* é uma solução de *software* de automação de testes voltada para o desenvolvimento de testes *end-to-end* em aplicações web. Ele se destaca por sua arquitetura moderna, facilidade de uso e recursos avançados que permitem a criação de testes robustos e eficientes, ele é usado principalmente pela sua automação de testes, processo de executar testes de forma automatizada, por meio do uso de ferramentas e *script*s. A automação permite a repetibilidade dos testes, economiza tempo e recursos, e ajuda a identificar falhas de forma mais eficiente. O *Cypress* é uma ferramenta amplamente utilizada para automação de testes em aplicações web (CYPRESS DOCUMENTATION, 2023).

A *Cypress* possui uma arquitetura única que o diferencia de outros *frameworks* de automação de testes. Ele é executado diretamente no navegador, controlando-o de forma direta, o que proporciona uma experiência de teste mais precisa e confiável. A arquitetura do *Cypress* é baseada em dois componentes principais: *o Cypress Test Runner e o Cypress Server* (CYPRESS DOCUMENTATION, 2023).

A *Cypress* suporta a abordagem de desenvolvimento orientado a comportamento (BDD - *Behavior-Driven Development*), que enfatiza a colaboração entre desenvolvedores, testadores e *stakeholders* para criar testes compreensíveis e de fácil manutenção. Os testes no *Cypress* podem ser escritos utilizando a sintaxe do *Cucumber*, que se segue uma estrutura específica, envolvendo a definição de recursos, cenários e passos onde os esses cenários de teste são

descritos em linguagem natural, facilitando a compreensão e a comunicação entre as equipes, além de ser flexível e pode ser adaptada às necessidades específicas de um projeto ou equipe. Além de oferecer uma série de recursos avançados que auxiliam no desenvolvimento e na execução de testes de alta qualidade. Alguns exemplos incluem a execução paralela de testes, a captura de *screenshots* e vídeos durante os testes, a depuração interativa e a capacidade de fazer requisições HTTP diretamente nos testes (CYPRESS DOCUMENTATION, 2023).

2.3 *Metabase*

O *Metabase* é uma plataforma de *BI* que permite a visualização e análise de dados de forma intuitiva e amigável. *BI* envolve a coleta, organização, análise e apresentação de informações relevantes para auxiliar na tomada de decisões estratégicas em uma organização. ele fornece percepções acionáveis a partir dos dados, facilitando a compreensão dos negócios e impulsionando o desempenho organizacional (SHARDA ET AL.., 2019).

Por isso, optou-se por utilizá-lo para a visualização de dados visto que a representação gráfica de informações facilita a compreensão e análise dos dados. Ela permite identificar padrões, tendências e *insights* de maneira rápida e intuitiva. No *Metabase*, a visualização de dados é feita por meio de painéis e gráficos interativos, que podem ser customizados de acordo com as necessidades do usuário, esses painéis são composições de gráficos, tabelas e outros elementos visuais que apresentam informações de forma sintetizada e organizada. Eles permitem monitorar métricas-chave, acompanhar o desempenho e identificar *insights* relevantes de maneira rápida e eficiente. Eles podem ser personalizados e compartilhados com outras pessoas na organização (FEW, 2009).

Focando em mantê-los atualizados, a automação desses dados no *Metabase* permite que estejam sempre acrescidos diariamente. Isso é especialmente útil quando os dados são provenientes de fontes que sofrem atualizações frequentes, garantindo que as visualizações e análises sejam sempre baseadas em informações precisas do dia a dia. (METABASE DOCUMENTATION, 2023).

2.4 Definição do Processo de ETL

O processo ETL (Extract, Transform and Load), é amplamente utilizado na área de *BI* para a integração e preparação de dados para análise (INMON & DINSTEL, 2015; KIMBALL ET AL., 2013). Ele consiste em três etapas fundamentais: extração, transformação e carga (JOVANOVIĆ, 2016).

A etapa de extração envolve a coleta de dados a partir de várias fontes, como bancos de dados, arquivos e serviços web (MATOVU & NWOKEJI, 2021). Durante essa etapa, os dados são recuperados e movidos para um ambiente de processamento posterior.

Após a extração, os dados passam pela etapa de transformação, na qual são modificados e ajustados para atender às necessidades específicas do projeto e às regras de negócio (JOVANOVIĆ, 2016). Durante essa etapa, ocorre a aplicação de operações como filtragem, agregação e correção de erros.

Por fim, os dados transformados são carregados em um sistema de destino, como um *data warehouse* ou um banco de dados analítico, onde ficam disponíveis para a análise (INMON & DINSTEL, 2015). Essa etapa envolve a definição de esquemas e a inserção dos dados de forma eficiente.

O processo ETL é fundamental para garantir a qualidade e a integridade dos dados, bem como para disponibilizar informações relevantes para a tomada de decisões (REDMAN, 2008). Ele permite a integração de dados de diferentes fontes e a preparação dos dados para a análise posterior (KIMBALL ET AL., 2013).

3 IMPLEMENTAÇÃO DA ANÁLISE DE TESTES AUTOMATIZADOS

Para este estudo, foi utilizado o 14.18.0 do *Node.js*[8], *Cypress* versão 7.7.0. Essas versões foram selecionadas por serem compatíveis entre si. A coleta dos dados foi realizada por meio do desenvolvimento de um script de teste automatizado utilizando o próprio *Cypress* que interpretou seu próprio *dashboard online* gratuito através de um teste, utilizando a linguagem *Javascript*. A seguir, este trabalho apresenta as 10 etapas para implementação da análise de teste automatizados por meio de *dashboards*.

3.1 Definição do *software* para o preenchimento do *dashboard*

Durante esta etapa do projeto, foi selecionado um software apropriado para fins de estudo. Devido à natureza sensível da identificação do *software*, a empresa responsável optou por não revelar seu nome específico, mantendo sua identidade e o produto testado em sigilo. No entanto, a empresa colaborou ao permitir o uso das métricas de teste, que foram essenciais para preencher o projeto de teste e para a montagem deste *dashboard*.

[8] NodeJs. Disponível em: <https://nodejs.org/en/>. Acessado em 2 de março de 2023.

Por meio desses dados aplicados, será possível realizar análises e avaliações das métricas obtidas, proporcionando uma compreensão mais aprofundada do desempenho do software e fornecendo *insights* valiosos para o processo de desenvolvimento.

É importante ressaltar que a escolha do software, com a devida autorização da empresa, é um aspecto crucial deste projeto, pois os resultados obtidos baseiam-se em dados reais e representativos do *software* em desenvolvimento, possibilitando uma análise precisa e confiável das métricas de teste.

3.2 Desenvolvimento dos Testes

Após a seleção do *software* em desenvolvimento, o próximo passo consistiu no desenvolvimento de uma abrangente bateria de testes *end-to-end*. Esses testes desempenham um papel fundamental, pois são responsáveis por verificar o código do software a cada mudança e identificar eventuais erros e bugs presentes.

É importante ressaltar que a criação e execução dos testes *end-to-end* são realizadas de forma sistemática e abrangente, abordando diferentes funcionalidades e casos de uso relevantes para o software em desenvolvimento. Essa abordagem minuciosa e completa dos testes assegura que o *software* seja submetido a um processo rigoroso de validação, garantindo a detecção de possíveis falhas e contribuindo para a entrega de um produto de alta qualidade.

3.3 Criação de um banco de dados

A fim de armazenar e organizar os resultados dos testes, foi adotada a utilização de um banco de dados relacional. Para isso, foi escolhido o PostgreSQL por sua licença de código aberto, a confiabilidade, os recursos avançados, sua escalabilidade e compatibilidade com padrões do mercado. Além disso, a ferramenta DBeaver[9] foi selecionada para facilitar a administração e as conexões com o banco de dados, por ser ferramenta gratuita e de código aberto que oferece suporte a vários bancos de dados, incluindo o próprio PostgreSQL, além de possuir uma interface intuitiva e amigável, que facilita a navegação e a execução de consultas SQL.

O primeiro passo nessa etapa foi criar uma conexão no PostgreSQL através do DBeaver, denominada "registroTestes", para estabelecer a comunicação com o banco de

[9] Dbeaver. Disponível em: <https://dbeaver.io/>. Acessado em 2 de março de 2023.

dados. Em seguida, um *database* chamada "postgres" foi criada para acomodar os resultados dos testes.

Para estruturar adequadamente o banco de dados, foram definidas as seguintes colunas:

- **data_run**: Armazena a data em que os testes foram executados.
- **run**: Registra seu número que também é o número de testes executados.
- **link**: Armazena o link de onde os dados foram extraídos.
- **skipped**: Indica a quantidade de testes pulados.
- **pending**: Registra a quantidade de testes que ficaram pendentes (impossibilitados por erros que quebram a cadeia de testes).
- **error**: Indica o número de testes que resultaram em erro.
- **passed**: Registra a quantidade de testes bem-sucedidos.
- **total**: Armazena o total de testes realizados.
- **project_id**: Identifica o ID do projeto ao qual os testes estão relacionados.

Essa estruturação do banco de dados permitirá a organização e o registro sistemático dos resultados dos testes, facilitando a consulta e a análise das métricas presentes no *dashboard*. Além disso, ao utilizar um banco de dados relacional como o PostgreSQL, é possível estabelecer relacionamentos entre os dados, proporcionando uma visão mais abrangente e integrada das informações coletadas nos testes.

3.4 Desenvolvimento de um *script* de coleta

A fim de construir um novo *dashboard* e garantir a correta extração das métricas, foi desenvolvido um *script* em *Javascript* para extrair as métricas relevantes do *dashboard online* do *Cypress* e armazená-las em um arquivo de texto (.txt). A escolha dessa linguagem foi feita devido à sua compatibilidade com o ambiente do *Cypress*, permitindo uma integração eficiente.

O *script* realiza a extração diária das métricas e as organiza em um arquivo de fácil acesso. Essa abordagem centraliza os dados coletados, facilitando a análise e tomada de decisões. Ele desempenha um papel fundamental no processo, contribuindo para uma análise mais precisa e eficiente dos resultados, o mesmo se encontra disponível no *Bitbucket*[10].

[10] Script desenvolvido. Disponível em: <https://bitbucket.org/MatheusMCamposSklonny/script/src/master//>. Acessado em 2 de março de 2023.

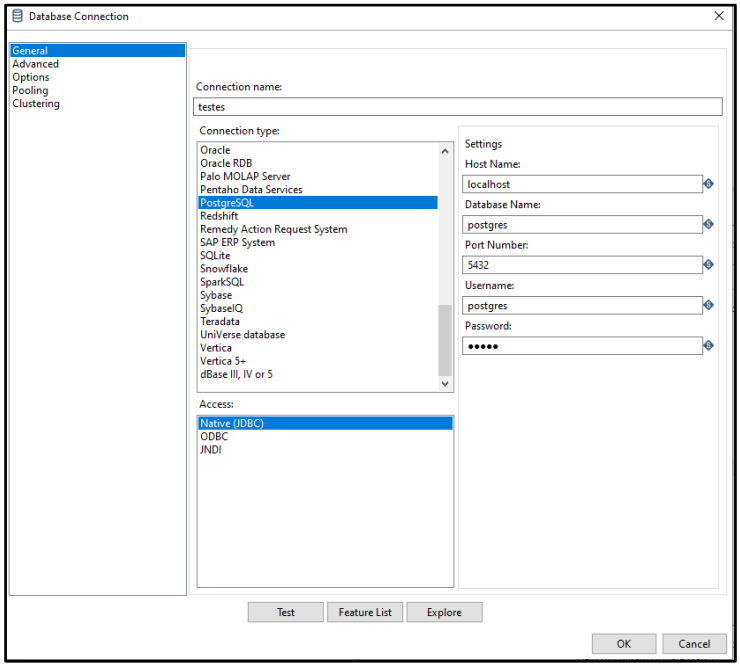
3.5 *Script* de conversão

A fim de construir o banco de dados que será usado para alimentar o nosso *dashboard*, é necessário realizar a conversão do arquivo de texto para o formato XML. Essa conversão é realizada por meio de um *script* desenvolvido no *software* Pentaho.

No processo de conversão, estabelecemos uma conexão direta com o banco de dados, inserindo as credenciais adequadas para permitir a integração perfeita entre o Pentaho e o banco. Essa fase do projeto foi denominada de "transformação pentaho".

O Pentaho oferece uma plataforma para a execução de tarefas de *ETL* de dados. Conforme apresentado na Figura 1, utilizando as devidas configurações, é possível extrair os dados do arquivo de texto e transformá-los em um formato compatível com o banco de dados.

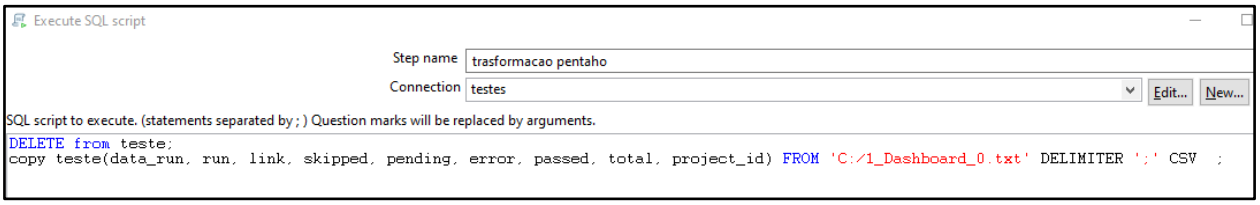
Figura 1 - Configurações utilizadas no Pentaho para gerar o script de conversão.



Fonte: Imagem produzida pelo próprio autor.

Ao realizar essa etapa de conversão utilizando o Pentaho, obtemos um resultado consistente e de alta qualidade, garantindo que os dados estejam prontos para serem utilizados na criação do *dashboard* e na geração das métricas desejadas, para isso se definiu uma *query* para tal conversão como pode ser visto na Figura 2.

Figura 2 - Definição da *query* que irá converter o arquivo de texto em um CSV.



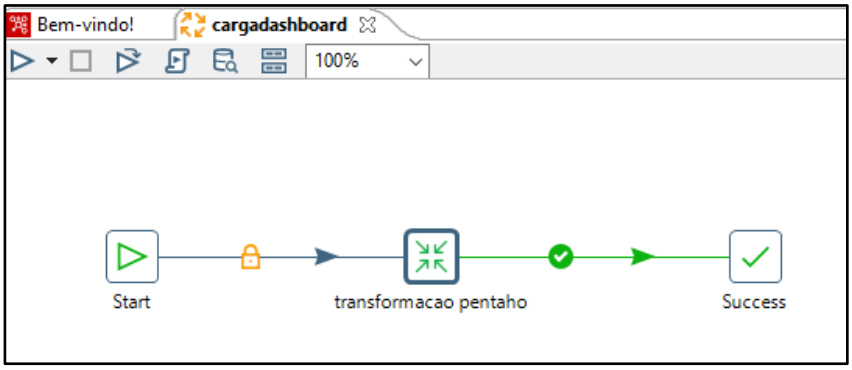
Fonte: Imagem produzida pelo próprio autor.

Para inserir dados na tabela "teste" a partir de um arquivo CSV localizado em *'C:/1_Dashboard_0.txt'*, foi utilizado o seguinte comando. Esse comando realiza a importação dos dados do arquivo para as colunas específicas da tabela, utilizando o delimitador ';' e considerando o formato do arquivo como CSV.

3.6 Criação do *Job*

Continuando no Pentaho, foi criado um *job* que será responsável por gerar o arquivo ".bat" para automatizar o processo. O *job* será salvo com a extensão ".kjb" e nomeado como "cargadashboard.kjb". O *job* terá a estrutura que pode ser visualizada na figura 3.

Figura 3 - Tela de criação do j*ob* no Pentaho



Fonte: Imagem produzida pelo próprio autor.

Início do *Job*: O *job* começa aqui.

1. **Etapa 1:** Conexão com o banco de dados: Se estabelece a conexão com o banco de dados PostgresSQL usando as credenciais adequadas.
2. **Etapa 2:** Transformação dos dados: Nesta etapa, se executa a transformação dos dados necessária para prepará-los para o carregamento no *dashboard*. Isso pode incluir filtros, transformações de colunas, cálculos, entre outros.

3. **Etapa 3:** Criação do arquivo de saída: Cria-se um arquivo de saída com os dados transformados. Este arquivo será utilizado posteriormente na etapa de carga do *dashboard.*
4. **Fim do *Job*:** O *job* é concluído.

O *job* "*cargadashboard.kjb*" será executado no Pentaho para gerar o arquivo ".bat" que automatizará o processo de carga dos dados no *dashboard*. Deve-se configurar corretamente as etapas do *job* de acordo com suas necessidades e especificações do seu ambiente.

Ao executar o *job*, o arquivo ".bat" será gerado e estará pronto para ser executado, realizando a carga dos dados no *dashboard* de forma automatizada. Em resumo, esse arquivo .bat é usado para automatizar a carga dos dados para o *dashboard* usando o *Pentaho Data Integration* e PostgreSQL. Ele exibe algumas mensagens visuais, define variáveis de ambiente, e executa o arquivo de *job "cargadashboard.kjb"* por meio do comando *"Kitchen.bat".* O *log* da execução é redirecionado. Agora com o .bat criado basta iniciá-lo para fazer a atualização da base.

3.7 Conexão com *Metabase*

Após a base de dados ser corretamente alimentada pelo arquivo convertido, é necessário iniciar a integração com a ferramenta *Metabase*. A *Metabase* é uma plataforma de análise e visualização de dados que permite criar painéis e fazer perguntas interativas sobre os dados armazenados.

Em seguida a instalação da *Metabase*, o próximo passo é configurar as credenciais de acesso e inserir um novo banco de dados, lembrando que é importante garantir que as permissões adequadas sejam configuradas para o usuário que irá interagir com a *Metabase* (METABASE DOCUMENTATION, 2023).

3.8 Criação das Perguntas e *Dashboards*

Depois de realizar as integrações necessárias, o próximo passo é segmentar os dados por meio da criação de "perguntas" na *Metabase*.

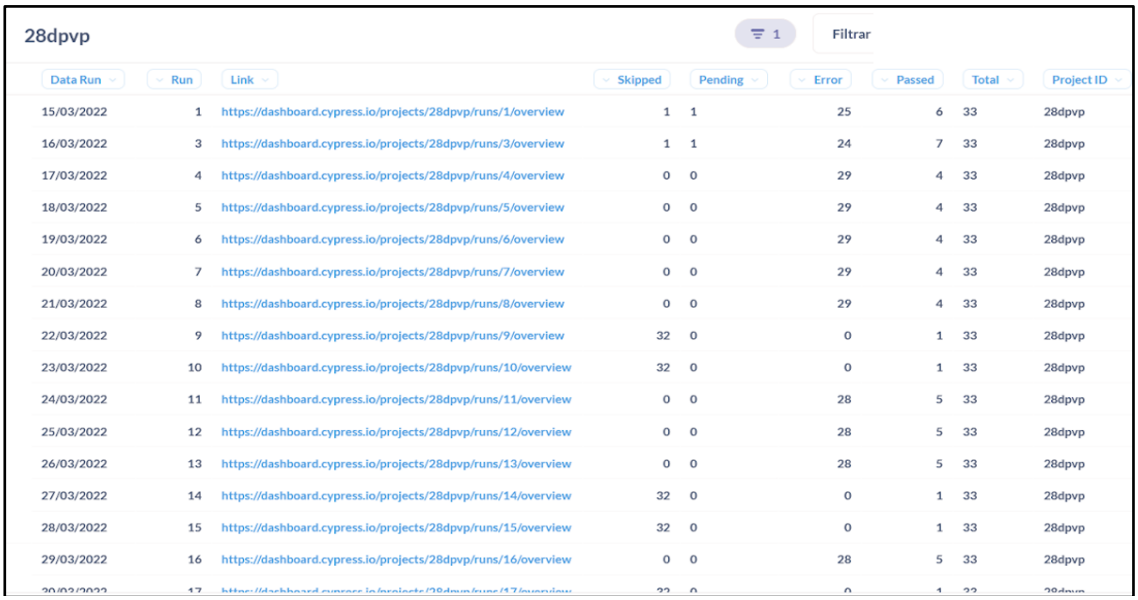
As "perguntas" são consultas interativas que podem ser feitas à base de dados para obter informações específicas. Essas perguntas permitem explorar os dados de forma mais detalhada e personalizada, filtrando e agrupando os resultados de acordo com os critérios desejados.

Ao criar uma pergunta na *Metabase*, opção encontrada no menu “novo” no canto superior direito, foi selecionada a tabela onde os dados foram inseridos e, em seguida, aplicar filtros e condições para restringir os resultados. Pode-se especificar os campos que deseja incluir na pergunta e definir métricas, como contagem, soma ou média, para realizar cálculos sobre os dados.

O próximo passo é selecionar a tabela onde os dados foram inseridos e aplicar filtros para segmentá-los de acordo com o *"Project ID*" desejado. Essa segmentação permite visualizar os dados específicos relacionados a um determinado projeto.

Uma tabela será gerada, que poderá ser salva para posteriormente ser usada para mais “perguntas”, há um exemplo de tabela na Figura 4.

Figura 4 - Tabela criada com base do filtro *ProjectID* exibindo apenas dados selecionados.



28dpvp

1 Filtrar

Data Run	Run	Link	Skipped	Pending	Error	Passed	Total	Project ID
15/03/2022	1	https://dashboard.cypress.io/projects/28dpvp/runs/1/overview	1	1	25	6	33	28dpvp
16/03/2022	3	https://dashboard.cypress.io/projects/28dpvp/runs/3/overview	1	1	24	7	33	28dpvp
17/03/2022	4	https://dashboard.cypress.io/projects/28dpvp/runs/4/overview	0	0	29	4	33	28dpvp
18/03/2022	5	https://dashboard.cypress.io/projects/28dpvp/runs/5/overview	0	0	29	4	33	28dpvp
19/03/2022	6	https://dashboard.cypress.io/projects/28dpvp/runs/6/overview	0	0	29	4	33	28dpvp
20/03/2022	7	https://dashboard.cypress.io/projects/28dpvp/runs/7/overview	0	0	29	4	33	28dpvp
21/03/2022	8	https://dashboard.cypress.io/projects/28dpvp/runs/8/overview	0	0	29	4	33	28dpvp
22/03/2022	9	https://dashboard.cypress.io/projects/28dpvp/runs/9/overview	32	0	0	1	33	28dpvp
23/03/2022	10	https://dashboard.cypress.io/projects/28dpvp/runs/10/overview	32	0	0	1	33	28dpvp
24/03/2022	11	https://dashboard.cypress.io/projects/28dpvp/runs/11/overview	0	0	28	5	33	28dpvp
25/03/2022	12	https://dashboard.cypress.io/projects/28dpvp/runs/12/overview	0	0	28	5	33	28dpvp
26/03/2022	13	https://dashboard.cypress.io/projects/28dpvp/runs/13/overview	0	0	28	5	33	28dpvp
27/03/2022	14	https://dashboard.cypress.io/projects/28dpvp/runs/14/overview	32	0	0	1	33	28dpvp
28/03/2022	15	https://dashboard.cypress.io/projects/28dpvp/runs/15/overview	32	0	0	1	33	28dpvp
29/03/2022	16	https://dashboard.cypress.io/projects/28dpvp/runs/16/overview	0	0	28	5	33	28dpvp

Fonte: Imagem produzida pelo próprio autor.

Para gerar o *dashboard* na *Metabase*, é necessário criar um novo painel no mesmo. Um painel é uma coleção de visualizações e perguntas relacionadas que são organizadas em uma única página para fornecer uma visão abrangente dos dados, para a criação de um painel o menu da Figura 6 será usado na opção “painel” que irá abrir uma caixa de diálogo.

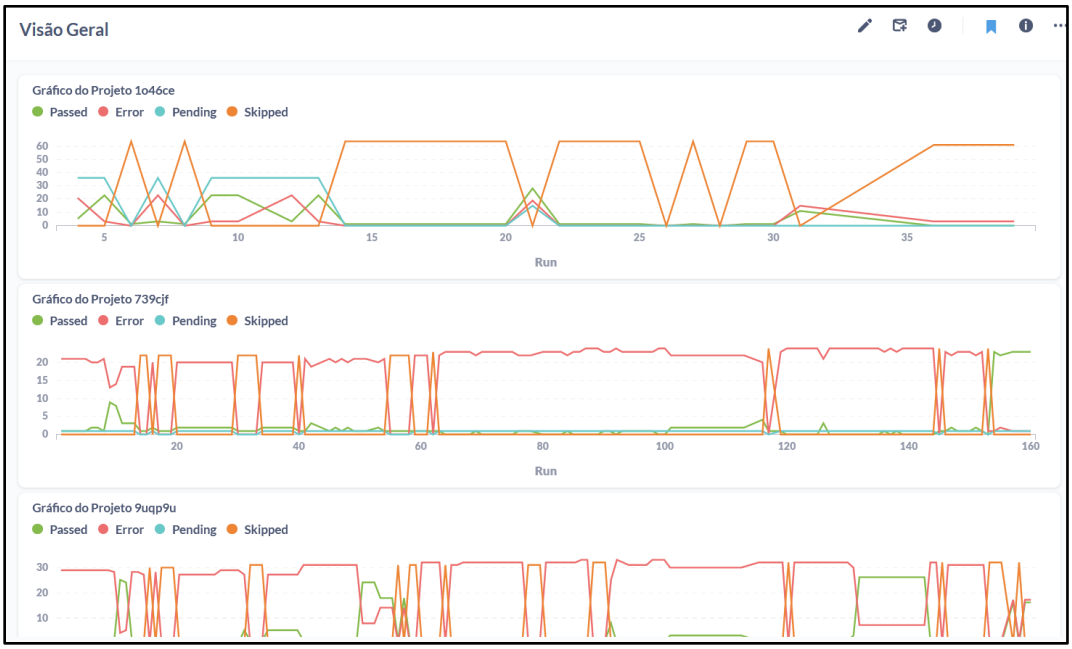
Nesse painel pode-se adicionar diversas “perguntas” tanto em formato de tabelas ou de formas gráficas definidas pelas configurações desejadas pelo usuário o que facilitará a visualização de diversas métricas e registros visuais extraídos, fazendo assim uma nova mostra de informações e possibilitando a criação de novas funções dessas métrica permitindo que haja um monitoramento dessas métricas focados em indicadores chaves definidos a

necessidade do usuário, uma análise comparativa que permitirá visualizar e contestar diferentes dados entre projetos e entre o desenvolvimento do projeto além do compartilhamento de informações que permite gerar e exportar estes painéis via PDF ou CSV, painel esse que foi exemplificado na Figura 5 e 6 com algumas extrações de dados mais pertinentes.

Figura 5 - Exemplo de painel completo.

Fonte: Imagem produzida pelo próprio autor.

Figura 6 - Exemplo de painel geral com dados de diferentes projetos.



Fonte: Imagem produzida pelo próprio autor.

3.9 Automatização do Processo

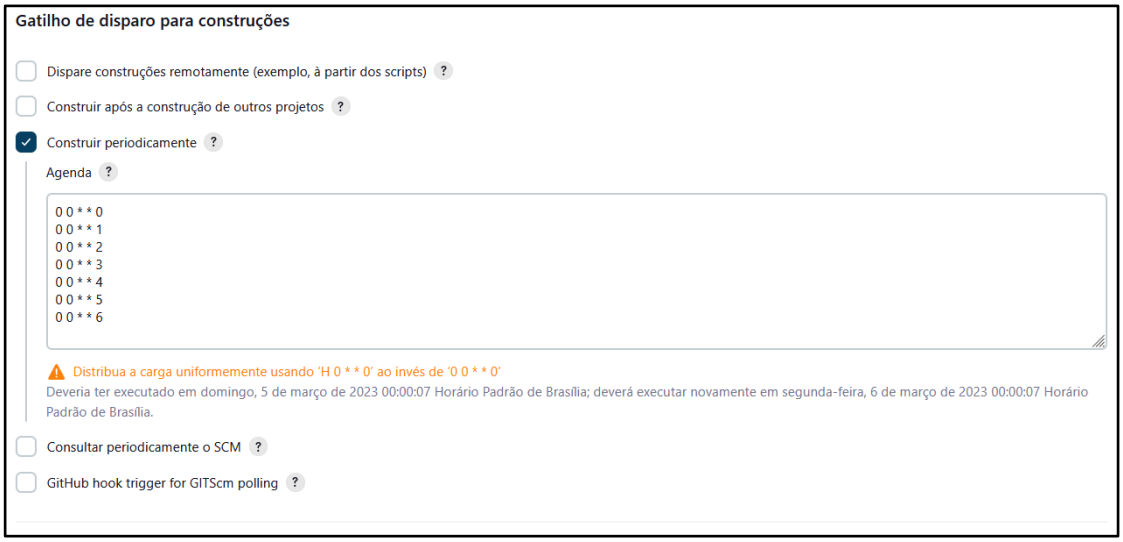
Para automatizar o processo e torná-lo mais confortável ao usuário, foi utilizado o *software* Jenkins. Após suas devidas instalações foi criado criar um j*ob* de nome *Dashboard* Automatizado. Primeiro deve-se inserir a URL do projeto com o *script* de coleta como apresentado na Figura 7.

Figura 7 - Demonstração do preenchimento dos campos e repositório onde o projeto foi exportado.

Fonte: Imagem produzida pelo próprio autor.

Adiante criou-se um “gatilho de disparo de construção". Há diversos tipos, um dos mais usados é construir após a construção de outro projeto, podendo executar apenas após o projeto testado está construído. Há um exemplo na Figura 8 de um gatilho diário e em seguida inserir o comando “*npm run gerar*” como está abaixo na figura 9.

Figura 8 - Exemplo de Gatilho de construção.



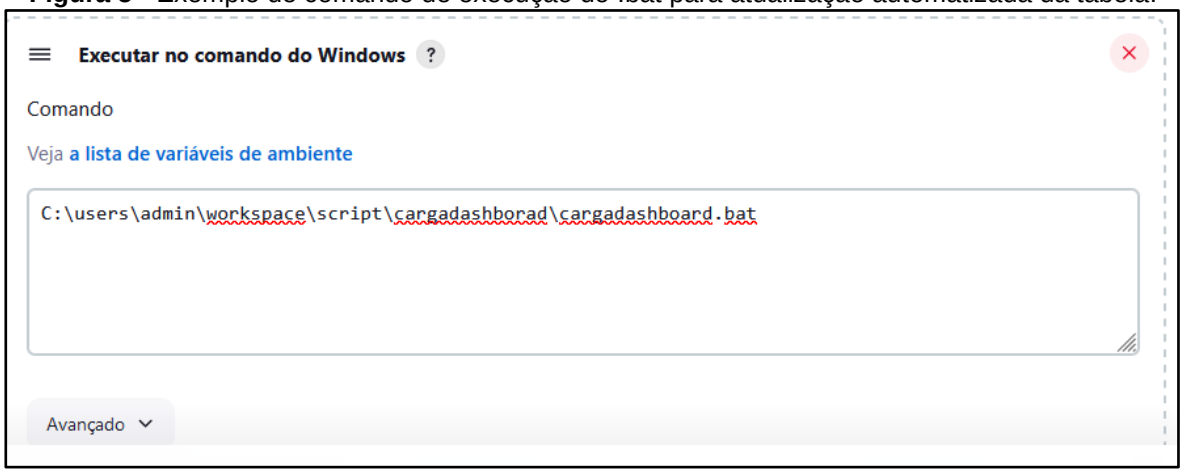
Fonte: Imagem produzida pelo próprio autor.

Figura 9 - Demonstração dos comandos de execução do projeto.

Fonte: Imagem produzida pelo próprio autor.

E por fim a execução do arquivo em lote (.bat) apontando seu diretório do *workspace* do Jenkins como exemplificado na Figura 10.

Figura 5 - Exemplo de comando de execução do .bat para atualização automatizada da tabela.



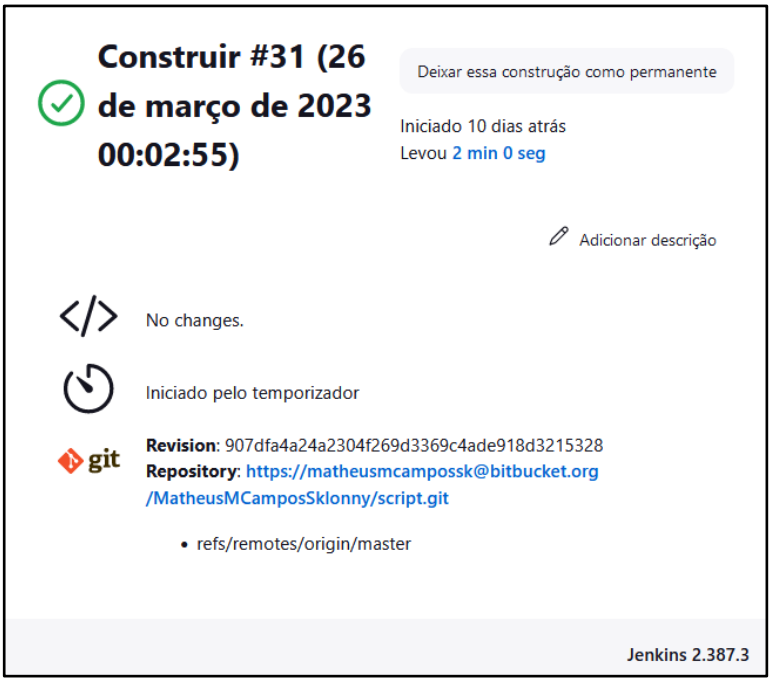
Fonte: Imagem produzida pelo próprio autor.

3.10 Construção do Projeto

Com o objetivo de tornar o processo mais eficiente e conveniente para o usuário, foi implementada a automatização do projeto por meio do software Jenkins. Após a instalação e configuração adequadas do Jenkins, foi criado um j*ob* denominado "*Dashboard* Automatizado".

Dessa forma, ao aguardar o horário programado ou acionar manualmente a execução do j*ob* e que se apresentará como na Figura 11, o projeto é construído e os dados do *dashboard* são atualizados automaticamente. Essa automatização torna o processo mais conveniente, evitando a necessidade de atualizações manuais frequentes e garantindo que as métricas estejam sempre atualizadas e disponíveis para análise.

Figura 11 - Exemplo de construção finalizada.



Fonte: Imagem produzida pelo próprio autor.

4 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi a criação de um *dashboard* de testes utilizando o *framework Cypress*, integrado ao *Metabase*. Ao longo do desenvolvimento, foram realizadas etapas fundamentais, como a seleção de um software em desenvolvimento para obtenção de resultados reais, o desenvolvimento dos testes *end-to-end*, a criação de um banco de dados para armazenar os resultados dos testes, o desenvolvimento de um *script* de coleta dos resultados, a conversão dos dados para XML utilizando o *software* Pentaho, a criação de um arquivo em lote (.bat) para automatizar a atualização dos dados, a conexão com o *Metabase* e a criação de perguntas e *dashboards*.

A criação do dashboard de testes *end-to-end* possibilitou a visualização de métricas e resultados de forma organizada e intuitiva. A integração entre o *Cypress* é até o *Metabase* permitiu a centralização dos dados e facilitou a leitura e interpretação dos resultados pelos usuários.

A criação do banco de dados utilizando o PostgreSQL e o gerenciamento das conexões com o DBeaver foram essenciais para armazenar e organizar os resultados dos testes. A utilização do Pentaho para a conversão dos dados e a criação do arquivo em .bat para a automação do processo de atualização do *dashboard* foram passos importantes para

otimizar o fluxo de trabalho.

A conexão com o *Metabase* permitiu a criação de perguntas personalizadas e a geração de *dashboards* com diferentes métricas e visualizações. Isso possibilitou uma análise comparativa entre projetos, a monitorização de indicadores-chave e o compartilhamento de informações através de exportação em PDF ou CSV.

A automação do processo, utilizando o *software* Jenkins, trouxe mais conforto e praticidade aos usuários. A construção periódica do projeto e a atualização automatizada dos dados garantiram que as informações no *dashboard* estivessem sempre atualizadas.

No entanto, é importante destacar que o desenvolvimento deste *dashboard* de testes *end-to-end* com *Cypress* no *Metabase* apresentou desafios técnicos, como a configuração adequada dos ambientes, a criação das consultas personalizadas e a integração entre as diferentes ferramentas. Superar esses desafios exigiu pesquisa, dedicação e auxílio do orientador.

Em conclusão, este trabalho proporcionou a criação de um *dashboard*, fornecendo uma ferramenta eficaz para visualização e análise dos resultados dos testes. A automação do processo e a centralização dos dados facilitaram a tomada de decisões e contribuíram para a melhoria contínua da qualidade do *software*. Recomenda-se a utilização desta abordagem em projetos futuros, adaptando-a conforme as necessidades específicas de cada contexto.

REFERÊNCIAS

Beizer, B. (1990). **Software Testing Techniques**. Van Nostrand Reinhold.

BERTOLINO, **A. Software Testing Research: Achievements, Challenges, Dreams.** In: International Conference on Software Engineering. IEEE, 2007. p. 1-1.

BOUMAN, Roland; VAN DONGEN, Jos. **Pentaho Solutions: Business Intelligence and Data Warehousing with Pentaho and MySQL**. Hoboken: Wiley, 2010.

CYPRESS Documentation. In: **Why Cypress?**: Features. [S. l.], 30 jan. 2023. Disponível em: https://docs.cypress.io/guides/overview/why-cypress/. Acesso em: 2 mar. 2023.

FEW, S. **Now You See It: Simple Visualization Techniques for Quantitative Analysis**. Analytics Press, 2009.

FEWSTER, M.; GRAHAM, D. **Software Test Automation: Effective Use of Test Execution Tools**. Addison-Wesley Professional, 2012.

Inmon, W. H., & Dinstel, S. (2015). **Building the Data Warehouse**. John Wiley & Sons.

Jovanović, M. (2016). **ETL Process: Concepts, Tools, and Challenges**. International Journal

of Advanced Computer Science and Applications, 7(9), 49-55.

Khan, A. M. (2016). **Visualizing Software Test Results**. In: Proceedings of the International Conference on Advances in Computing, Communication and Information Technology (ICACCI). IEEE.

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2013). **The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling**. John Wiley & Son.

Matovu, R., & Nwokeji, J. C. (2021). **A Systematic Literature Review on Big Data Extraction, Transformation and Loading (ETL)**. IIntelligent Computing, Proceedings of the 2021 Computing Conference, Volume 2 (pp.308-324).

METABASE documentation. In: **Metabase documentation**: Getting started with Metabase. [S. l.], c2023. Disponível em: https://www.metabase.com/learn/getting-started/getting-started. Acesso em: 2 mar. 2023.

PENTAHO Documentation. In: **Work with data**. [S. l.], 04 Jun 2023. Disponível em: https://help.hitachivantara.com/Documentation/Pentaho/9.5. Acesso em: 1 Jun. 2023.

PRESSMAN, R. S. **Engenharia de Software: Uma Abordagem Profissional**. McGraw-Hill, 2014.

Redman, T. C. (2008). **Data Driven: Profiting from Your Most Important Business**. Harvard Business Press.

SHARDA, R.; DELEN, D.; TURBAN, E.; ARONSON, J. E. **Business Intelligence e Análise de Dados para Gestão do Negócio**. Pearson, 2019.