INSTITUTO FEDERAL DE SERGIPE

SISTEMAS DE INFORMAÇÃO

Reinan Gabriel Dos Santos Souza

Automatização e padronização da escrita acadêmica com Limarka e Marp: Um estudo de caso para o IFS

Lagarto - SE

2024

Reinan Gabriel Dos Santos Souza

Automatização e padronização da escrita acadêmica com Limarka e Marp: Um estudo de caso para o IFS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Lagarto do Instituto Federal de Educação, Ciência e Tecnologia, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.
Área de concentração: Computação.

Instituto Federal de Sergipe

Sistemas de informação

Orientador: Prof. Dr. Gilson Pereira Dos Santos Junior

Coorientador: Prof. MSc. Francisco Rodrigues Santos

Lagarto - SE

2024

Souza, Reinan Gabriel dos Santos.
S718a Automatização e padronização da escrita acadêmica com Limarka e Marp: um estudo de caso para o IFS / Reinan Gabriel dos Santos Souza. – Lagarto, 2024.
74 f. ; il.

Monografia (Graduação) – Bacharelado em Sistemas de Informação. Instituto Federal de Educação Ciência e Tecnologia de Sergipe – IFS, 2024.
Orientador: Prof. Dr. Gilson Pereira dos Santos Junior.
Co-Orientador: Prof. MSc. Francisco Rodrigues Santos.

1. Programa de computador. 2. Formatação. 3. Educação. 4. Informática-ensino aprendizagem. I. Instituto Federal de Educação Ciência e Tecnologia de Sergipe – IFS. II. Título.

CDU 37.018.43:004

Reinan Gabriel Dos Santos Souza

Automatização e padronização da escrita acadêmica com Limarka e Marp: Um estudo de caso para o IFS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Lagarto do Instituto Federal de Educação, Ciência e Tecnologia, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.
Área de concentração: Computação.

Monografia aprovada. Lagarto - SE, 04 de outubro de 2024:

Prof. Dr. Gilson Pereira Dos Santos Junior
Orientador

Prof. MSc. Francisco Rodrigues Santos
Coorientador

Prof. MSc. Gustavo da Silva Quirino
Convidado

Prof. MSc. Telmo Oliveira De Jesus
Convidado

Lagarto - SE
2024

Agradecimentos

Gostaria de expressar meus mais sinceros agradecimentos aos professores do Instituto Federal de Educação, Ciência e Tecnologia de Sergipe (IFS). Este trabalho de conclusão de curso só foi possível graças à orientação, ao apoio e à sabedoria que vocês generosamente compartilharam ao longo da minha jornada acadêmica.

Em especial, gostaria de agradecer ao Professor Francisco Rodrigues, que me ajudou imensamente ao longo do curso, sempre disposto a compartilhar seu conhecimento e fornecer orientações fundamentais para o meu desenvolvimento acadêmico. Seu apoio foi crucial em momentos decisivos, e por isso, sou profundamente grato.

Também gostaria de expressar meu agradecimento ao Marcelo Castro, meu chefe no trabalho, que não apenas me incentivou a concluir o TCC, mas também ajudou a gerenciar minha carga horária de maneira flexível, permitindo que eu conciliasse o trabalho com os compromissos acadêmicos. Sua compreensão e apoio foram determinantes para que eu pudesse seguir com meus estudos e concluir este importante ciclo.

Este trabalho é fruto do aprendizado e da dedicação que testemunhei no IFS. Agradeço a todos os professores e colegas que fizeram parte desta jornada, contribuindo de maneira significativa para o meu crescimento acadêmico e pessoal.

O sucesso é ir de fracasso em fracasso sem perder o entusiasmo. (Winston Churchill)

Resumo

Este trabalho apresenta o desenvolvimento de um *template* para a elaboração de trabalhos acadêmicos e apresentações, voltado especificamente para os alunos do curso de Bacharelado em Sistemas de Informação (BSI) do Instituto Federal de Sergipe (IFS). Utilizando as ferramentas Limarka e Marp, foi criado um ambiente unificado que permite a formatação automática de documentos conforme as normas da ABNT e a criação de *slides* para apresentações de TCC. A metodologia aplicada incluiu a análise comparativa de diversas plataformas de escrita acadêmica, a customização de temas visuais adequados ao contexto institucional, e a implementação de uma *pipeline* de Integração Contínua/Entrega Contínua (CI/CD) para automatizar a compilação e a distribuição dos documentos. Os resultados demonstraram que o *template* desenvolvido não só atende às necessidades acadêmicas, mas também simplifica o processo de escrita, formatação e apresentação, contribuindo para a eficiência e a qualidade dos trabalhos acadêmicos produzidos.

Palavras-chave: Limarka; Markdown; Automação; Marp; Ferramenta; Aprimoramento; Produtividade

Abstract

This work presents the development of a template for the preparation of academic papers and presentations, specifically aimed at students of the Bachelor of Information Systems (BSI) course at the Federal Institute of Sergipe (IFS). Utilizing the Limarka and Marp tools, a unified environment was created that enables the automatic formatting of documents according to ABNT standards and the creation of slides for thesis presentations. The applied methodology included a comparative analysis of various academic writing platforms, the customization of visual themes suitable for the institutional context, and the implementation of a Continuous Integration/Continuous Delivery (CI/CD) pipeline to automate the compilation and distribution of documents. The results demonstrated that the developed template not only meets academic needs but also simplifies the process of writing, formatting, and presenting, contributing to the efficiency and quality of the academic works produced.

Keywords: Limarka; Markdown; Automation; Marp; Tool; Improvement; Productivity

Lista de ilustrações

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
BSI	Bacharelado em Sistemas de Informação
CD	Continuous Deployment/Delivery
CI	Continuous Integration
CLI	Command Line Interface
CSS	Cascading Style Sheets
DaC	Docs as Code
GNU	GNU's Not Unix
HTML	Hypertext Markup Language
IFS	Instituto Federal de Sergipe
LaTeX	Lamport TeX
MS	Microsoft
PDF	Portable Document Format
PPTX	PowerPoint Open XML Presentation
SIGAA	Sistema Integrado de Gestão de Atividades Acadêmicas
TCC	Trabalho de Conclusão de Curso
VS Code	Visual Studio Code

Sumário

1 Introdução

A elaboração de trabalhos acadêmicos, particularmente um Trabalho de Conclusão de Curso (TCC), é um marco fundamental na formação dos estudantes de nível superior, representando a culminação de anos de aprendizado e pesquisa. De acordo com Felipe (2019), o TCC marca o começo da produção científica no ensino superior, constituindo o requisito final para a aprovação de estudantes na graduação e podendo também contribuir para os cursos de pós-graduação.

Nesse contexto, garantir a conformidade com as normas acadêmicas e manter um elevado padrão de qualidade é essencial para que os trabalhos reflitam adequadamente o conhecimento adquirido ao longo do curso. No entanto, o processo de formatação e organização desses documentos pode ser desafiador, especialmente quando realizado manualmente, o que proporciona uma inconsistências e erros que comprometem a apresentação final.

Para atender a essa demanda, ferramentas de automação e formatação, como o Limarka [1], têm ganhado destaque. Segundo Medeiros (2021), O Limarka é uma ferramenta voltada para a formatação automática de documentos acadêmicos em conformidade com as normas da Associação Brasileira de Normas Técnicas (ABNT). Sua utilização tem se mostrado valiosa para garantir que os trabalhos estejam em conformidade com as exigências acadêmicas. No entanto, como toda ferramenta, o Limarka enfrenta desafios que precisam ser superados para que sua eficácia e abrangência sejam maximizadas, como por exemplo, a dificuldade na instalação e nas configurações iniciais.

O presente TCC tem como objetivo principal o aprimoramento do Limarka, integrando-o com metodologias modernas de desenvolvimento de software, práticas de automação e ferramentas colaborativas, visando criar um ambiente de trabalho mais eficiente e integrado para os estudantes do curso de Bacharelado em Sistemas de Informação (BSI) do Instituto Federal de Sergipe (IFS). Uma das principais inovações propostas é a integração do Limarka com o Marp [2]. Para Ayers (2023), Marp é uma ferramenta de apresentação robusta e de fácil manuseio que facilita a elaboração de slides visualmente impressionantes por meio do uso de Markdown. Essa integração permite que os estudantes desenvolvam não apenas seus TCCs, mas também as apresentações de defesa, dentro de um mesmo ambiente de trabalho, garantindo consistência visual e eficiência no processo de criação.

Outro aspecto central deste projeto é a reestruturação da documentação do Limarka, utilizando a metodologia *Docs as Code* (DaC). Esta abordagem trata a documentação com a mesma importância que o código-fonte, garantindo que ela evolua em paralelo com o software

[1] https://github.com/abntex/limarka
[2] https://marp.app/

e esteja sempre atualizada (SILVA, 2022). A reescrita da documentação do Limarka não apenas facilita o acesso dos estudantes às informações necessárias, como também promove uma maior autonomia no uso da ferramenta, tornando o processo de aprendizado mais intuitivo e menos suscetível a erros comuns.

A implementação de *pipelines* automatizadas de integração contínua (CI) para a compilação e publicação de documentos e apresentações representa uma evolução significativa no fluxo de trabalho dos estudantes. Essas *pipelines* automatizam o processo de geração de diferentes formatos de saída como Portable Document Format (PDF), Hypertext Markup Language (HTML), PowerPoint Open XML Presentation (PPTX), assegurando que cada versão do documento esteja em conformidade com as normas e pronta para ser utilizada ou compartilhada. Segundo RedHat (2023), conforme os aplicativos evoluem, as funcionalidades de CI/CD contribuem para a redução da complexidade, o aumento da eficiência e a simplificação dos processos de trabalho.

O trabalho também se propôs a criar um tema acadêmico específico para o Marp, alinhado aos padrões visuais utilizados no IFS, proporcionando aos alunos um recurso estilístico que atende às expectativas institucionais. A criação desse tema, bem como sua integração ao fluxo de trabalho do Limarka, facilita a padronização das apresentações e reforça a identidade visual dos trabalhos acadêmicos apresentados pelos estudantes.

Portanto, este TCC não se limita apenas a aprimorar uma ferramenta existente, mas busca estabelecer um novo paradigma para a elaboração de TCC, integrando práticas modernas de desenvolvimento, automação e colaboração. As melhorias introduzidas têm o potencial de transformar o processo de escrita acadêmica no curso de BSI do IFS Campus Lagarto, tornando-o mais eficiente, acessível e alinhado às melhores práticas da indústria de software. Acredita-se que as soluções apresentadas não apenas beneficiarão os alunos durante a elaboração de seus TCCs, mas também os prepararão para enfrentar os desafios técnicos e profissionais que encontrarão em suas carreiras futuras.

1.1 Motivação

A motivação deste trabalho consiste em oferecer uma solução para os estudantes de BSI do IFS que simplifique o processo de elaboração de seus trabalhos de conclusão de curso. Ao integrar ferramentas como o Limarka e o Marp, o projeto visa proporcionar uma experiência automatizada e padronizada para a formatação e apresentação do TCC, aliviando os alunos das complexidades relacionadas às normas da ABNT e à criação de apresentações de qualidade.

Além disso, o trabalho busca preparar os estudantes para o mercado de trabalho, ao introduzir práticas modernas de desenvolvimento colaborativo e automatizado, utilizando *pipelines* de integração contínua e uma documentação acessível e atualizada. Dessa forma, a proposta não só atende às necessidades imediatas dos alunos, mas também os capacita com habilidades

valiosas para suas futuras carreiras.

1.2 Objetivo

Neste tópico são apresentados os objetivos desta pesquisa, divididos em geral específicos.

1.2.1 Objetivo Geral

Desenvolver e implementar um template para trabalhos acadêmicos utilizando o Limarka e Marp, que facilite a elaboração de documentos e apresentações no contexto do curso de BSI do IFS, assegurando a conformidade com as normas da ABNT e promovendo a integração de práticas modernas de desenvolvimento colaborativo.

1.2.2 Objetivos Específicos

- Comparar diferentes ferramentas de escrita acadêmica, identificando as vantagens e desvantagens de cada uma para determinar as melhores práticas na criação de documentos e apresentações;
- Integrar o Limarka e o Marp ao ambiente de desenvolvimento dos alunos, criando um fluxo de trabalho que simplifique a criação e formatação de um TCC e apresentações;
- Desenvolver um tema personalizado para o Marp, alinhado aos padrões visuais e acadêmicos do IFS, facilitando a criação de apresentações padronizadas;
- Implementar uma *pipeline* de CI/CD para automatizar a compilação e publicação de documentos e apresentações em formatos diversos, como PDF, HTML e PPTX.

1.3 Metodologia

A metodologia adotada neste trabalho é de natureza aplicada, com o objetivo de desenvolver um *template* para a elaboração de trabalhos acadêmicos e apresentações, utilizando as ferramentas Limarka e Marp, voltadas ao contexto do curso de BSI do IFS. O desenvolvimento foi conduzido de forma iterativa e incremental, seguindo as etapas descritas a seguir.

1.3.1 Revisão bibliográfica e análise de ferramentas

Inicialmente, foi realizada uma revisão bibliográfica para identificar as melhores práticas e as ferramentas disponíveis para a elaboração de trabalhos acadêmicos e apresentações. A revisão incluiu a análise de ferramentas como Manubot [3], Authorea [4], Overleaf [5], Google Docs

[3] https://manubot.org/
[4] https://www.authorea.com/
[5] https://pt.overleaf.com/

[6], Microsoft Word [7] e LibreOffice Writer [8]. Cada ferramenta foi avaliada com base em suas funcionalidades, facilidade de uso, suporte a normas acadêmicas, e capacidade de integração com sistemas de controle de versão. Esta análise comparativa foi fundamental para embasar a escolha do Limarka e do Marp como ferramentas principais para o desenvolvimento do *template*.

1.3.2 Definição dos requisitos do template

Com base em um estudo realizado com quatro alunos na disciplina de Trabalho de Conclusão de Curso I do curso de BSI, foram definidos os requisitos funcionais e não funcionais do *template*. Esses requisitos incluem a conformidade com as normas da ABNT adotadas pelo IFS, a personalização do tema para apresentações no Marp, a integração com sistemas de controle de versão e a facilidade de uso para alunos com diferentes níveis de familiaridade com ferramentas de formatação e desenvolvimento de software.

1.3.3 Desenvolvimento do template e customização do Marp

A etapa seguinte envolveu o desenvolvimento do *template* de escrita acadêmica utilizando o Limarka, e a criação de um tema personalizado para o Marp, adaptado aos padrões visuais e acadêmicos do IFS. O tema do Marp foi implementado em *Cascading Style Sheets* (CSS), garantindo que as apresentações sigam um estilo coerente e profissional. Adicionalmente, o *template* foi integrado ao ambiente de desenvolvimento, facilitando o processo de formatação e a modularização da escrita.

1.3.4 Implementação de Pipeline de CI/CD

Para garantir a automação e a consistência na compilação e distribuição dos documentos e apresentações, foi implementada uma *pipeline* de CI/CD no GitHub. A *pipeline* automatiza a compilação dos slides e documentos em formatos PDF, HTML e PPTX, a partir dos arquivos Markdown criados pelos alunos. A cada alteração enviada para o repositório principal, a *pipeline* é acionada, gerando os artefatos nos formatos desejados e disponibilizando-os para *download*.

1.3.5 Validação e testes

A validação do *template* e das ferramentas desenvolvidas foi realizada por meio de testes práticos com um aluno, a quem essa ferramenta foi apresentada. Esse aluno utilizou o *template* para a criação de seu trabalho acadêmico e apresentação, fornecendo *feedback* sobre a usabilidade, a adequação às normas da ABNT e a eficácia das ferramentas na simplificação do

[6] https://docs.google.com/
[7] https://www.microsoft.com/pt-br/microsoft-365/word
[8] https://pt-br.libreoffice.org/

processo de escrita e formatação. Com base nas considerações recebidas, foram feitas melhorias incrementais nas ferramentas e na documentação.

1.3.6 Documentação e treinamento

Por fim, foi elaborada uma documentação detalhada, utilizando a abordagem de DaC, para guiar os alunos na utilização das ferramentas desenvolvidas. A documentação abrange todo o processo, desde a instalação das ferramentas até a personalização avançada do *template* e do tema do Marp, assegurando que os alunos possam tirar o máximo proveito dos recursos oferecidos. Complementando essa documentação, foi gravado um tutorial no YouTube [9], que serve como aula de treinamento para capacitar os alunos no uso dessas ferramentas.

[9] https://youtu.be/zuw0Fo1la2U

2 Fundamentação teórica

Este capítulo apresenta a fundamentação teórica que sustenta o aprimoramento da ferramenta Limarka, abordando conceitos, teorias e práticas relevantes. Serão discutidos aspectos relacionados ao uso de ferramentas no contexto acadêmico, bem como as tecnologias e softwares empregados para otimizar a escrita de documentos acadêmicos. Essa revisão teórica visa contextualizar o aprimoramento da ferramenta Limarka, oferecendo uma base sólida para as melhorias propostas e destacando a importância de cada elemento abordado na evolução do Limarka.

2.1 Ferramentas tradicionais de escrita

Ferramentas como Microsoft Word, Google Docs ou Open Office são amplamente utilizadas para a criação de documentos acadêmicos, mas, segundo Tenen e Wythoff (2022), ao utilizar essas ferramentas, o que aparece na tela nem sempre corresponde exatamente ao resultado final quando o arquivo é salvo. Por trás da *interface* visível de palavras, frases e parágrafos, existe uma camada complexa de código, compreensível apenas pelas máquinas. Essas ferramentas, embora populares, frequentemente apresentam limitações em termos de flexibilidade e controle sobre a formatação final, o que pode resultar em frustrações para os usuários que buscam resultados precisos e personalizáveis.

> Os programas de processamento de texto, como o Word, confundem a composição com a composição tipográfica. Eles funcionam fazendo com que você especifique a aparência de um documento, não como ele está estruturado. Um exemplo clássico são os títulos de seção. Em uma linguagem de marcação típica, você especifica que algo é um título marcando-o como título. Em um programa de processamento de texto, você pode especificar que algo é um título aumentando o tamanho da fonte e colocando-o em negrito. [...] Isso levou a toda uma indústria de pessoas que se especializaram em transformar arquivos do Word em alguma aparência de consistência (SHIEBER, 2014, p. 1, tradução nossa).

A popularidade dessas ferramentas está enraizada na familiaridade para a criação de documentos formatados de maneira visual. De acordo com Cherem (2015), o principal obstáculo para a adoção de uma ferramenta informatizada é a falta de familiaridade com seu uso. A maioria dos usuários de processadores de texto, como Word ou LibreOffice, tende a explorar apenas de forma superficial as capacidades dessas ferramentas. No entanto, essa popularidade muitas vezes ofusca as limitações técnicas que elas impõem. O *design* visual pode induzir erros

de formatação invisíveis, especialmente em documentos complexos, e as correções geralmente requerem uma compreensão de um conjunto de recursos que nem todos os usuários possuem.

Segundo Silva (2024), os alunos enfrentam diversas dificuldades ao elaborar um texto acadêmico. Embora existam disciplinas na grade curricular destinadas a auxiliar nesse processo, a limitação de tempo dessas aulas muitas vezes impede que os estudantes aprendam métodos mais rápidos e eficientes de estruturar um texto. Essa limitação acaba por criar um cenário em que os estudantes, ao não dominarem plenamente as técnicas de estruturação textual, enfrentam desafios adicionais ao longo da elaboração de seus trabalhos acadêmicos.

A dependência de *interfaces* gráficas pode ocultar a estrutura real do documento, levando a inconsistências que só se tornam aparentes quando os documentos são transferidos entre diferentes sistemas ou quando impressos. A necessidade de uma *interface* gráfica sofisticada pode tornar esses programas pesados e menos responsivos, especialmente em dispositivos com hardware mais limitado. Esta complexidade pode se tornar um obstáculo significativo, principalmente para usuários que necessitam de um fluxo de trabalho mais direto e focado no conteúdo.

De acordo Cherem (2015), o uso de classes ou pacotes para recriar práticas idiossincráticas originadas de softwares de processamento de texto "visuais" como Word e LibreOffice muitas vezes resulta em qualidade nitidamente inferior, especialmente no que se refere à legibilidade e à harmonia do texto. Isso reforça o desafio enfrentado pelos usuários desses sistemas, que, além de lidarem com a complexidade da *interface*, muitas vezes obtêm resultados que comprometem a clareza e a eficácia de seus documentos acadêmicos.

> A prática de fazer as coisas por si próprio já se difundiu amplamente há bastante tempo, mas os resultados são frequentemente duvidosos porque os tipógrafos leigos não veem o que está errado e não conseguem perceber o que é importante. Assim, fica-se acostumado a uma tipografia pobre e incorreta. (WILLBERG; FORSSMANN, 1999, p. 48-49, tradução nossa).

Outro desafio é a compatibilidade de arquivos entre diferentes versões e plataformas, que pode causar problemas ao compartilhar documentos entre colaboradores que não utilizam o mesmo software ou versão. O formato proprietário dos arquivos utilizado pelo Microsoft Word, docx por exemplo, pode resultar em problemas de renderização ao ser aberto em versões mais antigas ou em softwares alternativos de processamento de texto. Para Mitolu (2023), uma possível solução para esse problema é a utilização do Markdown, um formato de texto simples e não proprietário, que pode ser usado em qualquer plataforma.

2.1.1 Limitações de ferramentas proprietárias

Ferramentas proprietárias frequentemente utilizam formatos fechados que limitam tanto a acessibilidade quanto a flexibilidade dos documentos. Devido a essa camada oculta, arquivos

como .docx e .pdf dependem de softwares específicos para serem visualizados corretamente. Isso restringe a interoperabilidade entre diferentes sistemas, dificultando a colaboração e a acessibilidade universal dos documentos (TENEN; WYTHOFF, 2022).

Simioni (2021) afirma que o formato docx, padrão do Word, embora amplamente utilizado, apresenta desafios de compatibilidade quando aberto em programas diferentes ou até mesmo em versões distintas do próprio Word. Esses problemas podem se manifestar de diversas formas, desde a visualização incorreta dos documentos até dificuldades de compatibilidade ao abrir arquivos em softwares alternativos.

> Os formatos de arquivo de programas de processamento de texto tendem a mudar. O próprio Word passou por vários formatos de arquivo incompatíveis nas últimas décadas, um a cada dois anos. Com o tempo, você terá que se manter atualizado com a versão mais recente do software para fazer qualquer coisa com um novo documento, mas atualizar o software pode significar que documentos antigos não serão mais renderizados de forma idêntica. Com o Markdown, nenhum software é necessário para ler documentos. Eles são apenas arquivos de texto simples com marcações relativamente intuitivas (SHIEBER, 2014, p. 1, tradução nossa).

As licenças de software também podem restringir o uso e a distribuição de documentos criados com essas ferramentas. Em um contexto acadêmico, onde a disseminação e a colaboração são essenciais, essas restrições podem ser limitantes, especialmente quando se trabalha em projetos de pesquisa com colaboradores de instituições que utilizam diferentes soluções tecnológicas (SILVA; MARTINS; COSTA, 2012).

Outro aspecto crítico é a obsolescência planejada, onde as atualizações de software podem tornar formatos mais antigos de arquivos incompatíveis com novas versões, forçando os usuários a atualizar constantemente seus aplicativos para garantir a continuidade da funcionalidade. Isso não só implica custos adicionais em termos de tempo e recursos, mas também pode resultar em perda de dados se a atualização falhar ou não for totalmente compatível (STROSS, 2013).

Portanto, a dependência de formatos proprietários pode gerar custos ocultos e desafios de interoperabilidade que minam a eficácia das ferramentas tradicionais de escrita. Isso reforça a necessidade de explorar alternativas mais abertas e flexíveis, que não apenas facilitam o fluxo de trabalho acadêmico, mas também proporcionam maior controle e independência tecnológica. Em um dos trabalhos consultados, vimos que a adoção de software livres foi altamente bem-sucedida, influenciando significativamente a filosofia operacional e de implementação da instituição. Os benefícios incluíram não apenas o custo extremamente baixo de aquisição e propriedade, mas também a evolução contínua garantida, com suporte disponível online por meio de fóruns de desenvolvedores e usuários, além do custo reduzido de suporte e manutenção (NILSON et al., 2006).

2.2 Markdown

De acordo com Silveira (2022), o Markdown é uma linguagem de marcação de texto simples que permite destacar elementos como tópicos, links e imagens sem a necessidade de utilizar marcações complexas, como as do LaTeX. Essa simplicidade torna o Markdown uma ferramenta acessível para a organização de conteúdo, mesmo para usuários com pouco conhecimento técnico.

Segundo Logan, Selhorn e Tacker (2022), escrever em Markdown é semelhante a trabalhar com texto simples, o que facilita a visualização das diferenças entre as versões de um arquivo e a identificação de quem fez cada alteração. Em contraste com fluxos de trabalho que envolvem revisões em PDFs, documentos do Word ou Google Docs com comentários, o uso do Markdown torna o processo muito mais eficiente. Isso evita a distribuição de versões desatualizadas de documentos e impede que atualizações inadvertidamente excluam os comentários de outros colaboradores.

Já para Mitolu (2023), o Markdown é uma ferramenta versátil e eficiente para escritores técnicos que desejam produzir conteúdo de alta qualidade. Utilizando tanto a sintaxe básica quanto a avançada, juntamente com atalhos e extensões de teclado, os redatores podem otimizar seu fluxo de trabalho e ganhar tempo. O Markdown permite que você foque na criação do conteúdo, em vez de se preocupar com a formatação, acelerando o processo de escrita e aumentando a produtividade.

Conforme o estudo realizado por Shieber (2014, p. 3, tradução nossa), "o Markdown básico é suficiente para a grande maioria da escrita acadêmica [...] porque as notações de marcação foram projetadas para imitar os tipos de convenções textuais às quais as pessoas estão acostumadas".

2.2.1 Vantagens do Markdown

Segundo Tenen e Wythoff (2022, p. 1), "Markdown pode ser escrito em qualquer editor de texto simples e oferece um rico ecossistema de software que pode renderizar o texto em documentos belamente formatados". Essa simplicidade e versatilidade fazem do Markdown uma escolha atraente para escritores que desejam um controle preciso sobre a formatação de seus documentos, sem a complexidade das ferramentas tradicionais.

Um dos principais benefícios do Markdown é sua capacidade de ser traduzido em uma variedade de formatos de saída, como HTML, PDF e LaTeX, tornando-o extremamente flexível para diferentes necessidades de publicação. Segundo Medeiros (2021), a ferramenta Limarka pode gerar arquivos PDF de documentos acadêmicos, como artigos, dissertações e teses, em conformidade com as Normas da ABNT, a partir de textos escritos em Markdown. Ao utilizar essa ferramenta, os alunos conseguem manter um fluxo de trabalho consistente, independentemente do formato final desejado.

Markdown também promove uma abordagem mais limpa e organizada para a escrita, pois separa o conteúdo da apresentação. Ao focar apenas no texto e nos elementos estruturais, os escritores podem concentrar-se no conteúdo sem distrações causadas por formatação ou estilo visual. Isso é particularmente vantajoso em ambientes acadêmicos, onde a clareza e a concisão do conteúdo são essenciais (MAILUND, 2019).

> O Markdown é uma convenção para estruturar os seus documentos de texto simples semanticamente. A ideia é identificar estruturas lógicas no seu documento (títulos, seções, subseções, notas de rodapé, etc.), marcá-las com caracteres discretos e então "compilar" o texto resultante com um interpretador de composição tipográfica que formatará o documento consistentemente, de acordo com um estilo específico. (TENEN; WYTHOFF, 2022, p. 1)

Segundo Rimal (2021), a simplicidade do Markdown torna-o acessível para uma ampla gama de usuários, desde iniciantes até escritores experientes, sem a necessidade de treinamento extensivo. O aprendizado da sintaxe é intuitivo e direto, permitindo que os usuários comecem rapidamente a escrever e formatar seus documentos de maneira eficaz. Adicionalmente, a natureza baseada em texto do Markdown garante que os documentos sejam pequenos em tamanho e facilmente transportáveis.

Finalmente, o uso de Markdown fomenta a inclusão, permitindo que todos os usuários, independentemente do software disponível ou das habilidades técnicas, possam criar documentos profissionais e bem formatados. Isso democratiza a criação de conteúdo, promovendo a igualdade de acesso à publicação e à disseminação de informações em um ambiente acadêmico global.

2.2.2 Crescimento do uso de Markdown

O uso do Markdown está crescendo, não apenas na escrita acadêmica, mas como uma convenção para edição online em geral. Nas palavras de Krewinkel e Winkler (2017, p. 7, tradução nossa), "devido à sintaxe simples do MD, basicamente qualquer editor de texto é adequado para editar arquivos markdown". Esse crescimento é impulsionado pela sua facilidade de uso e pela capacidade de integração com sistemas de controle de versão como o Git, que são essenciais para projetos colaborativos em ambientes acadêmicos e profissionais.

O Markdown também é um facilitador para desenvolvedores e equipes de projeto que trabalham em ambientes ágeis, onde a documentação precisa ser rápida e de fácil atualização. Sua compatibilidade com sistemas de controle de versão e sua capacidade de ser lido facilmente por humanos tornam-no ideal para documentação de software e relatórios técnicos, onde a precisão e a clareza são fundamentais.

Além de sua popularidade crescente entre escritores individuais, o Markdown está se tornando um padrão de fato em plataformas de publicação online, como blogs, fóruns e *wikis*.

Isso se deve à sua capacidade de ser facilmente convertido em HTML, o formato básico da web, permitindo que os autores publiquem rapidamente conteúdo formatado de forma consistente em sites e outras plataformas digitais. Sendo que conforme Google (2024, tradução nossa) "Markdown é uma linguagem de marcação leve que muitos profissionais técnicos usam para criar e editar documentos técnicos".

A Adobe (2024) afirma que "os artigos técnicos da Adobe são escritos em uma linguagem de marcação simples chamada Markdown, que facilita a leitura e o aprendizado". Além disso, estendeu o Markdown de algumas maneiras para oferecer suporte a determinados recursos relacionados a ajuda, como notas, dicas e vídeos incorporados.

Dodd (2018) afirma que o GitLab destaca o uso do Markdown como uma maneira eficiente de aplicar estilo e formatação ao texto de forma simples e limpa, garantindo que o conteúdo mantenha sua formatação ao ser copiado e colado em diferentes aplicativos. Além disso, o Markdown facilita o processo de revisão, pois permite a visualização do texto em um formato simples, sem a presença de dados ocultos, o que simplifica as comparações entre versões.

No ambiente acadêmico, o uso do Markdown tem se expandido, especialmente quando combinado com plataformas de publicação acadêmica e softwares como o Pandoc [1]. Para Shieber (2014), as extensões do Markdown disponibilizadas pelo Pandoc abrangem quase todas as necessidades para a criação de documentos acadêmicos, incluindo links, referências cruzadas, figuras, citações, bibliografias (via BibTeX), composição tipográfica matemática (via LaTeX), entre outros recursos.

Com o aumento do uso do Markdown, especialmente em contextos acadêmicos e profissionais, sua integração com sistemas de controle de versão como o Git tornou-se fundamental para a criação de documentos colaborativos. Ferramentas como o Manubot [2], que possibilita a autoria colaborativa de manuscritos acadêmicos utilizando Markdown integrado a repositórios Git, ilustram como essas tecnologias estão sendo adotadas para facilitar a escrita e a publicação acadêmica de maneira eficiente e transparente (HIMMELSTEIN et al., 2019).

A integração de sistemas de controle de versão com ferramentas como o Quarto [3] e o R Markdown [4] tem desempenhado um papel crucial na garantia da reprodutibilidade em projetos acadêmicos, um aspecto de crescente importância na pesquisa científica contemporânea (MANG; PROKOSCH; KAPSNER, 2023).

Dessa forma, o uso do Markdown em conjunto com o controle de versão não apenas facilita a colaboração em projetos complexos, mas também aumenta a visibilidade e o reconhecimento das contribuições individuais em trabalhos coletivos, como destacado em debates sobre a maior valorização das contribuições femininas em projetos colaborativos acadêmicos

[1] https://pandoc.org/
[2] https://manubot.org/
[3] https://quarto.org/
[4] https://rmarkdown.rstudio.com/

(MALLETTE; ACKLER, 2018). Essas práticas têm transformado a forma como documentos técnicos e acadêmicos são criados, revisados e publicados, tornando o processo mais acessível, transparente e colaborativo.

2.3 Controle de versão e colaboração

O controle de versão é essencial em projetos de escrita colaborativa, pois permite que múltiplos autores trabalhem simultaneamente sem risco de sobrescrever o trabalho uns dos outros. Cada alteração é registrada com um histórico detalhado, possibilitando que revisões anteriores sejam restauradas caso necessário (LOGAN; SELHORN; TACKER, 2022). Isso é particularmente importante em projetos acadêmicos e de pesquisa, onde a precisão e a integridade dos dados são cruciais.

> O controle de versão, representado por sistemas robustos como o Git, desempenha um papel fundamental. Não se trata apenas de uma característica agradável, mas de um componente essencial no desenvolvimento de software contemporâneo. O versionamento permite que os desenvolvedores trabalhem em conjunto em um único código-fonte, independentemente de sua localização geográfica. (GOMES et al., 2023, p. 2)

O Git é uma ferramenta de controle de versão amplamente utilizada para gerenciar alterações em documentos, especialmente em ambientes colaborativos. Krewinkel e Winkler (2017, p. 9, tradução nossa) afirmam que "os programadores, especialmente quando trabalham em equipes distribuídas, contam com sistemas de controle de versão para gerenciar alterações". Essa prática permite um acompanhamento detalhado de todas as mudanças realizadas, facilitando a identificação de erros e a colaboração entre diferentes membros da equipe.

> O Git permite o trabalho paralelo de colaboradores e possui um sistema eficiente de fusão e resolução de conflitos. Um repositório Git pode ser usado por um único autor local para acompanhar as alterações ou por uma equipe com um repositório remoto, por exemplo, no github [...] o Git pode ser usado para controle de versão e escrita distribuída (KREWINKEL; WINKLER, 2017, p. 9, tradução nossa).

Além de sua aplicação em ambientes colaborativos, o Git desempenha um papel fundamental na escrita acadêmica e em projetos de pesquisa, onde a integridade e a rastreabilidade das alterações são essenciais. Segundo Himmelstein et al. (2019), o Git não apenas facilita a colaboração simultânea em documentos, mas também assegura que cada modificação seja registrada e rastreável, o que é crucial para manter a precisão em trabalhos acadêmicos. Essa

capacidade de gerenciar múltiplos colaboradores trabalhando paralelamente em diferentes seções de um documento sem o risco de perda de dados ou de sobrescrita acidental torna o Git uma ferramenta indispensável para projetos acadêmicos.

Dessa forma, ao utilizar o Git, os autores não só conseguem coordenar suas atividades de forma mais eficiente, mas também garantem que todas as versões do documento sejam preservadas, possibilitando uma revisão histórica detalhada e a restauração de versões anteriores caso necessário.

2.3.1 GitHub

O GitHub [5] é uma das plataformas mais populares para hospedagem de repositórios Git remotos, proporcionando diversas ferramentas que facilitam a colaboração e a revisão de documentos, especialmente os escritos em Markdown. Sua principal funcionalidade é fornecer controle de versão e recursos de colaboração para desenvolvedores de todos os níveis. Dessa forma, ao utilizar as funcionalidades do GitHub, a colaboração pode ser realizada de maneira eficiente e organizada (CARDOSO, 2024).

Ao utilizar o GitHub, equipes de escritores técnicos e acadêmicos podem trabalhar simultaneamente em diferentes seções de um documento, garantindo que todas as alterações sejam rastreadas e facilmente revisadas. Science (2017) afirma que a interface de revisão do GitHub possibilita que qualquer usuário comente sobre mudanças, seja de forma geral ou diretamente em linhas específicas, levantando questões ou pedindo ajustes.

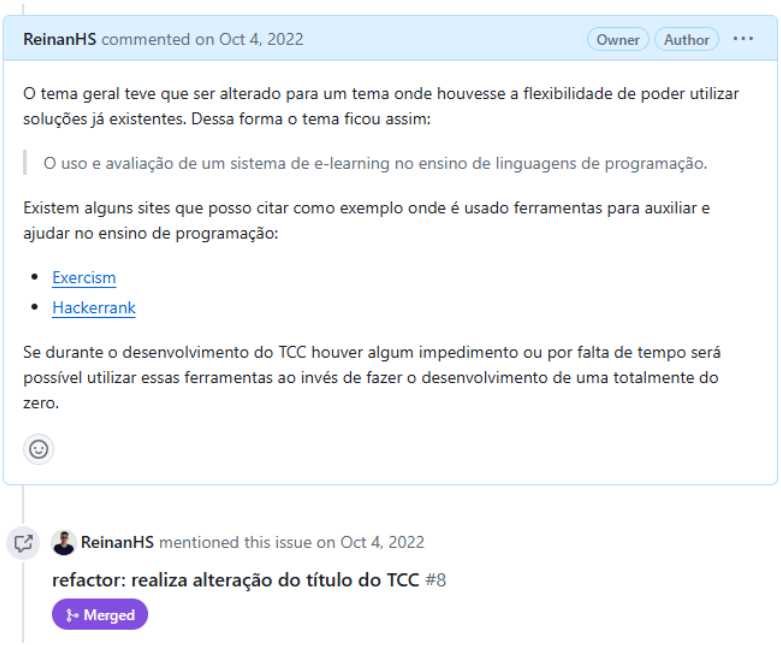
O sistema de *pull requests* do GitHub possibilita que as alterações sugeridas por diferentes colaboradores sejam revisadas e discutidas antes de serem incorporadas ao documento principal, contribuindo para a melhoria da qualidade do conteúdo final. A Figura 1 ilustra um exemplo de discussão entre um aluno e um professor na plataforma do GitHub, relacionada à elaboração de um trabalho acadêmico.

O GitHub oferece uma *interface* visual que simplifica o gerenciamento de versões e a fusão de diferentes contribuições, reduzindo os riscos de conflitos. Ferramentas como o GitHub Actions permitem a automação de tarefas, por exemplo, uma *action* pode clonar o repositório Git do GitHub e configurar automaticamente as ferramentas necessárias para compilar documentos Markdown em formatos finais, como PDF ou HTML (STARLING; SANTOS; GERAIS, 2022). Isso não só melhora a eficiência do fluxo de trabalho, mas também garante que todos os colaboradores estejam trabalhando com as versões mais atualizadas dos documentos.

A plataforma também oferece funcionalidades como a gestão de *issues* e *milestones*, que são particularmente úteis para coordenar a escrita de grandes projetos colaborativos. Por exemplo, *issues* podem ser usadas para atribuir tarefas específicas a membros da equipe, enquanto

[5] https://github.com/

Figura 1 – Debate sobre a escolha do título do TCC via GitHub Issues



Fonte: Autor, (2024).

milestones ajudam a acompanhar o progresso em relação a prazos importantes (MINETTO, 2017).

Essa combinação de ferramentas torna o GitHub uma opção excelente para equipes que procuram um ambiente sólido para colaborar em documentos acadêmicos e técnicos. A Figura 2 demonstra um exemplo de controle de atividades que pode ser gerenciado por meio de *issues* e *milestones* que podem ser organizadas em um quadro Kanban, categorizando as atividades em diferentes fases, como *todo*, *in progress*, e *done*. Essa visualização facilita o acompanhamento do progresso do projeto, permitindo que todos os membros da equipe tenham uma visão clara do que já foi concluído e do que ainda está em andamento, promovendo uma gestão mais eficiente e colaborativa do trabalho acadêmico.

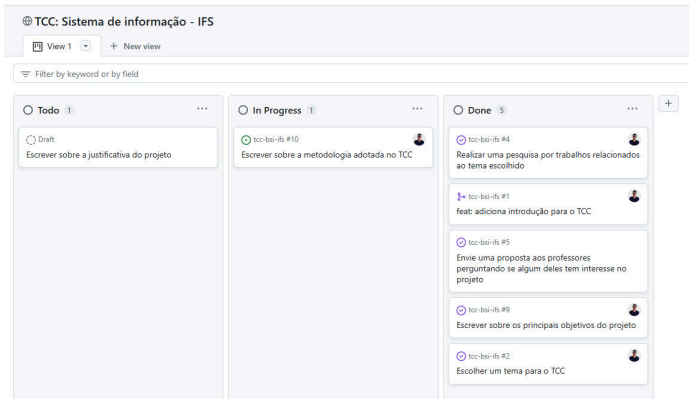
2.3.2 GitLab

Assim como o GitHub, o GitLab [6] é uma plataforma para hospedagem de código, permitindo que desenvolvedores trabalhem em projetos públicos ou privados, sejam eles open-source ou não (SIMIONI, 2023). Para equipes que trabalham na escrita de documentos em Markdown, o GitLab oferece uma série de funcionalidades que facilitam a colaboração e revisão, além de permitir uma personalização mais profunda do fluxo de trabalho.

Uma das principais vantagens do GitLab é sua funcionalidade de *merge requests*, que

[6] https://about.gitlab.com/

Figura 2 – Controle de atividades através do Github



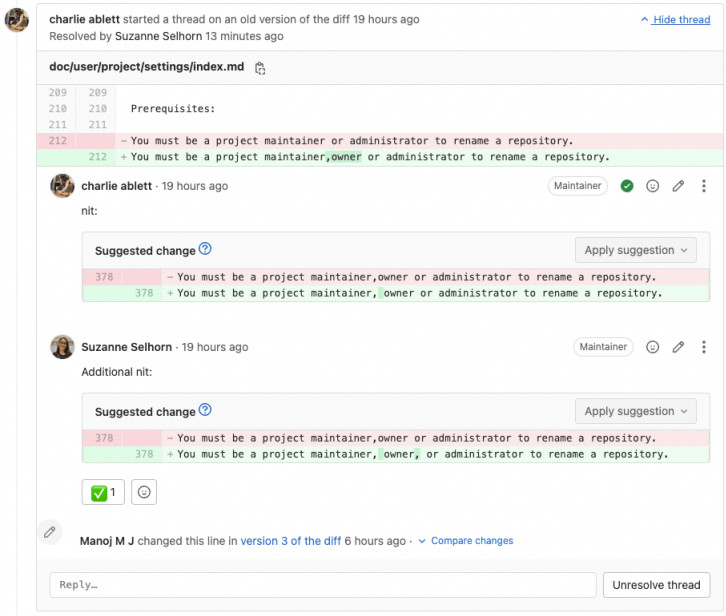
Fonte: Autor, (2024).

opera de forma semelhante aos *pull requests* do GitHub, permitindo que as alterações sejam revisadas e comentadas antes de serem integradas ao repositório principal. Conforme observado por Logan, Selhorn e Tacker (2022), durante uma solicitação de mesclagem, você pode facilmente sugerir mudanças com um simples clique em um botão de Sugestão. É possível comentar em linhas específicas ou em várias, fornecer alterações ou edições, e a pessoa que criou a solicitação pode aplicar a sugestão diretamente ou propor uma alternativa para discussão. Para incluir outros colaboradores na conversa, basta mencionar o nome de usuário deles em um comentário, o que fará com que recebam uma notificação como item de tarefa no GitLab. Dessa forma, qualquer modificação pode ser debatida de maneira transparente e inclusiva. A Figura 3 é um exemplo da colaboração pelo GitLab ilustra como as sugestões de alterações podem ser feitas diretamente em uma *merge request*, permitindo uma interação rica entre os colaboradores.

Essa abordagem assegura que todas as contribuições sejam rigorosamente avaliadas, promovendo a qualidade e a consistência do documento final. Ademais, o GitLab permite a configuração de *pipelines* de CI/CD diretamente no repositório, o que possibilita a automação da compilação de documentos Markdown sempre que uma nova contribuição é incorporada. A compilação do Markdown também pode ser hospedada em páginas do GitLab, tornando-a acessível na internet de forma gratuita (LOGAN; SELHORN; TACKER, 2022).

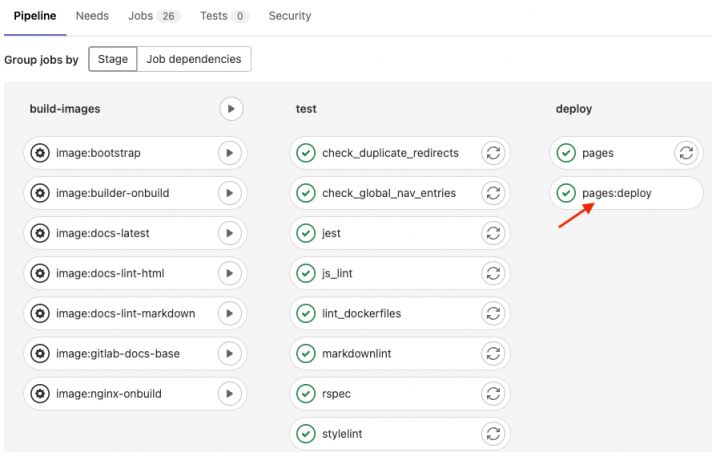
A Figura 4 ilustra o processo de compilação de documentos Markdown por meio do CI/CD no GitLab, destacando as diferentes etapas, como a construção de imagens, testes de validação e finalmente, a implantação nas páginas do GitLab. Cada etapa é executada automaticamente, garantindo que o documento publicado esteja sempre atualizado com as últimas mudanças realizadas no repositório.

Figura 3 – Colaboração na escrita de documentos em Markdown pelo Gitlab



Fonte: Logan, Selhorn e Tacker (2022).

Figura 4 – Pipeline de CI/CD da compilação de documentos Markdown



Fonte: Logan, Selhorn e Tacker (2022).

2.4 Documentação como código integrada ao fluxo de escrita acadêmica

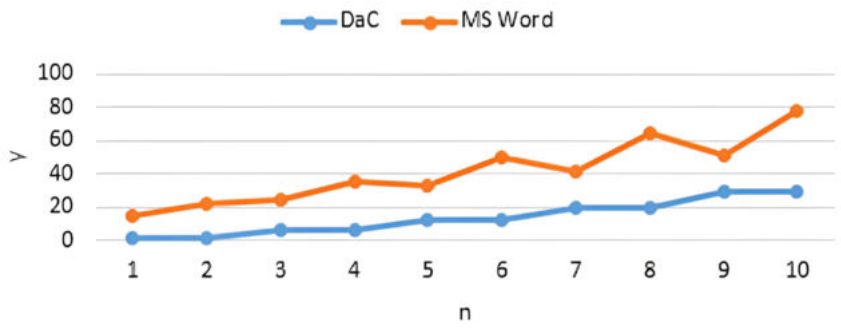
A abordagem de documentação como código DaC envolve a criação da documentação utilizando as mesmas ferramentas empregadas no desenvolvimento de software. Isso inclui monitorar o progresso e gerenciar ideias ou problemas por meio de *issues* em plataformas como Jira, GitLab ou GitHub, além de utilizar controle de versão com Git. A documentação deve ser redigida em Markdown e passar por *pipelines* de validação automatizados via CI/CD, que realizam testes automáticos para garantir sua precisão e consistência (SILVA, 2022). No contexto acadêmico, essa prática pode ser uma poderosa aliada na produção de documentos científicos, como TCCs, dissertações e teses, especialmente quando combinada com ferramentas como Git, *pipelines* CI/CD e Markdown.

No experimento conduzido por Barmina (2023), foi realizada uma comparação entre a eficácia da metodologia do DaC e o uso do Microsoft Office Word na criação de documentos. Os resultados são resumidos de acordo com a Figura 5, onde demonstram que a metodologia DaC apresenta uma curva de complexidade significativamente inferior à do Microsoft Word. Ao longo do eixo X, que representa diferentes etapas ou tarefas do processo de documentação, a abordagem DaC mantém a complexidade em níveis mais baixos, enquanto a utilização do MS Word mostra um aumento constante e acentuado na complexidade conforme as tarefas avançam.

Os valores de y, que poderiam representar indicadores como tempo de execução, número de revisões ou quantidade de erros, são consistentemente mais elevados para o MS Word, sugerindo que essa ferramenta requer mais esforço ou recursos para concluir as mesmas tarefas que são realizadas com maior eficiência pelo DaC. Esse comportamento ressalta as vantagens da abordagem da documentação como código, especialmente em projetos que exigem a colaboração de múltiplos autores ou a repetição de etapas de revisão, onde a simplicidade e a consistência da metodologia DaC permitem um fluxo de trabalho mais eficiente e menos suscetível a complicações. Os resultados do experimento foram sintetizados na Figura 5 para uma visualização comparativa.

Os resultados do experimento de Barmina (2023) indicam que a abordagem DaC foi consideravelmente mais eficaz do que o uso de editores de texto tradicionais como o Microsoft Office Word. A implementação de DaC reduziu a complexidade no processo de criação de documentação em aproximadamente 35%. Portanto, o uso de DaC para a criação de diversos tipos de documentação técnica é recomendado para redatores técnicos, pois acelera o tempo de desenvolvimento e entrega da documentação ao usuário final. No entanto, é importante notar que essa abordagem requer que os redatores tenham um certo nível de treinamento para utilizá-la de forma eficaz.

Figura 5 – Avaliação da eficiência da metodologia Docs-as-Code na redação de documentos



Fonte: Barmina (2023, p. 46).

2.5 Limarka

Desenvolvida por Eduardo de Santana Medeiros Alexandre, a ferramenta Limarka possibilita a redação de trabalhos acadêmicos utilizando Markdown em conformidade com as normas da ABNT. Nos casos em que a sintaxe do Markdown não é suficientemente expressiva para atender a essas normas, o usuário pode inserir código LaTeX, e o Limarka simplifica esse processo. Além disso, a ferramenta é compatível com sistemas operacionais GNU's Not Unix (GNU)/Linux, Windows e macOS, permitindo sua instalação e uso em diversas plataformas (FILHO, 2017).

Segundo Alexandre (2017), a ferramenta do Limarka incorpora funcionalidades avançadas que simplificam a redação e a organização de trabalhos acadêmicos, incluindo o suporte ao uso de Markdown. Adicionalmente, a ferramenta disponibiliza uma ampla gama de recursos que facilitam a criação de documentos complexos, como a gestão de referências bibliográficas, inserção de figuras e tabelas. Essas funcionalidades permitem aos usuários produzir documentos que estejam em conformidade com os mais rigorosos padrões acadêmicos e profissionais, seguindo as normas da ABNT.

> O limarka é uma ferramenta que possibilita o usuário escrever o seu trabalho de conclusão de curso (monografia, dissertação ou teste) em Markdown. Ela converte o trabalho produzido em Markdown para LaTeX, utilizando um template baseado no modelo de trabalho acadêmico do abnTeX2, e produz PDFs em conformidade com as normas da ABNT (ABNTEX, 2019, p. 1).

O Limarka oferece uma *interface* amigável e intuitiva que facilita a navegação e o uso, mesmo para usuários sem experiência técnica. Isso promove a adoção e o uso eficaz da ferramenta, permitindo que todos os usuários, independentemente de suas habilidades ou experiência, possam criar documentos de alta qualidade de maneira eficiente e eficaz.

Segundo Medeiros (2021), uma das principais vantagens do Limarka é que o Markdown é uma linguagem de marcação mais leve e menos complexa que o LaTeX, apresentando uma curva de aprendizado significativamente menor. Apesar de sua simplicidade, o Markdown mantém muitas das vantagens oferecidas pelo LaTeX para a documentação acadêmica, especialmente quando comparado a ferramentas como o Microsoft Word ou LibreOffice Writer.

Conforme Alexandre (2016a), “a ferramenta foi inserida na comunidade dos desenvolvedores do abnTeX, que são comprometidos em produzir documentos e estilos LaTeX para produção de documentos em conformidade com as Normas da ABNT”. O Limarka baseia-se nos modelos desenvolvidos e mantidos pela comunidade. Após o desenvolvimento da ferramenta, a pesquisa avançou para a fase de análise do uso pelos usuários.

3 Trabalhos relacionados

Nesta seção, serão apresentados trabalhos relevantes que abordam a utilização de ferramentas e metodologias baseadas em Markdown para a escrita acadêmica, com foco em automatização de formatação e controle de versão. Esses estudos fornecem uma base importante para o desenvolvimento e a adoção de soluções que simplificam o processo de criação de documentos acadêmicos, garantindo conformidade com normas estabelecidas e promovendo um fluxo de trabalho colaborativo e eficiente.

O trabalho de Alexandre (2017), aborda a criação da ferramenta Limarka, que propõe a escrita de trabalhos de conclusão de curso em conformidade com as normas da ABNT usando a linguagem de marcação Markdown. O artigo consiste tanto no desenvolvimento da ferramenta quanto em um processo conduzido com estudantes da graduação e da pós-graduação que utilizam o Limarka para escrever uma proposta de TCC. A proposta de conceder uma alternativa ao LaTeX, simplificando o processo de escrita devido a uma linguagem de marcação mais acessível, foi pensada para manter o rigor imposto pelas normas acadêmicas brasileiras. Esse estudo é relevante para o meu TCC, pois reforça a viabilidade do uso de linguagens de marcação leve, como Markdown, na produção acadêmica, com o adicional de facilitar a automação e a conformidade com normas técnicas.

No mesmo contexto, o estudo de Oelen e Auer (2019), aborda uma área pouco explorada de acessibilidade na criação de conteúdo educacional. O trabalho dos autores é apresentar uma interface, desenvolvida pelos autores, que permite a produção de apresentações de slides por usuários com deficiência visual, com base em alunos cegos. O trabalho dos autores é apresentar o Markdown como solução, em vez do WYSIWYG para permitir uma produção mais acessível. Os autores ressaltam a importância da acessibilidade na produção de conteúdo, não apenas no consumo, para gerar uma experiência incluída para todos.

No trabalho de Grayson, Hilliker e Wares (2022), os autores consideram a questão do uso do R Markdown como uma ferramenta interativa para ensino em cursos de matemáticas e biologia. No estudo, foi mostrado o uso do R Markdown para criar módulos de ensino que combinem texto formatado com código ao mesmo tempo, o que permitirá aos instrutores não apenas orientar os alunos na tarefa da codificação, mas também verificar se os alunos utilizam corretamente os comandos do código, com a verificação instantânea dos resultados. Esses módulos de ensino proporcionam uma experiência de aprendizado prático, pois permitem que os próprios alunos codifiquem o algoritmo e, em seguida, executam-no diretamente dentro dos próprios documentos, tornando o processo de ensino mais interativo e dinâmico. Mais uma vez, o uso do R Markdown ajudou a desenvolver habilidades de programação, modelagem matemática e visualização de dados e treinou os alunos nesses módulos com a ênfase em republicação. Portanto,

acredito que o artigo ilustre de maneira eficaz como as ferramentas baseadas em Markdown podem ser úteis no ensino.

No artigo feito por Hofert e Kohm (2010), os autores abordam a criação de apresentações científicas com o uso da classe de documento scrartcl do KOMA-Script em LaTeX. O então principal objetivo do trabalho é fornecer oportunidades para que os pesquisadores possam criar apresentações de slides através de documentos LaTeX já existentes, como artigos científicos, de forma fácil e conveniente, sem precisar aprender comandos especializados, copiando e colando após cada uma das operações. Ao mesmo tempo, essa abordagem também fornece aos utilizadores mais flexibilidade na configuração de slides em comparação com os modelos padrões. Essa abordagem permite otimizar significativamente a preparação para a apresentação, sem impactar a qualidade visual, para alunos ou professores interessados em manter uma apresentação universal que pode ser usada em diferentes contextos acadêmicos.

Com base no estudo de Brilhaus et al. (2023), a criação de materiais didáticos abertos com o Markdown ajuda os alunos na elaboração de trabalhos acadêmicos, os conceitos utilizados no estudo têm ligação direta com o tema abordado neste trabalho. A proposta de Brilhaus et al. (2023) de criar arquivos em MD e YAML para melhor organizar e personalizar os materiais educativos ofereceu uma visão interessante de como desenvolver conteúdos reutilizáveis e adaptáveis em diferentes cenários de aprendizado. Dessa forma, a escolha de modularização e reutilização de conteúdo educacional através do Limarka e Marp é coerente com nosso objetivo final.

4 Análise comparativa das ferramentas de escrita acadêmica

A escolha da ferramenta adequada para a escrita acadêmica é crucial para garantir eficiência, precisão e conformidade com as normas exigidas. Com isso em mente, realizamos uma comparação das principais plataformas de escrita colaborativa disponíveis, analisando suas funcionalidades, vantagens e desvantagens em relação ao Limarka. A investigação focou em ferramentas amplamente utilizadas, como Manubot, Authorea, Overleaf, Google Docs, Microsoft Word e LibreOffice Writer.

4.1 Descrição das ferramentas de escrita acadêmica

A seguir, apresentamos uma descrição de cada uma das ferramentas investigadas e comparadas durante o processo de avaliação das plataformas de escrita acadêmica.

4.1.1 Manubot

O Manubot [1] foi escolhido para esta comparação devido à sua abordagem inovadora de escrita colaborativa, baseada em Git e Markdown. Ele é particularmente relevante para equipes que desejam combinar o controle de versão com a facilidade de uso do Markdown, enquanto automatizam aspectos importantes do processo de escrita acadêmica.

Desenvolvida para facilitar a criação de documentos acadêmicos e técnicos, a ferramenta permite que múltiplos colaboradores trabalhem simultaneamente em um projeto, aproveitando o controle de versão para rastrear mudanças e gerenciar contribuições (HIMMELSTEIN et al., 2019). Adicionalmente, o Manubot oferece suporte à criação automatizada de bibliografias e citações, o que facilita a conformidade com as normas acadêmicas.

Entre as vantagens do Manubot, destaca-se a capacidade de automatizar tarefas comuns na escrita acadêmica, como a formatação de citações e referências. Isso não apenas acelera o processo de escrita, mas também reduz o risco de erros humanos. Além do mais, o uso do Markdown simplifica a estruturação do texto, tornando a ferramenta acessível a usuários com diferentes níveis de habilidade técnica.

Por outro lado, o Manubot pode ter uma curva de aprendizado mais elevada para usuários que não estão familiarizados com Git e Markdown. Ademais, a ferramenta não oferece suporte nativo para formatação conforme o padrão da ABNT, o que pode complicar a escrita de trabalhos acadêmicos para os alunos do IFS.

[1] https://manubot.org/

4.1.2 Authorea

O Authorea [2] foi selecionado por sua versatilidade e capacidade de integrar diferentes formatos de texto, como LaTeX e Markdown, em um ambiente colaborativo. É uma ferramenta amplamente utilizada por pesquisadores para a criação de artigos científicos, tornando-se relevante para a comparação.

De acordo com Pepe e Shenk (2023), o Authorea é um editor de documentos colaborativo desenvolvido principalmente para pesquisadores, permitindo a escrita conjunta em tempo real utilizando texto normal, LaTeX e Markdown, tudo dentro do mesmo documento. Além de possibilitar a colaboração eficiente, cada artigo no Authorea funciona como um repositório git, permitindo o armazenamento de dados, figuras interativas e código. A ferramenta também facilita a integração de dados de pesquisa e a criação automatizada de bibliografias, suportando diversos estilos de citação.

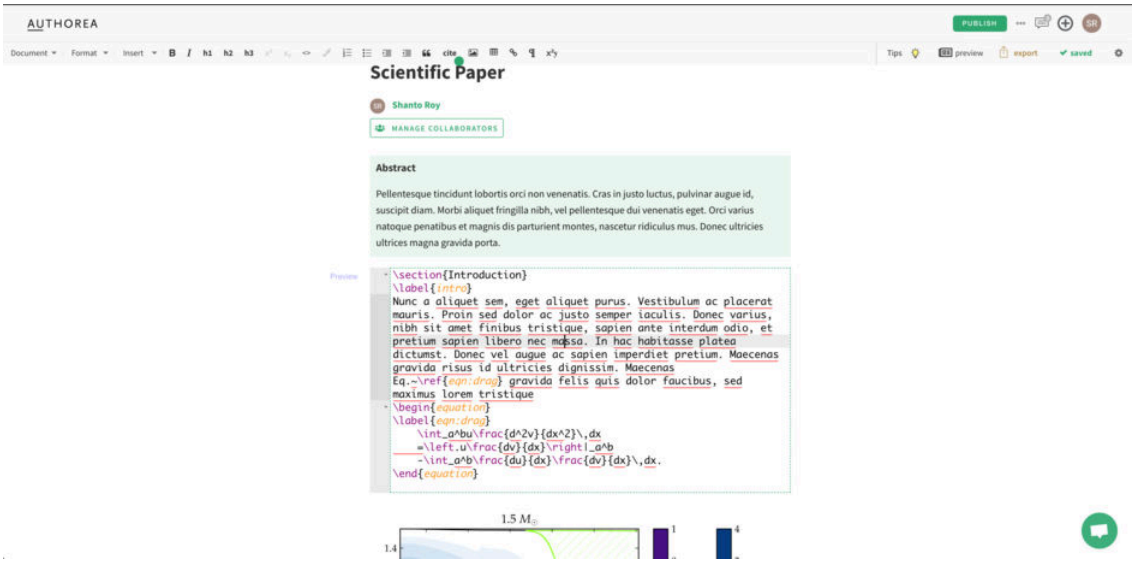
No contexto da escrita acadêmica, o Authorea é amplamente utilizado para a elaboração de artigos científicos, relatórios e documentos técnicos. Sua *interface* intuitiva permite que os usuários trabalhem juntos em tempo real, facilitando a colaboração entre membros de uma equipe de pesquisa. A ferramenta também suporta a revisão por pares e a submissão de manuscritos para publicações científicas, o que a torna uma escolha popular entre pesquisadores. Segundo Roy (2021), a plataforma Authorea é uma novidade que oferece um editor de LaTeX mais acessível. Ela reúne benefícios do Overleaf, Researchgate [3] e ArXiv [4]. Contudo, ainda é um serviço em crescimento, dando seus primeiros passos, com potencial para expandir suas funcionalidades no futuro.

Uma limitação do Authorea é que algumas de suas funcionalidades mais avançadas estão disponíveis apenas para usuários pagos, o que pode ser um desafio para estudantes e pesquisadores com recursos limitados. Outro ponto a ser considerado é que, segundo PUC-SP (2024), a ferramenta não oferece suporte para formatação de texto conforme o padrão ABNT, o que exige que o documento seja exportado para Word para que a formatação seja ajustada.

A Figura 6 apresenta o painel principal do Authorea, mostrando a *interface* gráfica que permite a criação de trabalhos acadêmicos através do editor de LaTeX. Os estudantes podem escrever e publicar seus trabalhos nesta plataforma, que disponibiliza vários recursos, incluindo um editor de fórmulas matemáticas. Adicionalmente, os documentos ficam acessíveis ao público, o que simplifica o compartilhamento das informações contidas no trabalho. A *interface* do Authorea é intuitiva, facilitando a integração de novos usuários que rapidamente se adaptam ao uso da ferramenta, especialmente por ela suportar LaTeX e Markdown.

[2] https://www.authorea.com/
[3] https://www.researchgate.net/
[4] https://arxiv.org/

Figura 6 – A interface do Authorea para redação de trabalhos acadêmicos



Fonte: Roy (2021).

4.1.3 Overleaf

O Overleaf foi incluído na comparação devido ao seu amplo uso na academia, especialmente entre cientistas e engenheiros que precisam de uma ferramenta poderosa para a criação de documentos técnicos em LaTeX. Sua popularidade e suporte a LaTeX fazem dele uma escolha natural para essa análise.

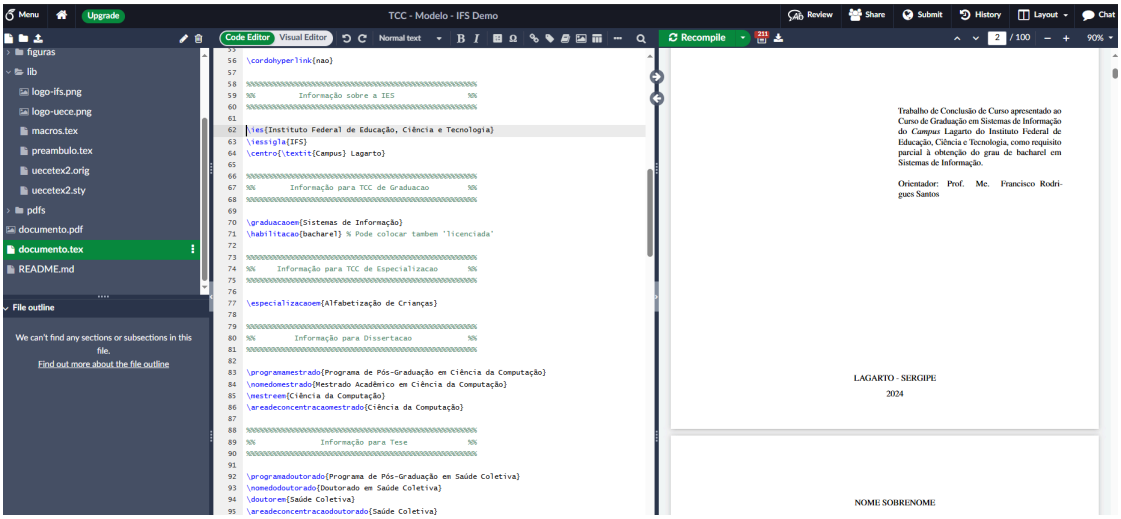
Como aponta Alonso e Berti (2019), o Overleaf é uma plataforma digital gratuita de escrita em LaTeX, amplamente adotada por instituições de pesquisa e ensino em todo o mundo devido à sua praticidade na formatação, acessibilidade e suporte à escrita colaborativa. A ferramenta é especialmente popular entre cientistas por oferecer suporte nativo ao LaTeX, uma linguagem de marcação reconhecida por sua capacidade de gerar documentos complexos e de alta qualidade.

Na escrita de trabalhos acadêmicos, o Overleaf oferece uma solução poderosa para a formatação de textos que exigem precisão, como artigos científicos, teses e dissertações. Sua *interface* permite que os usuários colaborem em tempo real, visualizando as alterações feitas pelos coautores instantaneamente. Além do mais, o Overleaf integra bibliotecas de referência e pacotes de LaTeX, facilitando a inclusão de citações, tabelas e gráficos.

No entanto, conforme destacado por Coelho (2018), a sintaxe do LaTeX pode ser complexa para algumas pessoas, exigindo paciência e dedicação para seu domínio. Embora existam *templates* básicos com a estrutura necessária para a elaboração do documento, o aluno ainda enfrentará uma curva de aprendizado significativa para utilizar essa ferramenta de maneira eficaz.

O Overleaf pode ser desafiador para estudantes que não possuem familiaridade com LaTeX, e alguns dos seus recursos mais avançados estão disponíveis apenas para usuários pagos. A Figura 7 ilustra o painel principal da ferramenta, mostrando um código em LaTeX para a criação da estrutura básica de um documento de TCC.

Figura 7 – A interface do Overleaf para redação de trabalhos acadêmicos



Fonte: Autor, (2024).

4.1.4 Google Docs

O Google Docs foi incluído na análise devido à sua ampla adoção em ambientes educacionais e pela facilidade de colaboração em tempo real que oferece. Sua acessibilidade e simplicidade o tornam uma escolha popular para equipes que buscam uma solução prática e eficiente.

Para Oliveira e Góes (2021), o Google Docs é uma plataforma disponível na web e em dispositivos Android e iOS, que oferece funcionalidades para criar, editar e visualizar documentos de texto, além de permitir o compartilhamento entre usuários. A ferramenta é amplamente utilizada em ambientes acadêmicos, pois possibilita a criação, edição e colaboração em documentos em tempo real.

Na escrita de trabalhos acadêmicos, Google Docs oferece uma solução acessível e prática, especialmente para equipes que precisam colaborar em tempo real. A ferramenta suporta a inclusão de gráficos, tabelas e links, além de oferecer uma gama de opções de formatação e compatibilidade com outros formatos de documento, como .docx e PDF. A integração com Google Drive também facilita o armazenamento e o acesso aos documentos de qualquer dispositivo.

Entretanto, a simplicidade do Google Docs pode ser uma limitação para a criação de documentos acadêmicos mais complexos, como aqueles que exigem uma formatação rigorosa

ou a inclusão de elementos avançados, como fórmulas matemáticas complexas. Para esses casos, ferramentas mais especializadas, como Overleaf ou Limarka, podem ser mais apropriadas.

4.1.5 Microsoft Word

O Microsoft Word foi selecionado devido à sua popularidade e presença dominante como ferramenta de processamento de texto em ambientes acadêmicos e profissionais. Sua vasta gama de recursos o torna uma escolha padrão para muitos usuários.

Pena (2021) afirma que o Microsoft Word é uma das principais plataformas para desenvolvimento de documentos, projetada para criar arquivos profissionais e de alta qualidade, otimizando e organizando projetos de maneira eficaz. Com uma ampla gama de recursos de criação e edição, o Word é amplamente utilizado por profissionais e acadêmicos, sendo a escolha padrão devido à sua familiaridade e funcionalidade abrangente.

No contexto acadêmico, o Microsoft Word é frequentemente utilizado para a criação de artigos, relatórios, dissertações e outros tipos de documentos que exigem formatação específica. O Word suporta a inclusão de gráficos, tabelas, referências bibliográficas e notas de rodapé, além de oferecer uma série de *templates* que facilitam o início da escrita. A integração com o Microsoft OneDrive também permite o armazenamento e a colaboração em documentos na nuvem.

Por outro lado, o Microsoft Word pode apresentar limitações para a criação de documentos técnicos ou científicos que exigem formatação complexa, como a tipografia matemática avançada. Somado a isso, a dependência do formato de arquivo proprietário .docx pode resultar em problemas de compatibilidade quando os documentos são compartilhados entre diferentes plataformas ou softwares.

4.1.6 LibreOffice Writer

O LibreOffice Writer foi incluído na comparação por ser uma alternativa gratuita e de código aberto ao Microsoft Word, amplamente utilizada por pessoas que buscam uma ferramenta acessível e flexível para a criação de documentos.

Segundo LibreOffice (2024), o LibreOffice Writer possui todas as funcionalidades necessárias em uma ferramenta avançada e abrangente de processamento de texto e editoração eletrônica. Ele é simples o bastante para redigir um memorando rápido, mas também é poderoso o suficiente para produzir livros completos com conteúdos, diagramas, índices e outras funcionalidades. Você pode focar na sua mensagem, enquanto o Writer atua como um assistente excepcional.

Uma das principais vantagens do LibreOffice Writer é o fato de ser uma solução gratuita e de código aberto, o que o torna acessível para qualquer pessoa, independentemente de recursos

financeiros. A capacidade de funcionar em múltiplas plataformas, como Windows, macOS e Linux, é outro ponto positivo, permitindo que usuários em diferentes sistemas operacionais utilizem a mesma ferramenta. Além disso, por ser um software de código aberto, o Writer é constantemente atualizado pela comunidade de desenvolvedores, o que garante a sua segurança e a adição de novas funcionalidades.

No entanto, o LibreOffice Writer também apresenta algumas desvantagens. Embora seja uma ferramenta poderosa, sua *interface* pode parecer menos intuitiva em comparação com outros processadores de texto, como o Microsoft Word. Por outro lado, algumas funcionalidades mais avançadas, como a integração com sistemas de controle de versão ou ferramentas de automação de tarefas, podem ser limitadas ou requerer extensões adicionais.

4.2 Comparação das funcionalidades oferecidas

Foi realizada uma investigação para analisar os recursos que diferenciam o Limarka das plataformas de escrita colaborativa existentes. Avaliamos as funcionalidades dessas ferramentas em junho de 2024, utilizando a versão gratuita quando disponível. O objetivo dessa investigação é comparar as capacidades dessas ferramentas e verificar se elas podem auxiliar os alunos do curso de BSI do IFS Campus Lagarto na elaboração de artigos acadêmicos de maneira simples e eficaz.

As funcionalidades selecionadas têm como objetivo simplificar e facilitar o processo de redação de documentos acadêmicos. É importante ressaltar que algumas das plataformas analisadas oferecem recursos adicionais por meio de assinaturas ou versões pagas. Entretanto, nesta análise, foram considerados apenas os recursos disponíveis gratuitamente em cada uma dessas ferramentas.

Ao analisar a Figura 8 de comparação das funcionalidades das ferramentas de escrita acadêmica, observou-se que o Limarka, embora seja uma ferramenta poderosa para formatação automática conforme as normas da ABNT e para gerenciamento de bibliografia, carece de certas funcionalidades presentes em outras ferramentas. Por exemplo, o Limarka não oferece suporte para edição on-line e não possui a funcionalidade de página de compartilhamento integrada, o que pode limitar a colaboração imediata entre os usuários.

Essa ferramenta também não inclui modelos de ferramentas de CI/CD, que são essenciais para automatizar tarefas repetitivas, como compilar documentos ou verificar a formatação. Isso garante que os documentos se mantenham sempre atualizados e consistentes, reduzindo o esforço manual necessário para essas atividades.

Entretanto, essas lacunas identificadas no Limarka são oportunidades de melhoria que serão abordadas neste trabalho. Uma das metas deste projeto é implementar funcionalidades adicionais, como a integração de uma página de compartilhamento e a possibilidade de edição

Figura 8 – Comparação das ferramentas com base nos requisitos funcionais

FUNCIONALIDADE	LIMARKA	MANUBOT	AUTHOREA	OVERLEAF	GOOGLE DOCS	MICROSOFT WORD	LIBREOFFICE WRITER
FERRAMENTAS DE CI/CD	✗	☑	✗	✗	✗	✗	✗
CITAÇÃO POR IDENTIFICADOR	☑	☑	☑	☑	✗	✗	✗
GERENCIAMENTO DE BIBLIOGRAFIA	☑	☑	☑	☑	☑	☑	☑
CONHECIMENTO TÉCNICO NECESSÁRIO	Médio	Médio	Alto	Alto	Baixo	Baixo	Baixo
SUPORTE AO GIT	☑	☑	✗	☑	✗	✗	✗
COMENTÁRIOS EMBUTIDOS	☑	☑	☑	☑	☑	☑	☑
FORMATAÇÃO AUTOMÁTICA PARA ABNT	☑	✗	✗	☑	✗	✗	✗
PÁGINA DE COMPARTILHAMENTO	✗	✗	☑	☑	☑	☑	✗
OPEN-SOURCE SOFTWARE	☑	☑	✗	☑	✗	✗	☑
FORMATO DO DOCUMENTO	Markdown	Markdown	Látex	Látex	DOCX/ODF	DOCX	ODF
EXECUÇÃO ONLINE	✗	✗	☑	☑	☑	☑	✗

Fonte: Autor, (2024).

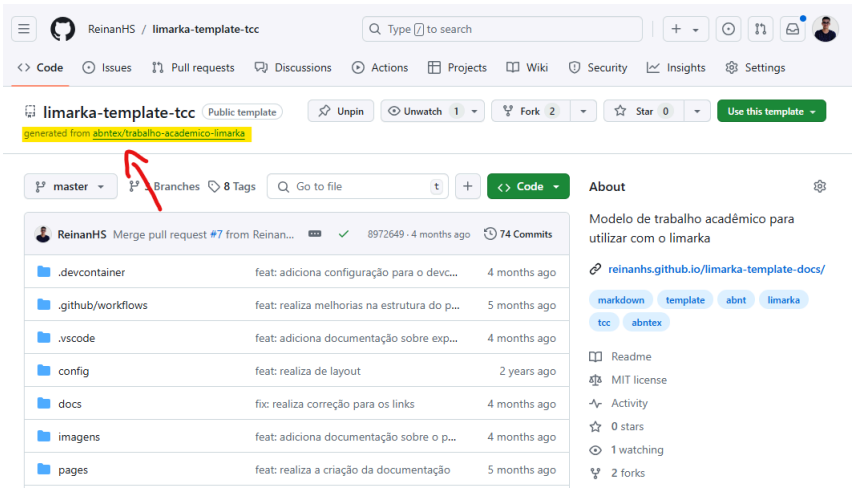
através de um navegador da web, sem a necessidade de instalação local, o que tornará o Limarka uma ferramenta ainda mais competitiva. Com essas melhorias, espera-se que o Limarka se torne uma solução completa e preferida para a escrita acadêmica no contexto dos estudantes do curso de BSI do IFS Campus Lagarto, oferecendo todas as funcionalidades essenciais em uma única plataforma.

5 Processo de fork e reestruturação do repositório Limarka

A ferramenta Limarka, conhecida por sua capacidade de estruturar documentos acadêmicos em conformidade com as normas da ABNT, oferece uma estrutura baseada em Markdown, simplificando o processo de formatação e escrita (FILHO, 2017). Disponível no repositório do Abntex, o Limarka tem sido uma escolha popular entre estudantes que buscam uma abordagem eficiente para a criação de trabalhos acadêmicos.

Ao criar um *fork* do repositório original [1], melhorias na estrutura organizacional e na funcionalidade da ferramenta puderam ser implementadas sem afetar o projeto principal. Como explicado por Github (2024), "em projetos de código aberto, os *forks* são usados com frequência para iterar ideias ou alterações antes que elas sejam incorporadas ao repositório upstream". A Figura 9 ilustra o novo repositório 'limarka-template-tcc' criado a partir do *fork* do Limarka original, destacando as mudanças iniciais feitas para adaptar o modelo às necessidades específicas dos alunos de BSI do IFS Campus Lagarto.

Figura 9 – Ilustração do repositório criado a partir do fork do Limarka



Fonte: Autor, (2024).

Após a criação do novo repositório, renomeamos para limarka-template-tcc [2] para torná-lo mais claro e específico. A nova nomenclatura reflete a ferramenta utilizada, a natureza do repositório como um modelo, e sua aplicação específica para trabalhos de conclusão de curso. Iniciamos as primeiras melhorias no projeto, com uma reestruturação da organização dos ar-

[1] https://github.com/abntex/trabalho-academico-limarka
[2] https://github.com/ReinanHS/limarka-template-tcc

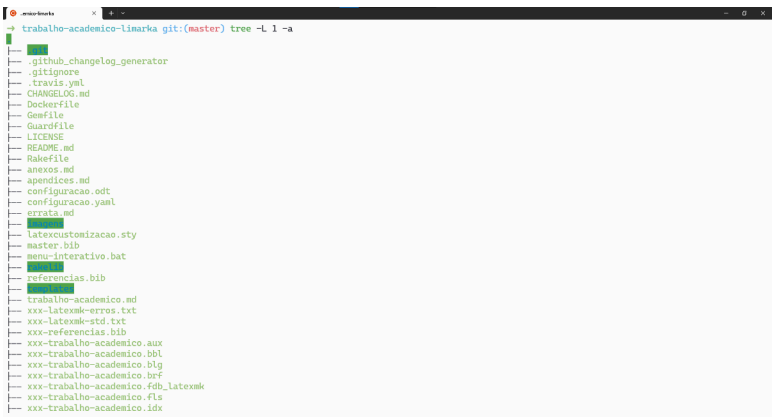
quivos, incluindo a criação de diretórios específicos. Essa reorganização visa alinhar o projeto a padrões modernos, como o uso de diretórios específicos para configuração, documentação, e scripts de automação, adotando convenções amplamente utilizadas em projetos de software *open-source*.

Ghafouri (2021) afirma que a estruturação de pastas em uma solução é um dos primeiros passos essenciais ao iniciar qualquer projeto de software. Ele ressalta que, para garantir uma boa organização e separação de tarefas, é fundamental estar familiarizado com as práticas amplamente utilizadas em projetos de software. Isso não só facilita a compreensão da estrutura do projeto, como também torna o processo de adaptação mais simples para novos desenvolvedores que ingressam na equipe, destacando a importância dessa etapa na implementação de projetos de software.

5.1 Organização da estrutura de pastas

A estrutura de pastas original do modelo feito pelo Alexandre (2016b), embora funcional, apresentava certas limitações em termos de organização e facilidade de uso. No *fork* limarka-template-tcc, a reorganização das pastas foi realizada com o objetivo de tornar o projeto mais intuitivo e acessível para os usuários. Dado que novas pastas foram adicionadas, seguindo padrões amplamente utilizados por diversas plataformas, e outras pastas foram criadas para auxiliar na melhor organização do conteúdo do documento, apresentamos a seguir uma ilustração da estrutura de arquivos e diretórios do modelo original, gerada pelo comando *tree*. Esse comando exibe a hierarquia básica dos arquivos e diretórios presentes no projeto original do Alexandre (2016b), conforme mostrado na Figura 10.

Figura 10 – Ilustração da estrutura de arquivos do projeto original

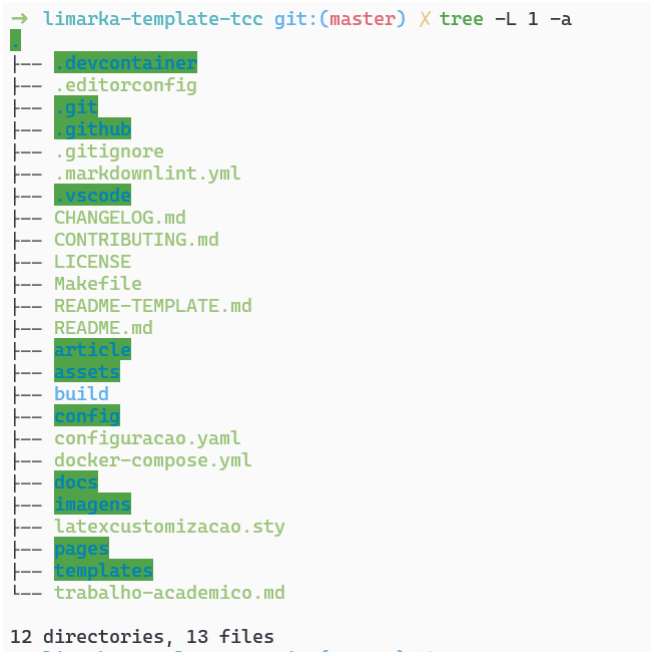


Fonte: Autor, (2024).

Ao analisarmos essa estrutura, identificamos oportunidades para simplificar e aprimorar o modelo, tornando-o mais intuitivo para os alunos do IFS. Na Figura 11, o comando *tree* foi

executado no diretório do novo projeto, que já incorpora as melhorias propostas. Essa visualização permite observar a organização dos arquivos e pastas, destacando diretórios importantes como *.devcontainer*, *article*, *config* entre outros. Esses diretórios foram criados para acomodar os arquivos essenciais do projeto e facilitar a personalização e compilação de documentos acadêmicos no Limarka.

Figura 11 – Ilustração da nova estrutura de arquivos do projeto



Fonte: Autor, (2024).

Essas melhorias refletem uma reestruturação significativa, com a criação de diretórios específicos como *assets*, *build*, *config*, *docs*, e *pages*, que facilitam a navegação e a manutenção do projeto. O novo *layout* é mais organizado, permitindo uma separação clara entre arquivos de configuração, recursos visuais, documentos, e *templates*.

A inclusão de arquivos como *CONTRIBUTING.md* e *.editorconfig* também reflete um compromisso com as boas práticas de desenvolvimento e com a documentação, facilitando futuras contribuições e a utilização do projeto.

> "Um arquivo CONTRIBUTING.md, em seu repositório ou site de software livre, fornece potencial colaboradores do projeto com um pequeno guia de como eles podem ajudar com seu projeto ou grupo de estudo. É convenção colocar a palavra contributing em maiúscula como título do arquivo, e salvá-lo como um recurso em markdown (daí a extensão .md)" (MOZILLA, 2024, tradução nossa).

Graças a essa reorganização, colaboradores e usuários que buscam modificar, personalizar ou contribuir para o projeto encontrarão uma estrutura mais organizada e moderna em comparação a vários repositórios existentes no GitHub. Isso facilita a configuração de diversos recursos dentro do projeto. A Tabela 1 descreve a função de cada diretório na estrutura do projeto.

Tabela 1 – Descrição da estrutura de diretórios.

Diretório	Descrição
.github	Inclui recursos importantes como templates para issues e pull requests, além de guias para contribuição.
.vscode	Armazena configurações recomendadas para usuários do VS Code, como extensões e preferências de formatação.
article	Contém os arquivos Markdown utilizados no documento.
build	Destinado aos outputs do processo de build, como PDFs gerados a partir do Markdown.
config	Contém arquivos de configuração que determinam o comportamento de várias partes do projeto, como parâmetros de compilação.
docs	Hospeda a documentação do projeto, que pode incluir manuais do usuário, especificações técnicas e guias de instalação.
imagens	Repositório para as imagens utilizadas no documento, organizando os recursos visuais.
pages	Contém os arquivos Markdown utilizados para a criação da página web.
templates	Armazena os templates LaTeX que definem a aparência e estrutura do documento final.

Fonte: Autor, (2024).

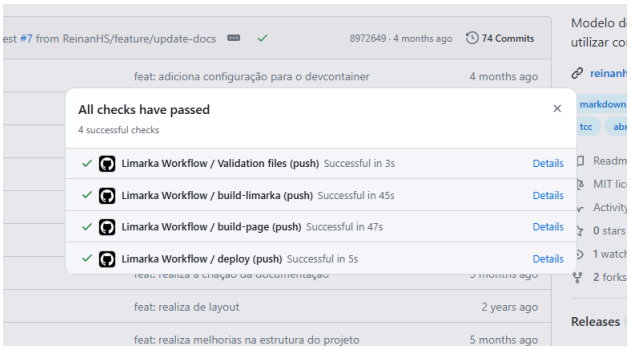
5.2 Diretório .github

O diretório *.github* foi criado com o propósito de configurar o GitHub Actions, a integração com CI/CD é fundamental para a automação de testes, *builds* e *deploys*, garantindo que o documento esteja sempre em um estado funcional e pronto para ser compartilhado. Segundo a RedHat (2023), “uma implementação bem-sucedida de CI significa que, após um desenvolvedor consolidar mudanças no código de uma aplicação, essas atualizações são validadas com a automação da compilação e dos testes em diferentes níveis da aplicação”. Esse diretório também centraliza templates para *issues* e *pull requests*, facilitando a contribuição e a manutenção do projeto por parte dos usuários.

A Figura 12 ilustra as ferramentas de CI/CD que foram integradas ao projeto como resultado da criação deste diretório e das configurações aplicadas dentro dele. A partir dessa imagem,

é possível observar que, sempre que houver uma alteração na *branch* principal do repositório, uma série de validações será iniciada automaticamente para garantir que as mudanças seguem um conjunto de regras predefinidas. Dentro desse fluxo de trabalho, há uma validação denominada “Validation files”, que verifica se os arquivos estão de acordo com a estrutura pré-definida pelo arquivo *.markdownlint.yml*. Segundo a documentação do (ANSON, 2024), esse arquivo define regras que são automaticamente aplicadas durante o fluxo de trabalho, garantindo que todos os arquivos Markdown estejam corretamente formatados.

Figura 12 – Ilustração do fluxo de validação pelo GitHub Actions



Fonte: Autor, (2024).

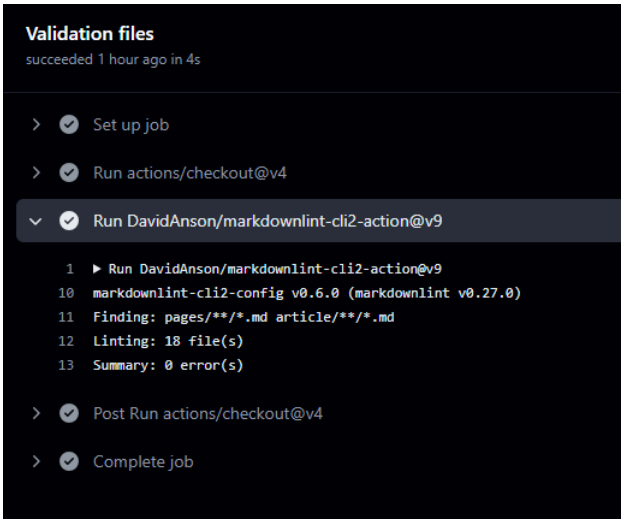
Caso algum arquivo não siga a formatação estipulada por esse arquivo de configuração, a execução do *pipeline* falhará, e a compilação do documento só estará disponível após a correção dos erros indicados. A Figura 13 ilustra a execução dessa validação por meio das ferramentas de CI/CD do GitHub Actions.

Além da validação da estrutura dos arquivos em Markdown, foi desenvolvida uma *job* para automatizar a compilação do documento em PDF utilizando a ferramenta Limarka. Essa *job* desempenha um papel crucial ao permitir que os alunos verifiquem automaticamente se o documento está sendo compilado corretamente. Em caso de erro durante o processo de compilação, o aluno pode visualizar diretamente no painel do GitHub Actions a causa da falha, possibilitando uma correção mais ágil e precisa.

A integração dessa *job* oferece diversas vantagens, incluindo a redução do tempo gasto na verificação manual da compilação e a garantia de que o documento esteja segundo os requisitos estabelecidos. Além dessas vantagens, a *pipeline* pode ser configurada para gerar notificações automáticas por e-mail, alertando o aluno ou orientador sobre a conclusão da compilação ou sobre possíveis problemas, o que contribui para um fluxo de trabalho mais eficiente e colaborativo.

A Figura 14 ilustra a execução dessa *job* no GitHub Actions, destacando como as mensagens de sucesso são exibidas no painel de execução, fornecendo um *feedback* imediato para o usuário. Esse processo automatizado não só simplifica o desenvolvimento do trabalho acadê-

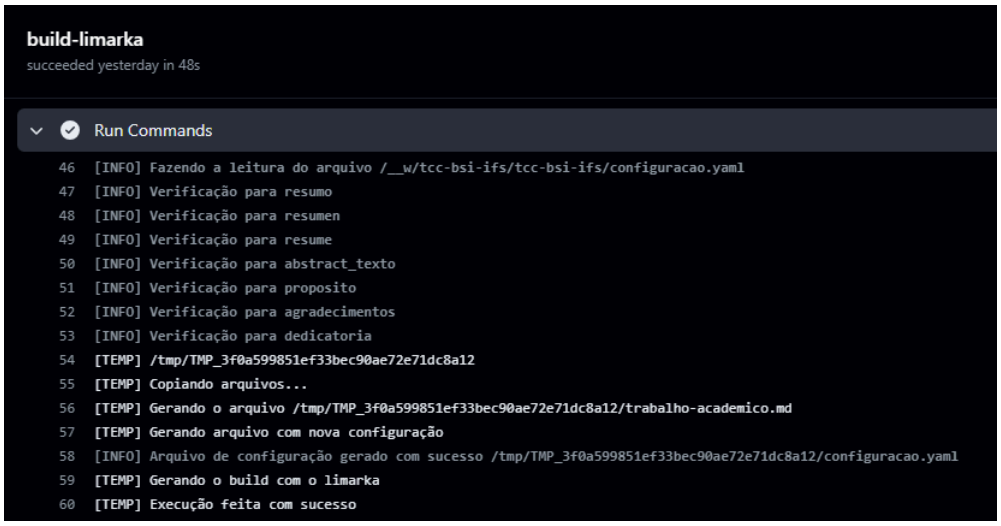
Figura 13 – Ilustração do fluxo de execução da validação do markdown



Fonte: Autor, (2024).

mico, mas também promove uma maior confiabilidade na produção de documentos acadêmicos em conformidade com as normas estabelecidas.

Figura 14 – Execução da compilação em PDF com Limarka



Fonte: Autor, (2024).

Após a compilação do documento em PDF, uma segunda *job* é iniciada, dedicada à criação de uma página web estática utilizando HTML, CSS e JavaScript. Essa página serve como uma plataforma para o compartilhamento do trabalho acadêmico de forma acessível via web. A *job* executa uma ferramenta desenvolvida especificamente para compilar as informações necessárias e realizar a publicação no GitHub Pages, um serviço que permite a hospedagem de

sites estáticos diretamente a partir de um repositório do GitHub. Isso garante que o trabalho acadêmico esteja disponível online, facilitando sua distribuição e acesso por outros pesquisadores, orientadores e colegas.

As configurações estão no arquivo *.github/workflows/limarka.yml*, que desempenha um papel crucial na integração da ferramenta Limarka com as ferramentas de CI/CD. Este arquivo define o fluxo de trabalho automatizado que abrange desde a validação do conteúdo em Markdown até a compilação do documento em PDF e a posterior publicação em uma página web estática via GitHub Pages.

Essa configuração assegura que cada alteração no repositório seja automaticamente processada, garantindo que o documento esteja sempre atualizado e em conformidade com as normas estabelecidas. Isso não só automatiza tarefas repetitivas, como também minimiza erros, promovendo um fluxo de trabalho mais eficiente e confiável para os alunos ao longo do desenvolvimento de seus trabalhos acadêmicos. Essa abordagem também facilita a colaboração, permitindo que todos os envolvidos no projeto acompanhem as mudanças e revisões em tempo real, com validações automáticas para assegurar a qualidade do documento final.

5.3 Diretório .vscode

A criação do diretório *.vscode* foi implementada para otimizar o uso do Visual Studio Code (VS Code) no contexto da escrita acadêmica, especialmente no que diz respeito às melhorias propostas para o projeto Limarka. O VS Code é amplamente reconhecido por sua *interface* amigável e altamente personalizável, o que o torna a ferramenta recomendada para os alunos utilizarem durante a redação de seus trabalhos acadêmicos. Essa escolha se deve à sua capacidade de atender tanto a iniciantes quanto a usuários avançados, proporcionando uma experiência de escrita fluida e eficiente.

Conforme destacado por Providello (2023), uma das grandes vantagens do VS Code é sua *interface* intuitiva, que permite até mesmo aos usuários novatos se sentirem confortáveis desde o início. Essa simplicidade, aliada à vasta gama de extensões e opções de personalização, permite que os alunos adaptem o ambiente de trabalho às suas necessidades específicas, aumentando a produtividade e a eficiência. A criação do diretório .vscode possibilita configurar o VS Code de maneira a atender às demandas específicas da escrita acadêmica, integrando ferramentas que facilitam o processo de redação e revisão de textos.

5.3.1 Configuração de extensões

O arquivo de configuração *.vscode/extensions.json* foi criado para recomendar um conjunto de extensões que otimizam o ambiente de desenvolvimento no VS Code para a escrita acadêmica. A Figura 15 lista as extensões recomendadas, acompanhadas de suas respectivas funcionalidades.

Figura 15 – Listagem das recomendações de extensões para o VSCode

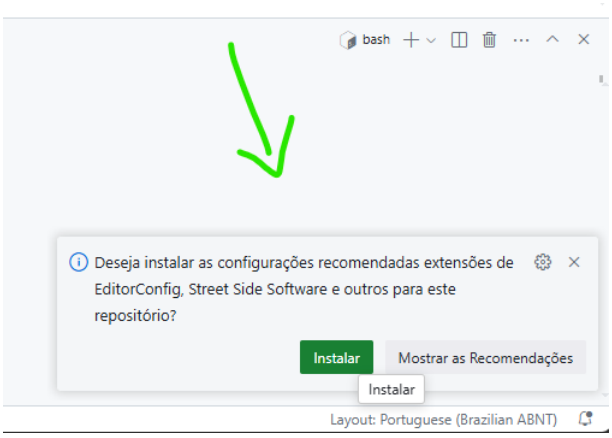
Extensão	Descrição
Markdown Snippets for MDX	Fornece snippets de código para Markdown, acelerando a criação de estruturas de documentos.
EditorConfig	Mantém estilos de codificação consistentes entre diferentes editores e IDEs.
Markdown Table Formatter	Facilita a criação e formatação de tabelas em Markdown.
Brazilian Portuguese	Verifica a ortografia em português brasileiro, garantindo a correção do texto acadêmico.
Live Share	Permite a colaboração em tempo real dentro do VS Code, útil para revisões e coautoria.
Marp	Permite a criação de um conjunto de slides escrito em Markdown no VS Code.

Fonte: Autor, (2024).

Essas extensões ajudam os alunos a manter a consistência e a qualidade na escrita acadêmica. Por exemplo, a extensão de verificação ortográfica para português é essencial para assegurar que o texto esteja livre de erros gramaticais, enquanto a extensão de formatação de tabelas simplifica a criação de tabelas bem organizadas e esteticamente agradáveis. A integração com o Live Share permite que os alunos colaborem em tempo real, tornando o processo de coautoria e revisão mais eficiente.

Quando o usuário abre o editor, essas configurações são automaticamente importadas, e as extensões recomendadas são sugeridas para instalação. A Figura 16 demonstra como essas sugestões são apresentadas aos alunos ao abrirem o projeto no VS Code.

Figura 16 – Exemplificação da recomendação de instalação de extensões no VSCode



Fonte: Autor, (2024).

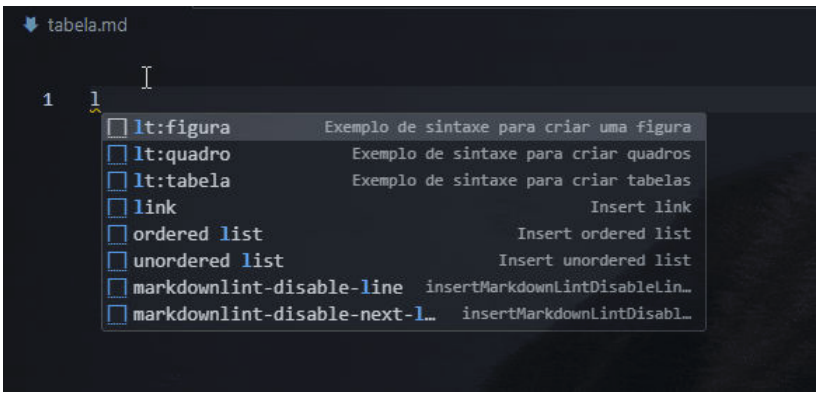
5.3.2 Configuração de *snippets*

Snippets são blocos de código predefinidos que podem ser inseridos automaticamente no texto, economizando tempo e garantindo a consistência na formatação. Para auxiliar os alunos, foi criado o arquivo *.vscode/markdown.code-snippets*, que oferece snippets específicos para a escrita acadêmica em formato Markdown, facilitando a criação de quadros, tabelas e figuras de forma rápida e padronizada.

Segundo Cubos Academy (2023), "snippets são pequenos trechos de código pré-definidos que podem ser inseridos rapidamente em seu editor de texto. Eles são uma forma de automação que economiza tempo e ajuda a evitar erros comuns". Os snippets configurados neste arquivo ajudam os alunos a inserir rapidamente estruturas comuns em seus documentos, como tabelas e quadros, sem a necessidade de digitar todo o código manualmente.

Por exemplo, ao digitar um prefixo como *lt:quadro*, o aluno pode gerar automaticamente o código necessário para criar um quadro com a formatação correta. Isso é particularmente útil em documentos técnicos que exigem uma estrutura rigorosa e precisa. A Figura 17 ilustra a utilização de um snippet.

Figura 17 – Exemplo de sintaxe para criar uma figura



Fonte: Autor, (2024).

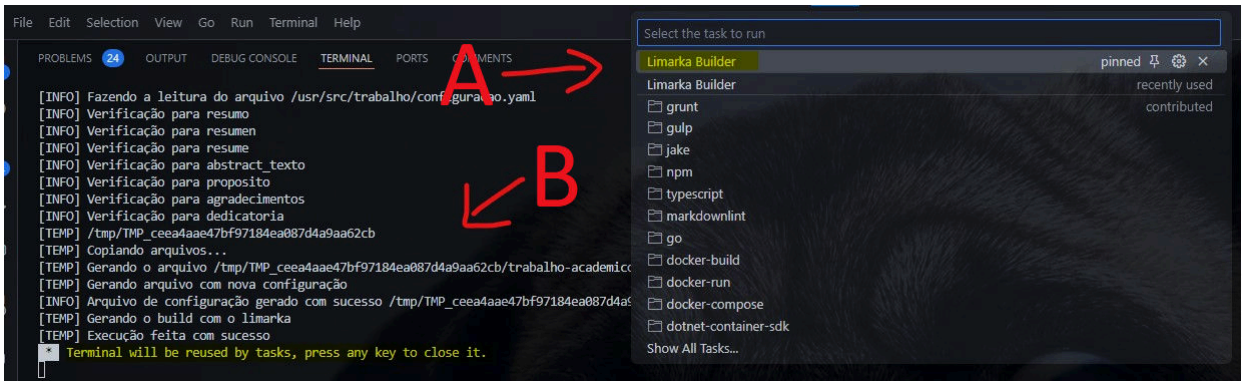
5.3.3 Configuração de *tasks*

O arquivo *.vscode/tasks.json* foi criado para definir tarefas automatizadas que facilitam o processo de compilação do documento no VS Code. Essas *tasks* permitem que os alunos compilem seus documentos Markdown em PDF com um simples comando, sem a necessidade de alternar entre o VS Code e o terminal. Segundo Visual Studio (2024), as tarefas no VS Code podem ser personalizadas para executar scripts e iniciar processos, permitindo o uso de várias ferramentas existentes diretamente dentro do VS Code, sem a necessidade de digitar comandos ou escrever código adicional.

Essa automação não só economiza tempo, mas também minimiza a possibilidade de erros durante o processo de compilação, proporcionando uma experiência de desenvolvimento mais integrada e eficiente.

A Figura 18 ilustra um exemplo de utilização de uma *task* no VS Code. Ao acessar o menu de *tasks*, os alunos encontrarão uma opção denominada "Limarka Builder" (identificada pela seta A da Figura 18) para realizar o build do documento. Ao selecionar essa opção, a compilação do trabalho será iniciada automaticamente no terminal, essa opção está sendo ilustrado pela seta B da Figura 18. Essa execução é facilitada por um script localizado na pasta *.vscode*, que foi desenvolvido em duas versões: uma para execução em Linux, utilizando Shell Script, e outra no formato *.bat*, para execução em sistemas Windows.

Figura 18 – Exemplo de execução de uma task pelo VS Code



Fonte: Autor, (2024).

5.4 Diretório *article*

Segundo a documentação da ferramenta Limarka, os usuários desenvolvem todo o seu trabalho acadêmico em um único arquivo chamado *trabalho-academico.md* (ALEXANDRE, 2017a). Embora essa abordagem seja funcional, ela apresenta desafios significativos à medida que o documento aumenta em extensão e complexidade. Concentrar todo o conteúdo em um único arquivo torna a navegação, edição e organização do texto cada vez mais difíceis, comprometendo a eficiência do processo de escrita e revisão. Gerenciar um arquivo extenso pode dificultar a localização de seções específicas, o que pode levar a erros e aumentar o tempo necessário para revisões.

Segundo Bolton (2023), trabalhar em um projeto utilizando múltiplos arquivos é frequentemente a melhor abordagem. Se o projeto for compilado, o compilador necessitará recompilar apenas os arquivos que foram alterados, ao invés de todo o projeto. Editores de texto

tendem a não funcionar bem com arquivos extremamente longos. A utilização de vários arquivos promove modularidade e além disso, editar arquivos menores é mais fácil e ágil.

A separação do conteúdo em pequenos arquivos traz diversos benefícios importantes. Em primeiro lugar, permite uma organização mais clara e modular do documento, onde cada seção ou capítulo pode ser editado e revisado de forma independente. Isso é particularmente vantajoso em contextos acadêmicos, onde diferentes partes do trabalho podem ser revisadas por diferentes orientadores ou membros da equipe. Além disso, a modularização ajuda a minimizar os riscos de erros, evitando a manipulação simultânea de grandes blocos de texto. Essa prática também facilita a reutilização de partes do texto em outros contextos ou trabalhos, bem como a integração de novos conteúdos sem comprometer a estrutura existente.

Com base nessa necessidade de modularização, foi proposta a criação do diretório *article* para armazenar as diferentes partes do trabalho acadêmico em pequenos arquivos Markdown. Essa abordagem facilita a revisão pelos orientadores e melhora a organização e manutenção do documento ao longo do tempo. Para Monutti (2024), a modularidade de código é uma técnica utilizada para organizar e aprimorar a legibilidade do código, dividindo-o em módulos menores. Isso não só facilita a leitura, tornando-a mais clara e direta, mas também simplifica a manutenção do código.

Em ambientes que utilizam sistemas de controle de versão como Git, cada alteração pode ser claramente identificada, revisada e, se necessário, revertida sem impactar o restante do documento. Essa metodologia também promove uma maior clareza na atribuição de responsabilidades entre os autores, permitindo que cada membro da equipe se concentre em partes específicas do trabalho, garantindo que o produto final seja coeso e bem organizado.

Além disso, Ao dividir o conteúdo em partes menores e mais gerenciáveis, os autores podem refinar cada seção de forma mais detalhada, assegurando que todos os argumentos sejam apresentados de forma lógica e estruturada. Isso não só melhora a qualidade geral do documento, mas também torna o processo de revisão mais eficiente, permitindo que os revisores se concentrem em seções específicas sem a necessidade de navegar por um documento extenso e potencialmente desorganizado.

A Figura 19 ilustra a utilização desse diretório para separar o documento em módulos. Como mostrado na imagem, o diretório permite que o aluno divida seu trabalho em pequenos arquivos Markdown, que posteriormente serão unificados no arquivo *trabalho-academico.md* por meio de uma importação desses módulos. Essa funcionalidade de importação de módulos é uma melhoria adicional implementada na ferramenta Limarka para suportar essa abordagem modular, contribuindo para uma maior eficiência e flexibilidade na escrita acadêmica.

Figura 19 – Exemplo para a utilização do diretório article

Fonte: Autor, (2024).

5.5 Diretório *build*

Conforme a documentação da ferramenta Limarka, os arquivos de compilação são gerados diretamente no diretório raiz do projeto (ALEXANDRE, 2017b), o que resultava em uma desorganização significativa. Essa prática não apenas dificultava a visualização clara dos arquivos principais do projeto, mas também complicava o gerenciamento desses artefatos em sistemas de controle de versão e nas pipelines de CI/CD. A mistura de arquivos de *build* com outros arquivos essenciais do projeto tornava o processo de manutenção e desenvolvimento mais propenso a erros e mais difícil de gerenciar.

Reconhecendo essa necessidade, foi implementado um diretório exclusivo denominado *build*, destinado a armazenar todos os artefatos gerados durante o processo de compilação. Com essa organização, os arquivos de *build* foram segregados dos arquivos principais do projeto, resultando em uma estrutura mais limpa e gerenciável. Além de melhorar a clareza e a organização do projeto, essa separação permite que o diretório *build* seja facilmente ignorado pelos sistemas de controle de versão, evitando o versionamento de arquivos temporários e focando apenas nos componentes essenciais do projeto.

A Figura 20 ilustra o novo *layout* dos arquivos de compilação, agora centralizados dentro do diretório *build*. Essa abordagem oferece várias vantagens, como a simplificação do processo de limpeza, além de facilitar a integração com ferramentas de CI/CD, que podem agora focar diretamente no diretório *build* para realizar tarefas automatizadas, como a geração de PDFs e outros artefatos de saída. Em última análise, essa reestruturação promove uma manutenção mais eficiente e um fluxo de trabalho mais robusto, alinhado com as melhores práticas de

desenvolvimento de software.

Figura 20 – Exemplo para a utilização do diretório build

```
→ build git:(feature/structure-update) tree
.
├── xxx-latexmk-erros.txt
├── xxx-latexmk-std.txt
├── xxx-referencias.bib
├── xxx-texliveonfly-std.txt
├── xxx-trabalho-academico.aux
├── xxx-trabalho-academico.bbl
├── xxx-trabalho-academico.blg
├── xxx-trabalho-academico.brf
├── xxx-trabalho-academico.fdb_latexmk
├── xxx-trabalho-academico.fls
├── xxx-trabalho-academico.idx
├── xxx-trabalho-academico.ilg
├── xxx-trabalho-academico.ind
├── xxx-trabalho-academico.lof
├── xxx-trabalho-academico.log
├── xxx-trabalho-academico.pdf
├── xxx-trabalho-academico.tex
├── xxx-trabalho-academico.toc
└── xxx-trabalho-academico.xdv

0 directories, 19 files
```

Fonte: Autor, (2024).

5.6 Diretório .devcontainer

Para facilitar a utilização da ferramenta Limarka pelos alunos, foi implementado o diretório *.devcontainer*. Inicialmente, essa configuração não existia, e sua implementação foi necessária para proporcionar um ambiente de desenvolvimento padronizado e acessível, utilizando contêineres. O diretório *.devcontainer* é utilizado para especificar as configurações e requisitos necessários para rodar a ferramenta Limarka em um ambiente isolado, evitando a necessidade de instalação manual e configuração em cada máquina dos alunos.

Com a criação deste diretório e as configurações associadas a ele, os alunos podem executar a ferramenta Limarka diretamente através de um contêiner, sem precisar instalar o Limarka localmente. Isso é particularmente útil para evitar problemas de compatibilidade e para garantir que todos os alunos tenham acesso ao mesmo ambiente de desenvolvimento, independentemente de suas configurações locais. Os contêineres de desenvolvimento podem ser executados tanto localmente quanto remotamente, em nuvens privadas ou públicas, oferecendo flexibilidade e facilidade de uso.

De acordo com a documentação do Github (2023), “os arquivos de configuração para um contêiner de desenvolvimento estão contidos em um diretório .devcontainer em seu repositório. Você pode usar VS Code para adicionar arquivos de configuração para você”. É possível definir uma configuração única de contêiner de desenvolvimento para todo o repositório, configurações

distintas para diferentes *branches* ou múltiplas configurações. Quando há várias configurações disponíveis, os usuários têm a liberdade de escolher a que preferirem.

> "A especificação do contêiner de desenvolvimento procura encontrar maneiras de enriquecer os formatos existentes com configurações, ferramentas e configurações específicas de desenvolvimento, ao mesmo tempo em que fornece uma configuração simplificada e opção de contêiner único não orquestrado – para que possam ser usados como ambientes de codificação ou para integração e testes contínuos. Além dos metadados principais da especificação, a especificação também permite desenvolvedores para compartilhar e reutilizar rapidamente as etapas de configuração do contêiner por meio de recursos e modelos." (MICROSOFT, 2024, tradução nossa).

O arquivo *devcontainer.json* dentro do diretório *.devcontainer* foi configurado para definir as especificações do ambiente de contêiner que será utilizado pelos alunos. A configuração proposta é ilustrada na Figura 21. Nessa configuração, a imagem do contêiner utilizada é *reinanhs/limarka-help:latest*, que contém todas as dependências e configurações necessárias para rodar a ferramenta Limarka. Ao utilizar essa imagem, os alunos têm à disposição um ambiente pronto e otimizado para o desenvolvimento de trabalhos acadêmicos, sem a necessidade de instalar manualmente as bibliotecas ou ferramentas adicionais. Isso simplifica significativamente o processo de configuração, permitindo que os alunos se concentrem na produção de conteúdo acadêmico.

Figura 21 – Exemplo da configuração para o devcontainer

```
{
  "name": "limarka-template-tcc",
  "image": "reinanhs/limarka-help:1.0.0",
  "features": {
    "ghcr.io/devcontainers-contrib/features/zsh-plugins:0": {}
  }
}
```

Fonte: Autor, (2024).

A configuração também inclui a adição de funcionalidades extras por meio do recurso *features*, como *plugins* Zsh, que aprimoram a experiência do usuário dentro do terminal do contêiner. Esses *plugins* podem incluir melhorias na autocompletação, sugestões de comandos e outras funcionalidades que facilitam a interação com o ambiente de desenvolvimento.

A implementação do diretório *.devcontainer* é uma iniciativa que atende às necessidades dos estudantes, permitindo o uso da ferramenta diretamente pelo navegador via GitHub Codespaces, oferecendo uma solução robusta e eficiente para o uso do Limarka em diferentes ambientes de desenvolvimento.

5.7 Diretório pages

A criação do diretório *pages* foi fundamental para o desenvolvimento da ferramenta limarka-render-html, que trouxe uma série de benefícios significativos ao processo de compilação e compartilhamento de trabalhos acadêmicos. Ao organizar as configurações e o conteúdo em arquivos Markdown separados dentro do diretório *pages*, a ferramenta proporcionou uma forma modular e organizada de gerenciar as informações do projeto.

A utilização do *limarka-render-html* para gerar uma página estática em HTML representa um avanço importante na maneira como os alunos podem compartilhar seus trabalhos acadêmicos. Segundo a cloudflare (2024), "um site estático é composto de uma ou mais páginas HTML que são carregadas sempre da mesma maneira [...] as páginas web estáticas são arquivos HTML simples que podem ser carregados rapidamente".

Com essa ferramenta, os estudantes podem disponibilizar seus documentos em um formato acessível e visualmente organizado, permitindo que orientadores e outros interessados acessem o trabalho diretamente pela web, sem a necessidade de downloads ou de software específico para visualização.

Outro benefício significativo é a possibilidade de integração com ferramentas de CI/CD, que automatizam o processo de geração e atualização dessas páginas web. Fazer uso da implementação CD significa que as mudanças feitas por desenvolvedores em uma aplicação na nuvem podem entrar em vigor em questão de segundos, caso passem em todos os testes automatizados testes automatizados, como explica a RedHat (2023). Essa agilidade facilita o recebimento e a integração constante do *feedback* dos usuários.

Isso garante, ainda, que qualquer alteração realizada no conteúdo ou nas configurações seja refletida automaticamente na versão online do documento, mantendo-o sempre atualizado e disponível para consulta. Adicionalmente, a inclusão de funcionalidades como compartilhamento em redes sociais e a adição de caixas de comentários tornam a interação com o conteúdo mais dinâmica e colaborativa, permitindo *feedback* instantâneo e a disseminação do trabalho em um âmbito mais amplo.

Assim, o diretório pages permite organizar os arquivos específicos do projeto, como por exemplo o arquivo de resumo e abstract, e servir como local base na compilação das páginas que serão publicadas em um site estático no GitHub Pages. A Figura 22 mostra a página estática criada por esta ferramenta no formato HTML, facilitando a publicação na internet e o acesso simplificado.

A flexibilidade proporcionada pela ferramenta *limarka-render-html* permite, também, que os que os alunos adaptem a estrutura da página conforme suas necessidades, seja para fins de apresentação ou para melhorar a navegabilidade do conteúdo. Essa modularidade e capacidade de personalização fazem do *limarka-render-html* uma ferramenta poderosa e versátil no apoio à produção e divulgação de trabalhos acadêmicos.

Figura 22 – Exemplificação da página em HTML gerada pela ferramenta

Fonte: Autor, (2024).

5.8 Diretório docs

A criação do diretório *docs* foi motivada pela necessidade de implementar a filosofia de documentação como código, que integra a produção de documentação aos mesmos processos e ferramentas utilizados no desenvolvimento do código. Essa abordagem permite que o conteúdo evolua em sincronia com o código, garantindo que esteja sempre atualizada e em conformidade com as melhorias propostas durante o desenvolvimento do template do Limarka. Segundo Logan, Selhorn e Tacker (2022), documentação como código é um método para desenvolver e publicar instruções de produtos. Utiliza as mesmas ferramentas e processos que o desenvolvimento de software, armazenando os arquivos de documentação junto aos arquivos de código em um repositório de controle de versão.

Armazenar toda a documentação relacionada ao uso e à configuração do template do Limarka, incluindo as novas funcionalidades e melhorias implementadas neste trabalho. Nele, estão organizadas instruções detalhadas, tutoriais e guias que auxiliam os usuários a compreender e utilizar as novas funcionalidades de maneira eficiente e eficaz. Ao centralizar a documentação neste diretório, asseguramos que todas as informações necessárias para o uso do template estejam acessíveis e estruturadas de forma lógica e coerente.

Além de servir como repositório central de material de referência, o diretório *docs* desempenha um papel crucial na automação da criação de conteúdo documental. Os arquivos contidos neste diretório são processados pela ferramenta Docusaurus [3], que converte o conteúdo

[3] https://docusaurus.io/

em uma documentação interativa e navegável. Segundo Palmowski (2023), docusaurus é um gerador de sites estáticos renomado que emprega React, uma das principais bibliotecas JavaScript, para a *interface* do usuário (UI) na criação de páginas. Fácil de configurar e personalizar, oferece todos os recursos necessários para desenvolver seu site estático. Foco no conteúdo, Docusaurus é ideal para construir sites de instruções, como *wikis*, e utiliza markdown, permitindo a compatibilidade para colaboração e armazenamento em repositórios git.

A documentação gerada apresenta todos os aspectos necessários para a utilização do template Limarka, desde a instalação inicial até a personalização avançada, e é mantida sempre atualizada conforme novas melhorias e correções são incorporadas ao projeto. Dessa forma, o diretório *docs* não apenas suporta a filosofia de DaC, mas também garante que a documentação seja uma fonte confiável e acessível de informações para todos os usuários do Limarka.

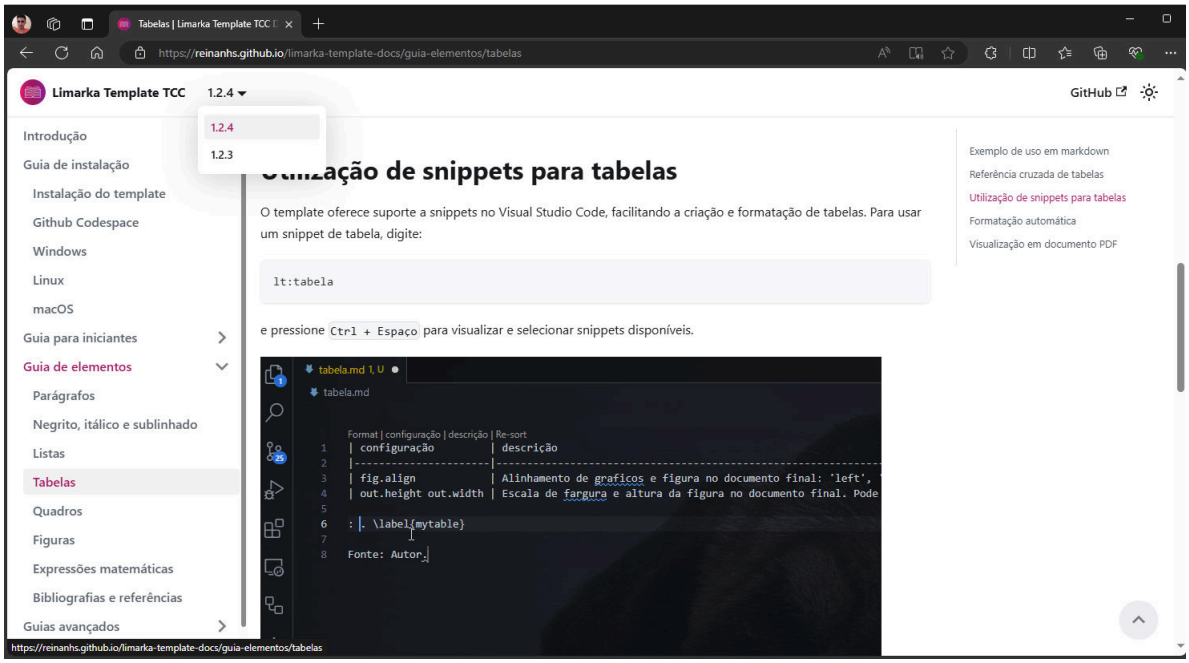
Para alinhar ainda mais o projeto com a filosofia de DaC e permitir uma maior customização, foi necessário criar um repositório separado denominado *limarka-template-docs*. Esse repositório foi concebido para manter a documentação como um projeto independente, mas profundamente conectado ao repositório principal, *limarka-template-tcc*. Dessa maneira, a documentação pode evoluir de forma paralela ao código, mantendo-se sempre em sincronia com as novas versões do template, sem comprometer a organização ou a integridade do código-fonte.

A integração entre os dois projetos foi realizada por meio de uma funcionalidade criada para baixar as alterações do projeto principal, utilizando a API do GitHub, e sincronizá-las com o projeto de documentação no Docusaurus. As modificações no conteúdo não impactem diretamente o repositório principal, permite uma separação clara entre o desenvolvimento de novas funcionalidades e a atualização da documentação, mantendo a consistência e clareza do projeto como um todo.

Para integrar esses dois repositórios, foi necessário desenvolver um código que utiliza a Application Programming Interface (API) do GitHub para transferir informações de um repositório para o outro. Esse código é responsável por baixar todas as versões publicadas no repositório principal e convertê-las para o formato do Docusaurus utilizando o conteúdo do diretório *docs*. Dessa forma, os alunos podem visualizar as diferentes versões da documentação associadas às respectivas versões do template, facilitando a compreensão das mudanças e mantendo a compatibilidade.

A Figura 23 ilustra a documentação gerada, onde o aluno pode acessar informações detalhadas sobre o template desenvolvido ao longo deste trabalho. Ademais, a *interface* permite que o aluno selecione a versão específica do template para comparar as diferenças nas configurações, especialmente em casos de quebra de compatibilidade entre versões.

Figura 23 – Ilustração da página de documentação



Fonte: Autor, (2024).

6 Disponibilizar o Limarka em um ambiente docker padronizado

O Limarka oferece uma imagem Docker oficial que pode ser utilizada em seu template para facilitar a compilação de documentos acadêmicos (ALEXANDRE, 2019). Contudo, a última atualização dessa imagem ocorreu em 2020, o que resultou em uma desatualização significativa dos componentes, trazendo à tona diversas vulnerabilidades de segurança. Essa situação é preocupante, especialmente em um ambiente acadêmico onde a integridade e a segurança dos dados são cruciais.

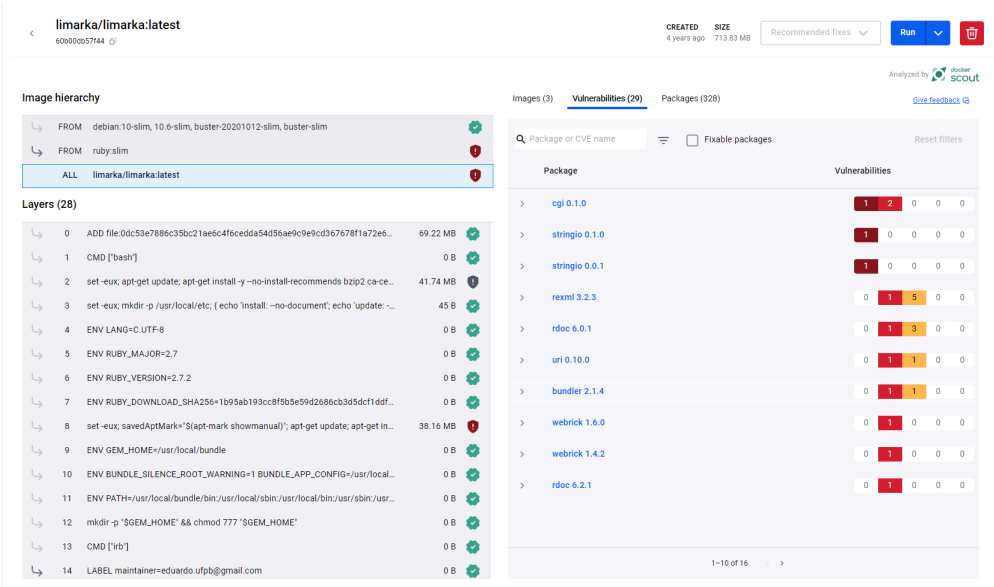
Uma análise recente realizada pelo Docker Scout revelou que a imagem Docker oficial do Limarka, em sua versão mais recente, apresenta 29 vulnerabilidades conhecidas. Segundo Reche (2024), o Docker Scout é uma ferramenta essencial para profissionais de DevOps que desejam assegurar a segurança e conformidade de suas imagens Docker. Na opinião de Guedes (2018), DevOps é um modelo que une filosofias culturais, práticas e ferramentas com o objetivo de aumentar a capacidade das empresas de entregar serviços de forma mais rápida e eficiente. Essa abordagem integra as áreas de desenvolvimento e operações de software, promovendo uma maior qualidade nas entregas.

Ao analisar o relatório gerado por essa ferramenta, foi constatado que a imagem foi construída com base no Debian 10 e no Ruby Slim, ambos em versões que já foram superadas por alternativas mais recentes e otimizadas. A Figura 24 ilustra essa análise de segurança, destacando as vulnerabilidades encontradas na imagem do Limarka.

Diante das falhas de segurança identificadas e da obsolescência dos componentes utilizados, tornou-se imperativo desenvolver uma nova imagem Docker para o Limarka. Essa nova imagem busca corrigir as vulnerabilidades existentes e atualizar os componentes essenciais, garantindo um ambiente mais seguro e eficiente para os usuários. Para facilitar essa atualização e centralizar as melhorias, foi criado um repositório no GitHub dedicado à manutenção e evolução da imagem Docker do Limarka.

Além da atualização e correção de vulnerabilidades, a criação dessa nova imagem Docker também visa a padronização do ambiente de desenvolvimento e execução do Limarka. Isso permitirá que estudantes e pesquisadores utilizem uma versão mais segura e otimizada da ferramenta, sem a necessidade de enfrentar as limitações e riscos associados à versão desatualizada. Com isso, espera-se que o uso do Limarka em ambiente Docker seja mais confiável, promovendo um fluxo de trabalho mais seguro e eficiente.

Figura 24 – Análise feita pelo Docker Scout na imagem Docker oficial do Limarka



Fonte: Autor, (2024).

6.1 Criação e automação do repositório limarka-docker

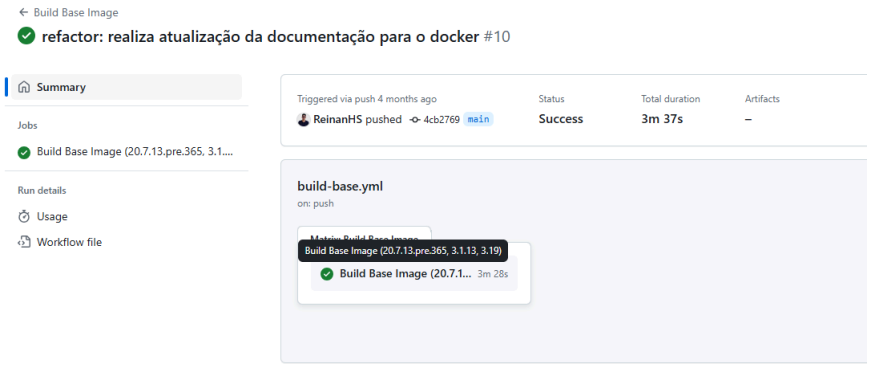
O repositório limarka-docker [1] foi criado com o objetivo de facilitar a manutenção e a atualização das imagens Docker utilizadas pelo Limarka. A criação desse repositório permitiu a personalização completa do ambiente de desenvolvimento e execução, incorporando todas as dependências e configurações necessárias para suportar as novas funcionalidades implementadas no projeto.

Uma das principais melhorias implementadas no repositório foi a automação do processo de construção e publicação das imagens Docker. Essa automação foi configurada para gerar novas versões das imagens sempre que alterações fossem feitas no repositório, assegurando que as melhorias e correções fossem disponibilizadas aos usuários de maneira rápida e eficiente. Para alcançar esse nível de automação, foram utilizadas as ferramentas de CI/CD oferecidas pelo GitHub Actions, permitindo um fluxo de trabalho contínuo e integrado ao ciclo de desenvolvimento. A Figura 25 ilustra o processo de automação do *build* e da publicação da imagem no Docker Hub.

Além dessas melhorias, uma mudança significativa realizada no repositório foi a substituição da base do sistema operacional da imagem Docker, que anteriormente utilizava o Debian, pela Alpine Linux. A escolha da Alpine foi motivada por diversas vantagens, entre elas, conforme cita Rodrigues (2021), a Alpine é uma imagem popular devido ao seu design minimalista, que inclui apenas o essencial para o funcionamento do sistema, resultando em uma imagem de

[1] https://github.com/ReinanHS/limarka-docker

Figura 25 – Exemplo da automatização do build e publicação da imagem Docker



Fonte: Autor, (2024).

menor tamanho e que consome menos espaço em disco. Essa característica é especialmente vantajosa em ambientes de produção, onde a eficiência e a economia de recursos são fundamentais.

Outro fator decisivo para a escolha da Alpine foi a facilidade de gerenciamento das vulnerabilidades de segurança. Comparada a distribuições mais tradicionais como Ubuntu, CentOS ou Debian, a Alpine oferece uma abordagem muito mais focada na eficiência e na segurança. Enquanto distribuições como Debian são amplamente utilizadas em servidores e *desktops*, a Alpine se destaca em cenários onde a segurança e a otimização de recursos são prioritárias (MEU LINUX, 2023). Essa mudança, portanto, não apenas contribui para a segurança do ambiente Docker utilizado pelo Limarka, mas também reforça o compromisso com a utilização de tecnologias que garantam um desempenho superior e uma menor superfície de ataque.

A implementação dessas melhorias no repositório limarka-docker reflete uma abordagem moderna e eficiente na gestão de ambientes Docker, alinhada com as melhores práticas de desenvolvimento de software.

7 Implementação de uma nova camada CLI

A ferramenta Limarka oferece uma *interface* de linha de comando que foi desenvolvida para atender usuários do Windows, especialmente aqueles que não têm familiaridade com comandos de terminal, permitindo o acesso por meio do navegador de arquivos. Essa funcionalidade, chamada de menu interativo, possibilita aos usuários a realização de uma sequência de comandos para compilar dados e criar documentos acadêmicos de forma automatizada (ALEXANDRE, 2017).

A *Command Line Interface* (CLI) do Limarka é fundamental para a execução eficiente do fluxo de trabalho, especialmente para aqueles que buscam uma experiência controlada e simplificada na criação de seus trabalhos acadêmicos. No entanto, visando aprimorar a experiência do usuário e automatizar processos adicionais, foi implementada uma nova camada CLI, denominada *limarka-help*.

Desenvolvida em PHP e distribuída como um executável, a *limarka-help* foi projetada para ser facilmente integrada aos ambientes de desenvolvimento dos usuários. Sua principal função é atuar como um intermediário, executando operações adicionais antes que o Limarka processe a compilação final do documento. Entre essas operações, destaca-se a capacidade de processar e compilar informações contidas no arquivo de configuração, além de incluir novos recursos que permitem a importação e combinação de arquivos Markdown no arquivo principal (*trabalho-academico.md*).

A motivação para a construção dessa nova camada surgiu da necessidade de oferecer maior flexibilidade e modularidade ao processo de compilação. Com a *limarka-help*, os usuários podem segmentar seus documentos em diferentes arquivos Markdown, o que facilita a manutenção, a revisão e a organização do conteúdo modular permite que textos como resumos, agradecimentos ou propósitos sejam mantidos em arquivos separados e facilmente integrados no documento principal durante a fase de compilação.

7.1 Benefícios e importância das melhorias

A introdução da *limarka-help* traz uma série de benefícios que aprimoram significativamente o fluxo de trabalho dos usuários do Limarka. Primeiramente, a modularidade proporcionada por essa nova camada simplifica a estruturação de documentos acadêmicos, permitindo que conteúdos extensos e distintos sejam organizados de forma eficiente.

7.2 Construção e funcionamento da camada CLI

O *limarka-help* foi desenvolvido como uma extensão simples, mas poderosa, da CLI original do Limarka. O script PHP que compõe essa nova camada começa carregando e verificando a existência de arquivos de configuração essenciais, como o *configuracao.YAML*. Em seguida, ele processa as informações contidas nesses arquivos, substituindo marcadores de importação por conteúdos reais retirados de outros arquivos Markdown. Esse processo de substituição é crucial para garantir que o documento final seja uma compilação precisa e coesa de todas as partes individuais do projeto.

A Figura 26 ilustra a configuração do arquivo YAML, onde o usuário pode utilizar a *tag* @*import* para importar o conteúdo de arquivos externos em Markdown para dentro do arquivo de configuração. Essa funcionalidade permite que o aluno mantenha o conteúdo modularizado, facilitando a organização e a manutenção do projeto.

Figura 26 – Exemplo de importação pelo arquivo de configuração

```yaml
coorientador: "Prof. Francisco Rodrigues Santos"
orientador: "Prof. Gilson Pereira Dos Santos Junior"
resumo: "@import(pages/resumo.md)"
abstract_texto: "@import(pages/abstract.md)"
proposito: "@import(pages/proposito.md)"
agradecimentos: "@import(pages/agradecimentos.md)"
```

Fonte: Autor, (2024).

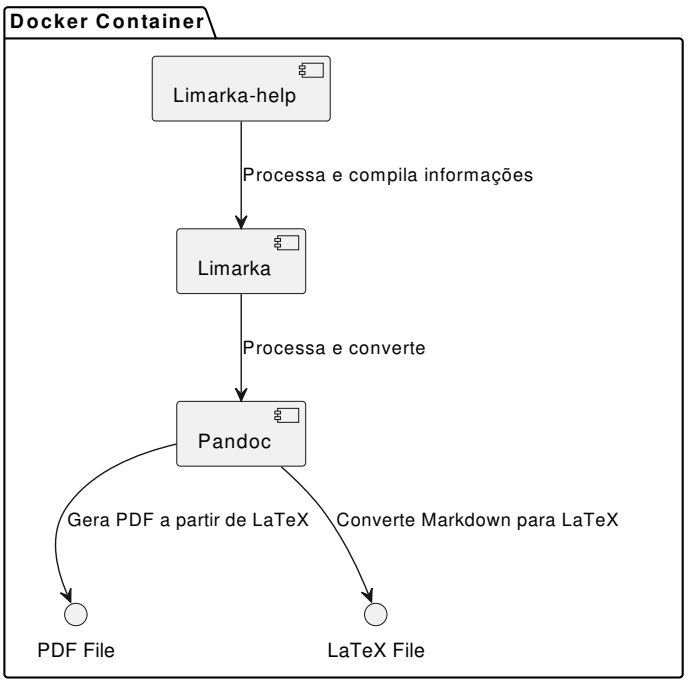
Com a *tag* @*import*, o usuário pode simplesmente apontar para o caminho do arquivo que deseja incluir, como mostrado na imagem. Por exemplo, @*import(pages/resumo.md)* instrui o *limarka-help* a buscar o conteúdo do arquivo *resumo.md* dentro do diretório *pages* e substituir a *tag* de importação pelo conteúdo real do arquivo durante o processo de compilação. Essa abordagem modulariza o conteúdo do trabalho acadêmico, permitindo que o aluno trabalhe em seções específicas de forma independente, enquanto o *limarka-help* garante que todas as partes sejam integradas de maneira coesa no documento final.

Característica importante do *limarka-help* é sua capacidade de criar um ambiente temporário onde toda a compilação é realizada. Isso significa que o documento original e seus arquivos associados permanecem inalterados até que o processo de compilação esteja completo e aprovado. Somente após a conclusão da compilação, os arquivos resultantes são movidos para o diretório de *build* designado, organizando de forma limpa os artefatos gerados e facilitando o gerenciamento desses arquivos.

A Figura 27 ilustra o diagrama de componentes que destaca a interação entre os diversos elementos envolvidos no processo de compilação utilizando o Limarka e o *limarka-help*. O Docker é apresentado como a camada fundamental onde todas as ferramentas operam, propor-

cionando um ambiente padronizado e isolado para garantir a consistência do processo como observado no item seção 6.1.

Figura 27 – Diagrama de componentes sobre organização do limarka-help



Fonte: Autor, (2024).

Dentro dessa camada Docker, o *limarka-help* atua como o primeiro componente a processar as informações do projeto. Ele realiza a leitura dos arquivos de configuração YAML, substituindo as *tags* de importação por conteúdo real dos arquivos Markdown. Após essa etapa, o *limarka-help* envia os dados processados para o Limarka, que é responsável por formatar e preparar o documento para a conversão final.

O Limarka, por sua vez, interage com o Pandoc em duas fases distintas: primeiro, ele utiliza o Pandoc para converter o arquivo Markdown em um formato LaTeX, uma linguagem de preparação de documentos que permite uma formatação sofisticada. Em seguida, o Pandoc converte o arquivo LaTeX resultante em um PDF, que é o formato final do documento.

A separação das responsabilidades entre o *limarka-help*, Limarka, e Pandoc facilita futuras atualizações e manutenções, permitindo que cada componente seja aprimorado independentemente, sem afetar os outros.

8 Trabalhos futuros

Nesta seção, serão discutidas possíveis direções para expandir e melhorar a ferramenta apresentada neste trabalho. As propostas visam aumentar a aplicabilidade da solução, sua eficiência e aceitação em diferentes contextos acadêmicos e profissionais. A seguir, destacam-se algumas iniciativas e estudos que poderão ser realizados em futuros desenvolvimentos.

8.1 Expansão da compatibilidade para outras normas

Um dos principais desenvolvimentos para a ferramenta seria aumentar a sua compatibilidade com as normas académicas de outras instituições, de modo a poder ser utilizada em contextos internacionais e também em cursos que exigem determinados formatos, além das normas da ABNT. No momento, a solução está sendo criada de acordo com as normas da ABNT, o que não é flexível para seu uso em outros contextos. Ampliar sua compatibilidade com outras normas acadêmicas significaria muito para aumentar seu uso em ambientes internacionais. Além disso, se isso for aliado a um sistema flexível que permita seu uso para definir preferências em função de uma ou mais normas diferentes, isso poderia aumentar a relevância da ferramenta se houver normas diferentes da ABNT.

8.2 Estudo sobre impacto da ferramenta entre estudantes

Uma investigação sobre a aceitação da ferramenta entre os estudantes é uma proposta relevante para validar sua usabilidade e eficácia. Seriam conduzidos estudos qualitativos e quantitativos para medir o impacto da ferramenta no processo de escrita acadêmica, buscando compreender as percepções dos usuários em relação à facilidade de uso e à qualidade dos documentos gerados.

Esse estudo poderia envolver a aplicação de questionários e entrevistas para coletar feedback dos usuários quanto à facilidade de uso, eficácia e impacto no processo de criação de documentos acadêmicos. Além disso, poderiam ser realizados testes de usabilidade com estudantes de diferentes áreas e níveis acadêmicos, como graduação e pós-graduação, para identificar possíveis barreiras no uso da ferramenta e áreas que necessitam de melhorias.

Seria interessante avaliar a eficiência da ferramenta em termos de economia de tempo e recursos em comparação com outros métodos tradicionais, como o uso de editores de texto convencionais ou ferramentas como o LaTeX. Este estudo ajudaria a identificar o impacto da ferramenta na produtividade dos estudantes e sua aceitação como um padrão de mercado.

8.3 Adaptação para outros cursos e áreas de conhecimento

Outra grande área de desenvolvimento seria a adaptação da ferramenta para apoiar a diversidade de áreas de conhecimento e cursos académicos. Cada ramo da ciência tem os seus requisitos especiais e peculiaridades na estrutura dos documentos e na formatação de elementos como tabelas, gráficos ou citações, entre outros. Os ramos da engenharia, por exemplo, necessitam provavelmente da inclusão de fórmulas matemáticas complexas, enquanto que nas ciências sociais é necessário dar mais importância à exatidão da citação das fontes.

8.4 Reduzir o tempo de compilação de documentos

Por fim, outra área de aprimoramento seria a otimização do tempo de compilação dos documentos. Reduzir o tempo necessário para gerar versões finais em PDF ou outros formatos poderia aumentar a produtividade dos usuários, tornando o processo de escrita mais ágil e eficiente.

O tempo de espera entre ajustes no conteúdo e a geração do documento final pode impactar negativamente a experiência do usuário, especialmente em projetos acadêmicos onde o refinamento e a revisão constante são partes fundamentais do processo de escrita. Para abordar essa questão, futuros trabalhos podem se concentrar na otimização do processo de compilação, explorando técnicas como a compilação incremental, onde apenas as partes alteradas do documento são recompiladas, ou a paralelização das tarefas de processamento.

8.5 Realizar a implementação de novas validações no CI

Para melhorar a eficiência do processo de desenvolvimento e garantir a conformidade dos documentos gerados, novas validações podem ser implementadas no pipeline de integração contínua (CI). Isso incluiria verificações automatizadas para garantir que os documentos estejam de acordo com as normas acadêmicas, evitando erros de formatação ou estrutura.

9 Conclusão

O presente trabalho se propôs a aprimorar o Limarka, uma ferramenta de formatação de trabalhos acadêmicos em conformidade com as normas da ABNT, visando atender de maneira mais eficaz as demandas dos alunos do curso de BSI do IFS - Campus Lagarto. Através da integração de metodologias de DevOps, bem como da ampliação de suas funcionalidades, o Limarka foi transformado em uma solução moderna, flexível e robusta, apta a simplificar e automatizar o processo de elaboração de TCC.

Um dos pontos de destaque deste trabalho foi a inclusão de uma camada CLI customizada, o Limarka-Help, que aprimorou significativamente a modularidade e a usabilidade da ferramenta. Através desta camada, foi possível oferecer aos usuários uma maneira simplificada de segmentar e integrar seções do trabalho, facilitando tanto a manutenção quanto a revisão dos documentos. Essa abordagem modular não apenas tornou o processo de criação mais eficiente, mas também reduziu a incidência de erros comuns, contribuindo para a produção de trabalhos acadêmicos de alta qualidade.

Outro avanço substancial foi a adoção de práticas de automação por meio da integração do Limarka com Docker e GitHub Actions. A utilização dessas tecnologias viabilizou a criação de um ambiente de desenvolvimento padronizado, onde estudantes e professores podem colaborar de forma eficiente, independentemente de suas configurações locais. O uso de *pipelines* automatizadas para a compilação e validação contínua dos documentos garante que cada versão do trabalho esteja sempre em conformidade com as normas acadêmicas, mantendo a consistência e a qualidade do produto final. Essa integração não só aumenta a produtividade dos alunos, mas também promove a adoção de boas práticas de desenvolvimento, alinhadas às tendências modernas da engenharia de software.

Além das melhorias técnicas, a inclusão do Marp como ferramenta para a criação de slides de apresentação diretamente a partir de arquivos Markdown trouxe um diferencial significativo. Este recurso permite que os alunos mantenham um fluxo de trabalho unificado, desde a criação do texto do TCC até a preparação da defesa, utilizando a mesma infraestrutura e as mesmas práticas de versionamento e automação. A criação de um tema acadêmico específico para o Marp, baseado nos padrões visuais do IFS, garantiu que as apresentações dos alunos sejam não apenas esteticamente agradáveis, mas também coerentes com a identidade institucional, facilitando a preparação e aumentando a confiança dos estudantes durante suas defesas.

A comparação detalhada realizada entre o Limarka e outras ferramentas de escrita acadêmica, como Manubot, Authorea, Overleaf, Google Docs, Microsoft Word e LibreOffice Writer, evidenciou que, apesar de suas limitações iniciais, o Limarka, após as melhorias implementadas, se posiciona como uma alternativa viável e competitiva para os alunos de BSI. Suas novas

funcionalidades, aliadas à simplicidade do Markdown e à robustez das práticas de DevOps, proporcionam uma experiência de uso que une o melhor de dois mundos: a flexibilidade e a eficiência de uma ferramenta moderna com a conformidade e a precisão exigidas por padrões acadêmicos rigorosos.

Dessa forma, conclui-se que as melhorias introduzidas no Limarka representam um avanço significativo na forma como os estudantes do curso de BSI do IFS podem produzir seu TCC. O trabalho realizado não só simplifica o processo de escrita e formatação, como também capacita os alunos a adotar práticas mais sofisticadas de desenvolvimento e colaboração, preparando-os melhor para os desafios técnicos e profissionais que enfrentarão após a conclusão do curso.

Referências

ABNTEX. *abnTeX*: Suíte para latex compatível com as normas da abnt. 2019. Disponível em: <https://www.abntex.net.br/>. Acesso em: 18 ago. 2024.

ADOBE. *Como usar o Markdown para escrever a documentação*. 2024. Disponível em: <https://experienceleague.adobe.com/pt-br/docs/contributor/contributor-guide/writing-essentials/markdown>. Acesso em: 12 ago. 2024.

ALEXANDRE, E. d. S. M. Utilização de markdown para elaboração de tccs: concepção e experimento da ferramenta limarka. Universidade Federal da Paraíba, 2017. Disponível em: <https://repositorio.ufpb.br/jspui/bitstream/tede/9034/2/arquivototal.pdf>. Acesso em: 18 ago. 2024.

ALEXANDRE, E. de S. M. *Estudante da UFPB cria ferramenta para ajudar na elaboração de trabalhos acadêmicos*. 2016. Disponível em: <https://www.ufpb.br/antigo/content/estudante-da-ufpb-cria-ferramenta-para-ajudar-na-elabora%C3%A7%C3%A3o-de-trabalhos-acad%C3%AAmicos>. Acesso em: 18 ago. 2024.

ALEXANDRE, E. de S. M. *Modelo de trabalho acadêmico para utilizar com o limarka*. 2016. Página do projeto. Disponível em: <https://github.com/abntex/trabalho-academico-limarka>. Acesso em: 28 ago. 2024.

ALEXANDRE, E. de S. M. *Editando o texto*. 2017. Página da documentação. Disponível em: <https://github.com/abntex/limarka/wiki/Editando-o-texto>. Acesso em: 28 ago. 2024.

ALEXANDRE, E. de S. M. *Gerando o PDF*. 2017. Página da documentação. Disponível em: <https://github.com/abntex/limarka/wiki/Gerando-o-PDF>. Acesso em: 28 ago. 2024.

ALEXANDRE, E. de S. M. *Gerando o PDF*. 2019. Página da documentação. Disponível em: <https://github.com/abntex/limarka/wiki/Limarka-com-Docker>. Acesso em: 28 ago. 2024.

ALONSO, A. C. R.; BERTI, L. F. Introducao a edicao de textos em latex utilizando a plataforma overleaf. *Semana da Matemática do Instituto de Matemática*, n. 3, 2019. Disponível em: <https://periodicos.ufms.br/index.php/SMIM/article/view/9002>. Acesso em: 18 ago. 2024.

ANSON, D. *Markdownlint*. 2024. Disponível em: <https://github.com/DavidAnson/markdownlint>. Acesso em: 28 ago. 2024.

AYERS, C. *Marp - Create Presentations with Markdown*. 2023. Disponível em: <https://chris-ayers.com/2023/03/26/marp-create-presentations-with-markdown>. Acesso em: 01 set. 2024.

BARMINA, A. A. Automating the process of creating technical documentation based on the docs as code approach. *BBK 1 H 34*, p. 2364, 2023. Disponível em: <https://na-journal.ru/pdf/nauchnyi_aspekt_5-2023_t19_web.pdf#page=42>. Acesso em: 18 ago. 2024.

BOLTON, D. *Is it better to write large programs in multiple files or one file?* 2023. Disponível em: <https://www.quora.com/Is-it-better-to-write-large-programs-in-multiple-files-or-one-file/answer/David-Bolton-2>. Acesso em: 28 ago. 2024.

BRILHAUS, D. et al. One resource to teach them all. In: *Proceedings of the Conference on Research Data Infrastructure*. [S.l.: s.n.], 2023. v. 1.

CARDOSO, R. *GitHub*: o que é e como usar? [guia completo]. 2024. Disponível em: <https://www.locaweb.com.br/blog/temas/codigo-aberto/github/>. Acesso em: 17 ago. 2024.

CHEREM, Y. A. A utilização do sistema de preparação de textos latex na produção de textos acadêmicos no brasil: uma investigação preliminar e perspectivas. *Universidade Federal de São Paulo*, v. 20, n. 1, p. 228–249, 2015. Disponível em: <https://ojs.uel.br/revistas/uel/index.php/informacao/article/view/18495>. Acesso em: 06 ago. 2024.

CLOUDFLARE. *O que é um gerador de site estático?* 2024. Disponível em: <https://www.cloudflare.com/pt-br/learning/performance/static-site-generator/>. Acesso em: 28 ago. 2024.

COELHO, B. *LaTeX*: as vantagens e desvantagens de usar em seu trabalho. 2018. Disponível em: <https://blog.mettzer.com/latex/>. Acesso em: 18 ago. 2024.

CUBOS ACADEMY. *Aprenda a criar seus snippets no VSCode de forma fácil e eficiente*. 2023. Disponível em: <https://blog.cubos.academy/programacao-snippets-no-vscode/>. Acesso em: 22 ago. 2024.

DODD, R. *A 5-minute Markdown tutorial*. 2018. Disponível em: <https://about.gitlab.com/blog/2018/08/17/gitlab-markdown-tutorial/>. Acesso em: 17 ago. 2024.

FELIPE, A. *Reta final para a graduação*: A importância do tcc. 2019. Disponível em: <https://blogs.uninassau.edu.br/noticias/comunicacao-social-fotografia-jornalismo-publicidade-e-propaganda/reta-final-para-graduacao>. Acesso em: 01 set. 2024.

FILHO, I. F. F. *Escrever trabalhos com normas ABNT utilizando markdown? É possível com o Limarka*. 2017. Disponível em: <https://yzakius.me/2017/05/01/escrever-trabalhos-com-normas-abnt-utilizando-markdown-e-possivel-com-o-limarka/>. Acesso em: 18 ago. 2024.

GHAFOURI, M. *Projects Folder Structures Best Practices*. 2021. Disponível em: <https://dev.to/mattqafouri/projects-folder-structures-best-practices-g9d>. Acesso em: 05 set. 2024.

GITHUB. *Introdução aos contêineres de desenvolvimento*. 2023. Disponível em: <https://docs.github.com/pt/codespaces/setting-up-your-project-for-codespaces/adding-a-dev-container-configuration/introduction-to-dev-containers>. Acesso em: 24 ago. 2024.

GITHUB. *Sobre bifurcações*. 2024. Disponível em: <https://docs.github.com/pt/pull-requests/collaborating-with-pull-requests/working-with-forks/about-forks>. Acesso em: 21 ago. 2024.

GOMES, R. V. C. et al. Git e github: Desenvolvendo habilidades essenciais para colaboração e controle de versões. *Sociedade Brasileira de Computação*, 2023.

GOOGLE. *Markdown (optional)*. 2024. Disponível em: <https://developers.google.com/tech-writing/one/markdown>. Acesso em: 17 ago. 2024.

GRAYSON, K. L.; HILLIKER, A. K.; WARES, J. R. R markdown as a dynamic interface for teaching: Modules from math and biology classrooms. *Mathematical Biosciences*, v. 349, p. 108844, 2022. ISSN 0025-5564. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0025556422000499>.

GUEDES, M. *Afinal, o que é DevOps?* 2018. Disponível em: <https://www.treinaweb.com.br/blog/afinal-o-que-e-devops>. Acesso em: 02 out. 2024.

HIMMELSTEIN, D. S. et al. Open collaborative writing with manubot. *PLOS Computational Biology*, Public Library of Science, v. 15, n. 6, p. 1–21, jun. 2019. Disponível em: <https://doi.org/10.1371/journal.pcbi.1007128>. Acesso em: 17 ago. 2024.

HOFERT, M.; KOHM, M. Scientific presentations with latex. *The PracTEX Journal*, 2010.

KREWINKEL, A.; WINKLER, R. Formatting open science: agilely creating multiple document formats for academic manuscripts with pandoc scholar. *PeerJ Preprints*, v. 5, p. e2648v2, mar. 2017. ISSN 2167-9843. Disponível em: <https://doi.org/10.7287/peerj.preprints.2648v2>. Acesso em: 12 ago. 2024.

LIBREOFFICE. *Writer*. 2024. Disponível em: <https://pt-br.libreoffice.org/descubra/writer/>. Acesso em: 18 ago. 2024.

LOGAN, D.; SELHORN, S.; TACKER, S. *Five fast facts about docs as code at GitLab*. 2022. Disponível em: <https://about.gitlab.com/blog/2022/10/12/five-fast-facts-about-docs-as-code-at-gitlab/>. Acesso em: 14 ago. 2024.

MAILUND, T. Writing markdown. In: ____. *Introducing Markdown and Pandoc: Using Markup Language and Document Converter*. Berkeley, CA: Apress, 2019. p. 13–22. ISBN 978-1-4842-5149-2. Disponível em: <https://doi.org/10.1007/978-1-4842-5149-2_3>. Acesso em: 15 ago. 2024.

MALLETTE, J. C.; ACKLER, H. Valuing womens contributions: Team projects and collaborative writing. ASEE Conferences, n. 10.18260/1-2–31223, jun. 2018. Disponível em: <https://peer.asee.org/31223>. Acesso em: 17 ago. 2024.

MANG, J. M.; PROKOSCH, H.-U.; KAPSNER, L. A. Reproducibility in 2023–an end-to-end template for analysis and manuscript writing. In: *Caring is Sharing–Exploiting the Value in Data for Health and Innovation*. [S.l.]: IOS Press, 2023. p. 58–62.

MEDEIROS, E. de S. *Documentação com Markdown*: Textos acadêmicos conforme as regras abnt com o limarka. 2021. Disponível em: <https://www.inteligenciaurbana.org/2021/05/markdown-parte3-limarka.html>. Acesso em: 07 ago. 2024.

MEU LINUX. *O Alpine Linux é uma distro leve, segura e altamente eficiente*. 2023. Disponível em: <https://meulinux.com.br/alpine/>. Acesso em: 31 ago. 2024.

MICROSOFT. *An open specification for enriching containers with development specific content and settings*. 2024. Disponível em: <https://containers.dev/>. Acesso em: 24 ago. 2024.

MINETTO, E. *Boas práticas na criação de milestones, tarefas, pull requests e commits*. 2017. Disponível em: <https://eltonminetto.dev/post/2017-11-13-boas-praticas-tarefas-pr-commits/>. Acesso em: 17 ago. 2024.

MITOLU, I. *Markdown for Technical Writers*: Tips, tricks, and best practices. 2023. Disponível em: <https://israelmitolu.hashnode.dev/markdown-for-technical-writers-tips-tricks-and-best-practices>. Acesso em: 13 ago. 2024.

MONUTTI, D. *Como Trabalhar com Código Modular?* 2024. Disponível em: <https://www.hashtagtreinamentos.com/codigo-modular-javascript>. Acesso em: 28 ago. 2024.

MOZILLA. *Wrangling Web Contributions*: How to build a contributing.md. 2024. Disponível em: <https://mozillascience.github.io/working-open-workshop/contributing/#:~:text=A%20CONTRIBUTING.md%20file%2C%20in,your%20project%20or%20study%20group.> Acesso em: 21 ago. 2024.

NILSON, D. M. J. et al. Utilização de softwares livres em educação a distância em medicina e saúde: uma experiência de 6 anos. In: *Congresso Brasileiro de Informática em Saúde*. [S.l.: s.n.], 2006.

OELEN, A.; AUER, S. Content authoring with markdown for visually impaired and blind users. In: *2019 IEEE International Symposium on Multimedia (ISM)*. [S.l.: s.n.], 2019. p. 285–290.

OLIVEIRA, E. D. S. de; GÓES, F. K. F. da S. O uso do google docs para aprendizagem colaborativa. *Revista Campo do Saber*, v. 7, n. 1, 2021. Disponível em: <https://periodicos.iesp.edu.br/campodosaber/article/view/379>. Acesso em: 18 ago. 2024.

PALMOWSKI, M. *Crie e Personalize um Site Docusaurus (Com Recurso de Blog)*. 2023. Disponível em: <https://kinsta.com/pt/blog/docusaurus/>. Acesso em: 31 ago. 2024.

PENA, B. *Descubra 11 funções do Word para Facilitar seus Projetos!* 2021. Disponível em: <https://voitto.com.br/blog/artigo/microsoft-word>. Acesso em: 18 ago. 2024.

PEPE, A.; SHENK, J. *Getting Started with Authorea*. 2023. Disponível em: <https://advance.sagepub.com/users/3/articles/319801-getting-started-with-authorea>. Acesso em: 18 ago. 2024.

PROVIDELLO, W. *Visual Studio Code*: A importância de uma poderosa ferramenta de desenvolvimento. 2023. Disponível em: <https://www.dio.me/articles/visual-studio-code-a-importancia-de-uma-poderosa-ferramenta-de-desenvolvimento>. Acesso em: 22 ago. 2024.

PUC-SP. *Gerenciadores de Referências e Citações*. 2024. Disponível em: <https://www.pucsp.br/laboratorio-de-pesquisa-em-administracao/gerenciadores-de-referencias-e-citacoes>. Acesso em: 18 ago. 2024.

RECHE, G. *Melhorando a Segurança e Conformidade com Docker Scout na Cultura DevOps*. 2024. Disponível em: <https://www.linkedin.com/pulse/melhorando-seguran%C3%A7a-e-conformidade-com-docker-scout-na-gabriel-reche-ieamf/>. Acesso em: 31 ago. 2024.

REDHAT. *O que é CI/CD?* 2023. Disponível em: <https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd>. Acesso em: 24 ago. 2024.

RIMAL, Y. Reproducible academic writing and interactive data visualization using r markdown (r programming flex-dashboard: Flex_dashboard packages). In: RATHORE, V. S. et al. (Ed.). *Rising Threats in Expert Applications and Solutions*. Singapore: Springer Singapore, 2021. p. 603–615. ISBN 978-981-15-6014-9.

RODRIGUES, V. *Porque você deveria rever o uso de imagens Alpine em Dockerfiles para Python e qual imagem escolher*. 2021. Disponível em: <https://vilsonrodrigues.medium.com/872d8b4a3e54>. Acesso em: 24 ago. 2024.

ROY, S. *Authorea*: An alternative to overleaf, researchgate, or arxiv? 2021. Disponível em: <https://shantoroy.com/latex/authorea-dot-com-an-alternative-to-overleaf-researchgate-arxiv/>. Acesso em: 01 set. 2024.

SCIENCE, T. D. *Opportunities and Obstacles for Deep Learning in Biology and Medicine*. 2017. Disponível em: <https://towardsdatascience.com/opportunities-and-obstacles-for-deep-learning-in-biology-and-medicine-6ec914fe18c2>. Acesso em: 04 set. 2024.

SHIEBER, S. M. Why scholars should write in markdown. *Harvard University*, 2014. Disponível em: <https://archive.blogs.harvard.edu/pamphlet/files/2014/08/markdownpost-amsart.pdf>. Acesso em: 15 ago. 2024.

SILVA, E. *Docs-as-Code*: Documentação como código. 2022. Disponível em: <https://esli.blog.br/docs-as-code-documentacao-como-codigo>. Acesso em: 18 ago. 2024.

SILVA, M. J. C. N. d. Escrita acadêmica: manual de elaboração das referências usando o mendeley. *Universidade Federal do Maranhão*, p. 1–27, 2024. Disponível em: <https://repositorio.ufma.br/jspui/handle/123456789/1014>. Acesso em: 06 ago. 2024.

SILVA, T. L. da; MARTINS, F. N.; COSTA, W. T. Criação de tutoriais e relatos de experiências sobre a utilização de softwares livres no ensino de engenharia. In: *COBENGE-XL Congresso Brasileiro de Educação em Engenharia*. [S.l.: s.n.], 2012.

SILVEIRA, P. *Markdown*: como trabalhar com essa linguagem de markup? 2022. Disponível em: <https://www.alura.com.br/artigos/como-trabalhar-com-markdown>. Acesso em: 07 ago. 2024.

SIMIONI, D. *Fundador do LibreOffice comenta sobre MS Office*. 2021. Disponível em: <https://diolinux.com.br/noticias/fundador-do-libreoffice-comenta-sobre-ms-office.html>. Acesso em: 06 ago. 2024.

SIMIONI, D. *Git, GitHub e GitLab*: Entenda as diferenças e como utilizá-los. 2023. Disponível em: <https://diolinux.com.br/editorial/git-github-e-gitlab.html>. Acesso em: 17 ago. 2024.

STARLING, P. H. C.; SANTOS, V. A. d; GERAIS, M. Github actions: Uma análise do impacto de actions de code quality e code review em repositórios open-source do github. 2022. Disponível em: <http://bib.pucminas.br:8080/pergamumweb/vinculos/000020/000020b3.pdf>. Acesso em: 17 ago. 2024.

STROSS, C. *Why Microsoft Word Must Die*. 2013. Disponível em: <https://www.antipope.org/charlie/blog-static/2013/10/why-microsoft-word-must-die.html>. Acesso em: 15 ago. 2024.

TENEN, D.; WYTHOFF, G. Autoria sustentável em texto simples usando pandoc e markdown. *The Programming Historian em Português*, ProgHist Ltd, 2022. Disponível em: <https://www.proquest.com/openview/6c9a039194da04c759fddc16c0b457be/1?pq-origsite=gscholar&cbl=6458207>. Acesso em: 06 ago. 2024.

VISUAL STUDIO. *Integrate with External Tools via Tasks*. 2024. Disponível em: <https://code.visualstudio.com/docs/editor/tasks>. Acesso em: 28 ago. 2024.

WILLBERG, H. P.; FORSSMANN, F. *Erste Hilfe in Typografie*: Ratgeber für gestaltung mit schrift. [S.l.]: Verlag Hermann Schmidt, 1999. ISBN 3874394743.