INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL – IFRS

GIANCARLO FONTELA DA LUZ

Intrinsically Fascinating MarkDown: Conversão de arquivos MarkDown para PDF utilizando tecnologias web.

Canoas
2024

CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GIANCARLO FONTELA DA LUZ

Intrinsically Fascinating MarkDown: Conversão de arquivos MarkDown para PDF utilizando tecnologias web.

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul –Campus Canoas.

Profa. Dra. Carla Odete Balestro Silva
Orientadora

Canoas
2024

Ministério da Educação
Secretaria de Educação Profissional, Científica e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
Campus Canoas

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 11 dias do mês de DEZEMBRO de 2024, às 10 horas, em sessão pública na sala E7 do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, Campus Canoas, na presença da Banca Examinadora presidida pelo(a) Professor(a):

CARLA ODETE BALESTRO SILVA e

composta pelos examinadores:

1. PROF. DR. DIEGSON SOARES SILVEIRA
2. PROF. DR. MARIANO NICOLAO,
3. _____,

o(a) aluno(a) GIANCARLO FONTELA DA LUZ

apresentou o Trabalho de Conclusão de Curso intitulado:

INTRINSICALLY FASCINATION MARKDOWN: CONVERSÃO DE ARQUIVOS MARKDOWN PARA PDF UTILIZANDO TECNOLOGIAS WEB

como requisito curricular indispensável para a integralização do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Após reunião em sessão reservada, a Banca Examinadora deliberou e decidiu pela APROVAÇÃO do referido trabalho, divulgando o resultado formalmente ao aluno e demais presentes e eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

Presidente da Banca Examinadora

Examinador 01

Examinador 02

Examinador 03

Aluno

RESUMO

O presente produto teve como objetivo produzir uma ferramenta para a criação de documentações de projetos de software, com intenção de aproveitar conhecimentos já adquiridos por desenvolvedores ao longo de suas carreiras. Sendo uma aplicação de terminal, o público-alvo se restringe a desenvolvedores com experiência em desenvolvimento e metodologias ágeis. O produto permite a criação de documentações em PDF a partir de arquivos Markdown, além de possuir capacidade para personalização utilizando tecnologias *web*, como HTML, CSS e JavaScript, permitindo que as equipes que utilizarão o produto personalizem seu conteúdo de acordo com suas necessidades. O desenvolvimento foi guiado pelo padrão cascata, com objetivos, claros e concisos do começo ao fim, com modelagem utilizando diagramas UML para facilitar a visualização dos componentes. Para garantir qualidade no código produzido, foram feitos testes com automatizados com cerca de 80% de cobertura, além de testes com usuários qualificados para verificar a usabilidade. O produto foi publicado no GitHub e no NPM, gerenciador de pacotes do NodeJS.

Palavras-chave: Documentação, Metodologias Ágeis, Aplicação de Terminal.

ABSTRACT

The product has, as it is objective, produce a tool for creating documentation of software projects, with the intention of taking advantage of knowledge already obtained from developers in their carriers. Being a terminal application, the target audience is restricted to experienced developers with knowledge in agile methodologies. The product allows the creation of PDF documentations from Markdown files, and have customization capacity using web technologies, such as HTML, CSS and JavaScript, allowing teams to customize their documentations according to their needs. The development was driven by the waterfall development methodology, with clear and concise objectives from the start to finish, using UML diagrams to help visualize the components of the system. To guarantee quality o the code produced, automated testes with roughly 80% of coverage were done, and users tests were conducted to verify the usability of the product. The product was published on GitHub and on NPM, the NodeJS package manager.

Keywords: Documentation, Agile Methodoloy, Terminal Application.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CSS	*Cascading Style Sheets*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
NPM	*Node Package Manager*
PDF	*Portable Document Format*
URL	*Unique Resource Identifier*

SUMÁRIO

1 INTRODUÇÃO

Esta pesquisa constitui o projeto final do curso de Tecnologia em Análise e Desenvolvimento de Sistema, aplicando conceitos de Engenharia de Software e Desenvolvimento para *Web* na forma de um produto. O objetivo era criar uma ferramenta para contribuir no processo de documentação em metodologias ágeis na área de Tecnologia da Informação, tendo como público-alvo desenvolvedores.

Conforme Cockburn *et al*. (2001), a documentação faz parte do desenvolvimento com metodologias ágeis, embora o foco principal seja a funcionalidade do software. Com isto em mente, a pesquisa visou entregar um software que facilita a construção de documentações em aplicativos, reutilizando conhecimentos já adquiridos pelos desenvolvedores ao longo de suas carreiras. A solução proposta converte arquivos Markdown — “um formato de texto para escrever documentos estruturados” (Ehe *et al*., 2023, tradução própria) — para arquivos PDF (*Portable Document Format*), um formato com um “uso explosivo pela internet [...] se tornado o formato padrão para troca de documentos” (ADOBE, 2006). Para formatação e design, foram utilizadas tecnologias comuns em *websites*, como HTML, CSS e JavaScript. A abordagem da pesquisa foi qualitativa, com natureza aplicada e objetivo exploratório. A solução funciona baseada em NodeJS, uma plataforma para execução de código JavaScript, e utiliza seu gerenciador de pacotes, o NPM (Node Package Manager). A linguagem de desenvolvimento utilizada foi o TypeScript, uma linguagem com tipos que compila para JavaScript (TYPESCRIPT, 2024). Além disso, as tecnologias para *web* citadas acima anterior também foram utilizadas na ferramenta.

1.1 MOTIVAÇÃO

O Manifesto Ágil é uma declaração de valores para o desenvolvimento de software, com o objetivo de entregar melhores maneiras de desenvolver (COCKBURN *et al*., 2001). Contendo quatro valores e doze princípios, o Manifesto Ágil serve como base para diversas metodologias ágeis, essenciais para a melhor gestão de projetos e sucesso em equipes de desenvolvimento (LOSNAK, 2023).

Um dos valores do Manifesto Ágil é “Software em funcionamento mais que documentação abrangente” (COCKBURN *et al*., 2001), destacando a importância da documentação do projeto sem priorizá-la sobre a própria solução. Por conta disso, ter um processo de documentação mais eficiente torna todo o fluxo de desenvolvimento mais efetivo, além de maximizar o foco dos desenvolvedores no objetivo principal: o desenvolvimento do software.

Uma forma de documentar projetos é utilizar arquivos Markdown. Segundo o Github (2023), mais de 100 milhões de usuários utilizam o seu serviço de hospedagem de repositórios de código, que emprega o formato de arquivo Markdown para a documentação de repositórios. No entanto, como este formato não possui implementação própria, sendo uma “forma de escrever documentos estruturados” (EHE *et al*., 2023), não há como garantir que todos os leitores terão a mesma experiência em diferentes plataformas de leitura.

O resultado da pesquisa foi um produto de terminal — uma interface do usuário com o computador por meio do teclado (AMAZON, 2023) — que permite a conversão direta de arquivos Markdown para arquivos PDF, amplamente adotados na internet (ADOBE, 2006). Para lidar com a falta de estilização nativa do Markdown, a solução utiliza tecnologias web, como HTML, CSS e JavaScript, para aplicar estilizações semelhantes às usadas na criação de *websites*. Desta forma, as possibilidades de personalização são vastas, já que grande parte do que é possível aplicar em um *website* pode ser aplicado no documento gerado. Além disso, se houver desenvolvedores *web* na equipe, é possível utilizar seus conhecimentos de linguagens de marcação e estilização para criar *templates* das documentações finais.

A aplicação funciona por meio de comandos, com diversos serviços internos atuando de forma conjunta para entregar o resultado ao usuário, seja este para o gerenciamento de dados da aplicação ou para a produção de arquivos PDF. Sendo assim, é esperado que o usuário possua um certo nível de conhecimento com

aplicações de terminal, além de conhecimento em escrita e documentação de projetos de *software*.

1.2 OBJETIVOS

Este capítulo trata dos objetivos do produto, divido em objetivo geral e objetivos específicos.

1.2.1 OBJETIVO GERAL

Desenvolver solução que automatize a geração de documentações em formato PDF originando-se de arquivos Markdown para o conteúdo e tecnologias *web* para a estilização.

1.2.2 OBJETITIVOS ESPECÍFICOS

Seguem os objetivos específicos da pesquisa.

- Estudar bibliotecas e ferramentas existentes que possam contribuir para o projeto;
- Analisar requisitos de usuários para a conversão de arquivos;
- Modelar a solução utilizando diagramas UML;
- Testar a aplicação;
- Produzir documentação de uso da aplicação.

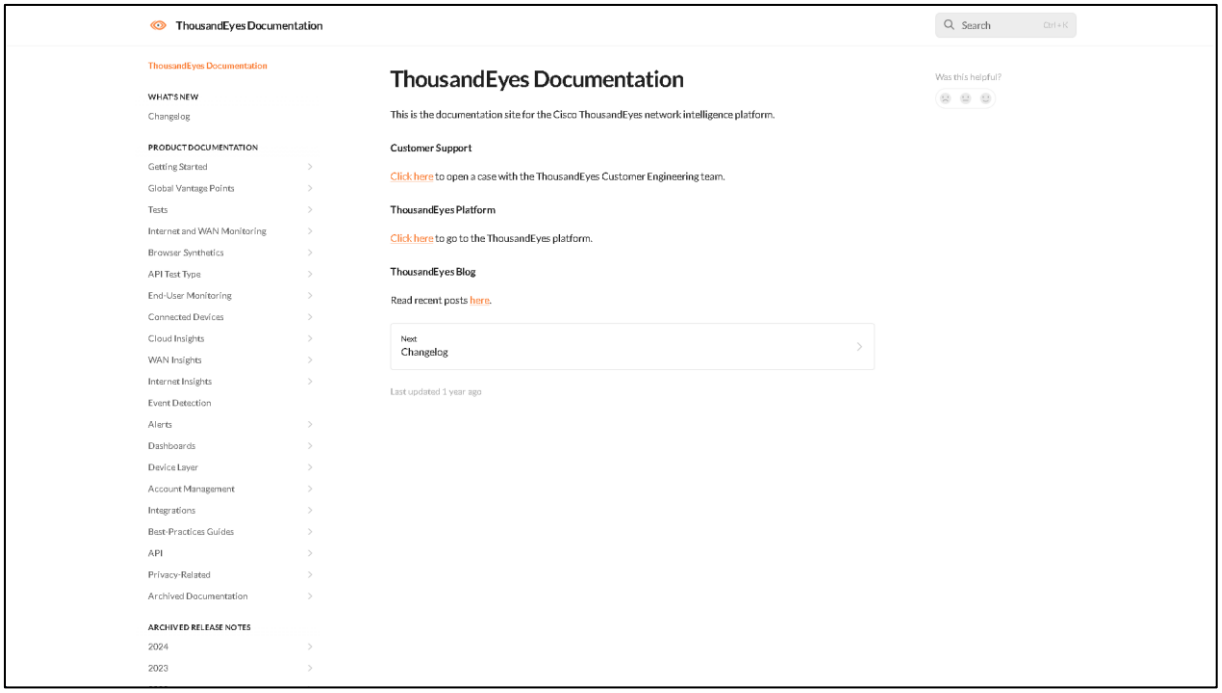
2 ESTADO DA ARTE

Este capitulo visa mostrar soluções semelhantes ao problema abordado.

2.1 GITBOOK

O GitBook[1] é um projeto open-source para a criação de documentações no formato *web* a partir de arquivos Markdown, com integrações para múltiplas plataformas, como GitHub, GitLab e Notion. Esta plataforma oferece comentários, personalização e diversas ferramentas de integração para facilitar o processo de criação de documentações.

Figura 1 - Exemplo de documentação usando o GitBook



Fonte: ThousandEyes Documentation (2024).

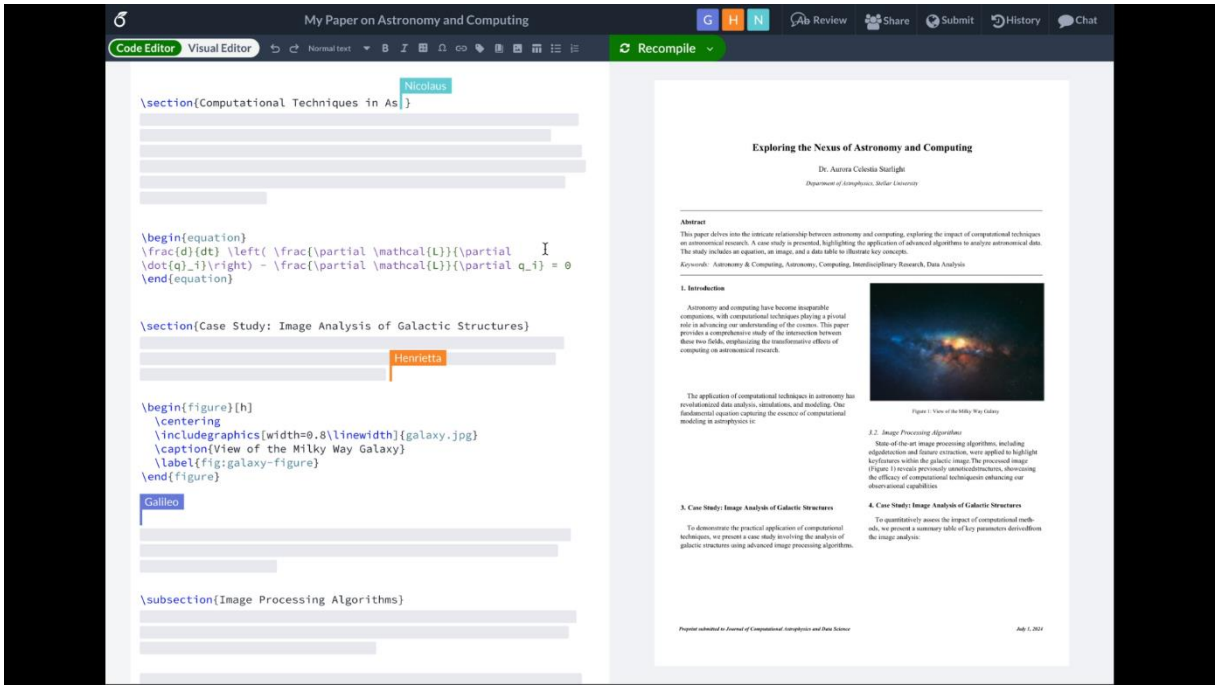
Conforme mostrado na Figura 1, a ferramenta pode ser utilizada para gerar documentações de alta qualidade sem um grande esforço dos desenvolvedores.

[1] Disponível em: https://www.gitbook.com/solutions/public-docs. Acesso em: 12 de novembro de 2024.

2.2 OVERLEAF

Overleaf é uma plataforma de edição de arquivos LaTeX, que é um sistema de composição tipográfica de alta qualidade, voltado para produções técnicas e acadêmicas (LATEX, 2024).

Figura 2 - Projeto no Overleaf



Fonte: Overleaf (2024).

Conforme visto na Figura 2, o Overleaf permite a edição de arquivos LaTeX diretamente no navegador, permitindo colaboração em tempo e real e a pré-visualização do arquivo PDF.

2.3 CONSIDERAÇÕES SOBRE AS FERRAMENTAS

Conforme visto, ambas as ferramentas entregam formas automatizadas e polidas de se fazer documentações em projetos de código. Uma possui integrações com repositórios de código, e outras possui uma linguagem poderosa que permite grande personalização do resultado final.

Porém, em cada uma delas, falta um aspecto: o GitBook não foi preparado para criação de PDFs, não sendo adequado para a produção de documentações fora do contexto *web*. Já o Overleaf utiliza uma linguagem complexa, o LaTeX, que, por

possuir um público-alvo mais acadêmico, dificilmente se aproveitará dos conhecimentos já adquiridos pelos desenvolvedores.

A ferramenta desenvolvida possui um apelo aos conhecimentos já adquiridos por desenvolvedores ao longo de suas carreiras, além de permitir que cada especialista atue em sua área, com desenvolvedores de código se preocupando apenas com o conteúdo da documentação que irão trabalhar, e designers e desenvolvedores *web* trabalhando em cima da tipografia e design desejado, sem um interferir no trabalho do outro.

3 REFERENCIAL TEÓRICO

Este capítulo aborda questões sobre o ciclo de desenvolvimento de *software*, o modelo de desenvolvimento cascata e recursos atrelados ao projeto.

Conforme Sommervile (2024):

> A engenharia de software tem por objetivo apoiar o desenvolvimento profissional de software, mais do que a programação individual. Ela inclui técnicas que apoiam especificação, projeto e evolução de programas, que normalmente não são relevantes para o desenvolvimento de software pessoal (SOMMERVILE, 2024, p. 17).

Logo, a Engenharia de software tem um valor importante no desenvolvimento do projeto. Para a produção deste trabalho, foi utilizado a metologia cascata, que segundo a Adobe (2024) se assemelha ao ditado popular “meça duas vezes, corte apenas uma”. O objetivo foi analisado e então desenvolvido, para no final ser testado, seguindo esta metologia como base.

Para medir o objetivo, foram utilizados os diagramas UML, que, de acordo com BOOCH *et al*. (1999, p. 1) é uma forma de produzir diagramas similares a plantas, ou projetos de outras áreas do conhecimento.

Utilizando o diagrama de caso de uso, é possível demonstrar o comportamento geral de um sistema, com seus usuários e procedimentos, permitindo que desenvolvedores e usuários concordem em como utilizar o sistema (BOOCH *et al.*, 1999, p. 40). O diagrama de sequência mostra as interações entre os objetos do sistema (BOOCH *et al.*, 1999, p. 50). Já o diagrama de atividade mostra o fluxo de estado da aplicação, mostrando a transição de informações (BOOCH, *et al.*, 1999, p. 159). Todos estes ajudam a montar uma ideia geral de como a aplicação deve funcionar. Usando estes princípios como guia, é possível chegar a um resultado previsível desde o período do planejamento.

Com respeito a aplicações *web*, o funcionamento delas funciona através de um protocolo chamado HTTP, que permite que usuários solicitem recursos utilizando, na maioria das vezes, navegadores. Para enviar estes recursos, é necessário um servidor HTTP, que geralmente utiliza HTML (*HyperText Transfer Protocol*), para definir o conteúdo que vai ser apresentado ao usuário, que, por sua vez, pode ser estilizada utilizando CSS (*Cascading Style Sheets*) e possuir interatividade com

JavaScript (MOZILLA, 2024). Utilizando estes três elementos em conjunto, além de outros recursos como imagens e vídeos, é possível criar uma aplicação *web*.

Também, para permitir uma comunicação bidirecional, existem os WebSockets, protocolo que permite uma troca de mensagens entre o servidor e o cliente, mantendo uma conexão aberta entre os dois (MOZILLA, 2024).

Além de navegadores *web*, existem outras maneiras de se comunicar com o usuário. Uma delas é por meio do terminal, que conecta o usuário com o *software* por meio do teclado (AMAZON, 2023).

Para armazenar dados de forma persistente foi utilizado o sistema de arquivos do usuário[2], que, segundo FREECODECAMP (2022), define como arquivos são nomeados, armazenados e recuperados de um dispositivo de armazenamento.

[2] O uso um de sistema gerenciadores de bancos de dados também é uma opção viável para a construção de aplicações. Neste caso especifico esta tecnologia não foi utilizada pois os dados armazenados já estão em formato de arquivo, logo, não há necessidade de uma ferramenta especifica para lidar com essas informações.

4 METODOLOGIA

A pesquisa utilizou uma abordagem qualitativa, que representa uma “sequência de atividades” (GIL, 1991, p. 133), a fim de alcançar a solução desejada, sendo estes passos o desenvolvimento do aplicativo.

Tem natureza aplicada, ou seja, para a “geração de conhecimento para solução de problemas específicos” (NASCIMENTO, 2016, p. 1) e envolveu a produção de um software com a intenção de solucionar o problema descrito na motivação, convertendo arquivos do formato Markdown (GITHUB, 2024) para o formato PDF utilizando tecnologias *web*, como HTML, CSS e JavaScript para criar *templates* de estilização reutilizáveis.

Com objetivo exploratório, a pesquisa aspira proporcionar uma solução para o problema proposto, a documentação na área de Tecnologia da Informação, sendo um “aprimoramento de ideias” (GIL, 1991, p. 41), baseando-se em tecnologias já existentes, supracitadas no parágrafo anterior.

Para obter dados amostrais foram feitas entrevistas, que, de acordo com Gil (1991, p. 114), é uma técnica de coleta de dados em que duas pessoas conversam, uma perguntando e outra respondendo, para um grupo de profissionais da área de Tecnologia da Informação inseridos em ambientes que aplicam metodologias ágeis, de forma a validar a utilidade da solução proposta. A experiência do autor deste trabalho com desenvolvedor *web* também contribuiu para a compreensão das necessidades da aplicação.

Para o desenvolvimento, foi o padrão cascata foi utilizado, que, segundo Adobe (2024, p.1) é “uma forma de gerenciamento de projeto que enfatiza uma progressão linear do início ao fim”, tendo uma evolução mais estável e controlada, considerando que os requisitos para o projeto estão bem claros nos objetivos específicos da pesquisa. Para modelar a aplicação, foram utilizados diagramas UML, "Uma linguagem que produz diagramas comparáveis [...] com plantas de projetos usadas em outras disciplinas" (BOOCH *et al*, 1999, p. 1, tradução nossa), a fim de distinguir bem as funções de cada componente.

As tecnologias utilizadas foram NodeJS, uma plataforma para execução de código JavaScript, e utiliza seu gerenciador de pacotes, o NPM (Node Package Manager), a linguagem de programação TypeScript, além de tecnologias *web*, como HTML, CSS e JavaScript.

O produto foi munido de testes automatizados, além de ter sido disponibilizado para potenciais usuários de forma a validar a aplicação.

4 ANÁLISE E INTERPRETAÇÃO DOS RESULTADOS

Este capítulo descreve as etapas de desenvolvimento baseando-se na modelagem, testes automatizados e testes com usuários.

4.1 ANÁLISE DE REQUISITOS E MODELAGEM

Cockburn *et al*. (2001) afirmam no Manifesto Ágil que a documentação faz parte do processo produtivo de *software*, porém, que o seu valor é menor que o desenvolvimento do produto, indicando assim que documentar deve ter menor prioridade sobre desenvolver.

Com o objetivo de diminuir a carga de documentar o produto, o presente projeto utilizou o Markdown, formato de arquivo de texto sem formatação para produzir arquivos estruturados (EHE *et al*., 2023), como base de informação, mesclando com tecnologias utilizadas em *websites*, como HTML, CSS e JavaScript, para a formatação e design.

Ao fornecer um formato de arquivo simples de escrever, os desenvolvedores terão mais tempo para se concentrar no objetivo principal: o *software*. Ao mesmo tempo permitindo a personalização do design e formatação de acordo com os requisitos do proprietário do projeto. Por ser um *template*, ou seja, um modelo, ele pode ser reaproveitado em múltiplas documentações, concentrando os esforços de estilização em um único projeto, evitando assim retrabalho para estilizar cada documentação, diminuindo assim a carga de trabalho dos designers.

Tendo em vista que o produto não tem como público-alvo pessoas leigas em relação a desenvolvimento de *software*, a interface de comunicação com o produto é o terminal, ou seja, uma interface do usuário com o computador por meio do teclado (AMAZON, 2023), em que o usuário escreve comandos em forma de texto e recebe resultados da mesma forma, sem o uso de ferramentas gráficas. Esta opção, além de permitir que os usuários interajam com o produto diretamente, permite que ele seja usado em ambientes não interativos, como fluxos de integração e entrega contínuos (REDHAT, 2024).

Para validar os resultados com usuários qualificados, foi elaborado um questionário contendo perguntas a respeito de suas qualificações e também sobre a ferramenta em si, de forma a, após o desenvolvimento, verificar se a aplicação realmente entrega valor ao usuário final.

A primeira pergunta é “Qual seu tempo de experiência em desenvolvimento de *software?*”, dividindo as opções entre “Menos de 1 ano”, “Entre 1 e 3 anos”, “Entre 3 e 5 anos” e “Mais de 5 anos”, assim dando uma visão mais ampla da experiência do entrevistado.

A segunda pergunta diz respeito ao cargo do entrevistando, visando validar a área de excelência, e como isto pode afetar a avaliação da ferramenta, sendo ela “Qual cargo melhor descreve sua função dentro do mercado de trabalho?”.

A próxima questão contempla a questão das metologias ágeis: “Qual item abaixo melhor descreve sua experiência com Metologias Ágeis?”, com as possíveis respostas sendo “Nenhuma”, “Consumi conteúdo sobre, mas nunca utilizei”, “Já trabalhei em alguns projetos”, “Frequentemente utilizo” e “É meu principal meio de produzir software”, para saber se a aplicação se enquadra dentro do ambiente do entrevistado.

As próximas 5 perguntas dizem respeito ao projeto em si, considerando que o usuário já realizou a instalação e testou a aplicação. A primeira destas questões é: “Qual item abaixo melhor descreve sua experiência com ferramentas de terminal (CLIs)?”, com as respostas podendo ser “Nenhuma”, “Raramente utilizo”, “Eventualmente utilizo”, “Frequentemente utilizo” e “Faz parte do meu fluxo de trabalho”, com o objetivo de analisar a experiência do usuário utilizando uma aplicação apresentada em um terminal.

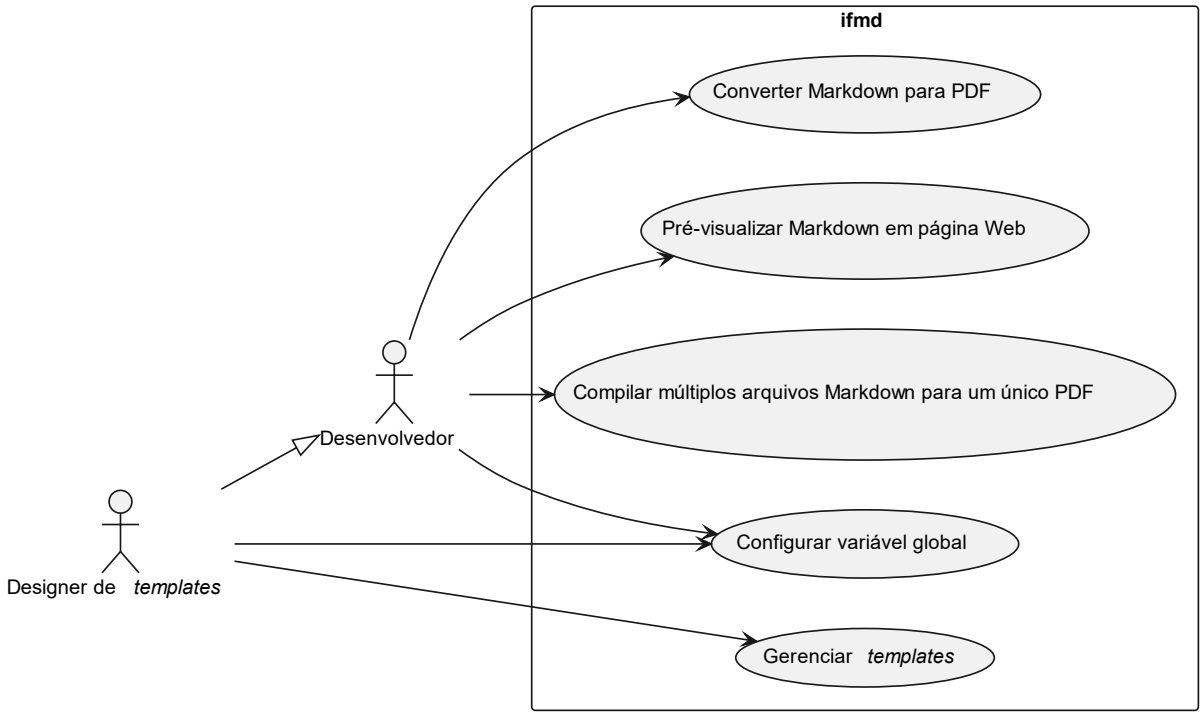
Perguntando diretamente sobre a aplicação, a próxima pergunta diz respeito a usabilidade: “Numa escala de 1 a 5, como a aplicação se encaixa em questão de usabilidade, sendo 1 muito ruim e 5, ótimo?”

Contemplando o objetivo do projeto, a seguinte questão é: “Numa escala de 1 a 5, como a aplicação se encaixa em questão ao seu objetivo, de acelerar a produção de documentações por parte dos desenvolvedores, ao mesmo tempo que permite personalização por parte dos *designers*, sendo 1 muito pouco e 5, muito?”

Finalizando as perguntas obrigatórias, é perguntado: “Você utilizaria esta ferramenta em seu fluxo de trabalho?”, com as opções “Sim”, “Não” e “Talvez”, solicitando a opinião profissional do entrevistado sobre a aplicação.

Por fim, como pergunta opcional, o usuário poderia deixar sugestões de melhoria para a aplicação.

Figura 3 - Diagrama de Caso de Uso para a ferramenta

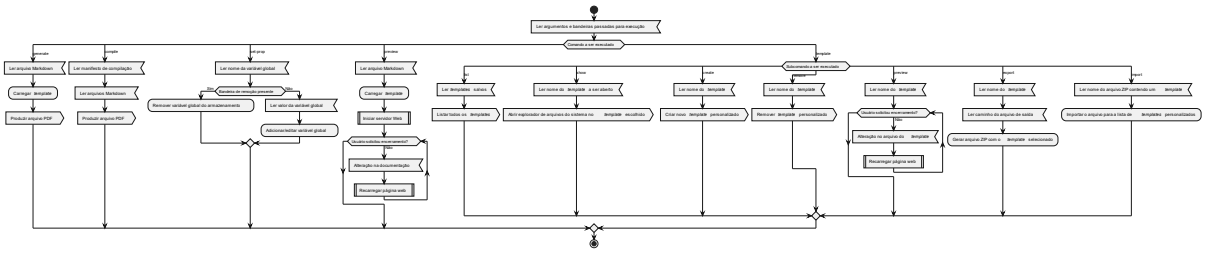


Fonte: Elaborado pelo autor (2024).

Após definir os requisitos do projeto, foi elaborado um diagrama comportamental de caso de uso, conforme a Figura 3, para definir os próximos passos de desenvolvimento. O produto possui dois atores: o desenvolvedor e o designer de *templates*, sendo que o designer de *templates* também é um desenvolvedor. As especificações deste diagrama estão presentes no APÊNDICE B.

O caso de uso “Gerenciar *templates*” é exclusivo ao Designer de *templates*, que englobam todas as operações de criação, edição, empacotamento e exclusão de *templates* de formatação e estilização. Os outros comandos estão relacionados ao uso de templates, como a produção de uma documentação em PDF ou compilação de vários arquivos em um único, e são exclusivos do desenvolvedor.

Figura 4 - Diagrama de Atividade

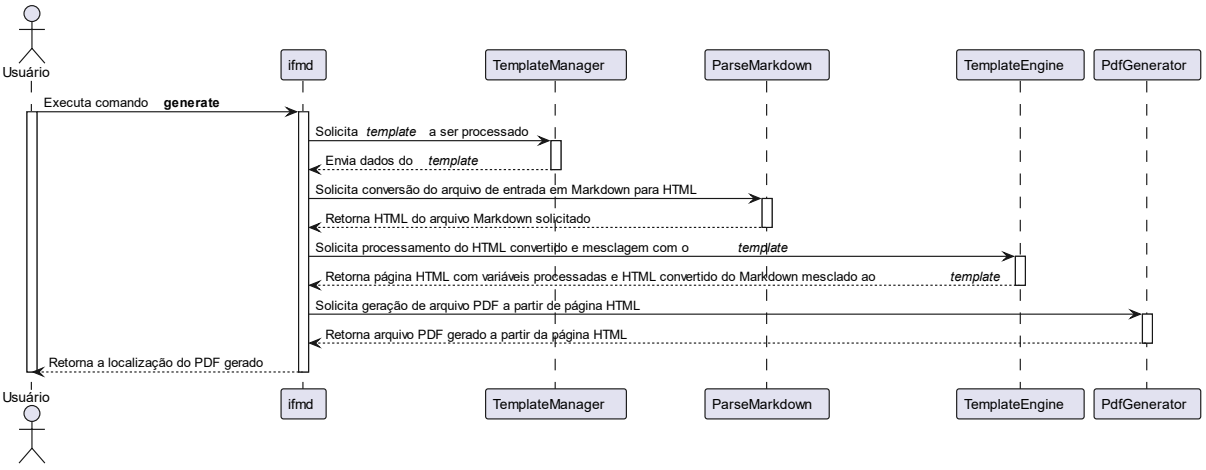


Fonte: Elaborado pelo autor (2024).

O diagrama de atividade, apresentado na Figura 4[3], mostra o fluxo de operações realizadas na aplicação, demonstrando o comportamento da aplicação em cada uma das situações em que foi projetado para atuar. O primeiro passo envolve ler e processar o comando enviado pelo terminal, como argumentos e bandeiras[4]. Depois disso a aplicação fará um desvio no fluxo para atender à solicitação especifica do usuário, levando-o até um dos cinco comandos principais.

Tendo uma visão geral do funcionamento da aplicação, foram feitas as definições de comportamento especificas de cada comando.

Figura 5 - Diagrama de Sequência do comando “generate”



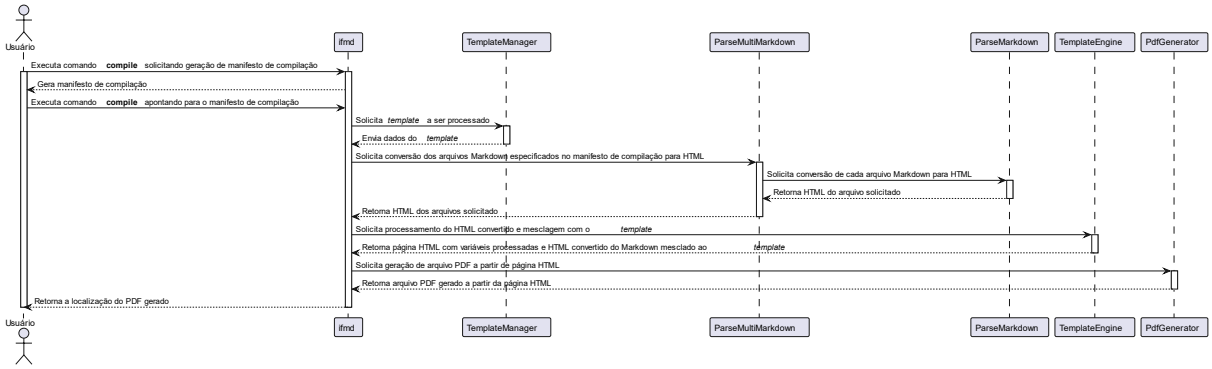
Fonte: Elaborado pelo autor (2024).

[3] Por conta do tamanho e dificuldade de visualização em documento impresso, uma versão em alta definição foi feita em modo paisagem no APÊNDICE C.

[4] Neste contexto, argumentos são textos blocos de texto separados por espaço inseridos após a invocação do comando, e bandeiras são argumentos que começam com um ou dois caracteres “-“, seguidos pelo nome da bandeira, com o valor que representam logo em seguida. Por exemplo: “comando argumento1 --bandeira valor1” tem como lista de argumentos o “argumento1” e possui uma bandeira com nome “bandeira” e valor “valor1”. Outras aplicações podem ter definições diferentes do que são argumentos ou bandeiras, esta definição é especifica para o presente projeto.

O primeiro comando é o “generate”, que, conforme a Figura 5, produz uma saída PDF a partir de um documento Markdown. É possível notar que existem diversos componentes internos que não são visíveis para o usuário final, como o “TemplateManager”, “ParseMarkdown”, “TemplateEngine” e “PdfGenerator”. Estes componentes são serviços especializados em uma área especifica do fluxo de trabalho, permitindo maior desacoplamento de responsabilidades, e, assim, deixando o código mais organizado e fácil de manter. Eles serão explicados em mais detalhes na seção 4.2 SERVIÇOS INTERNOS neste capítulo.

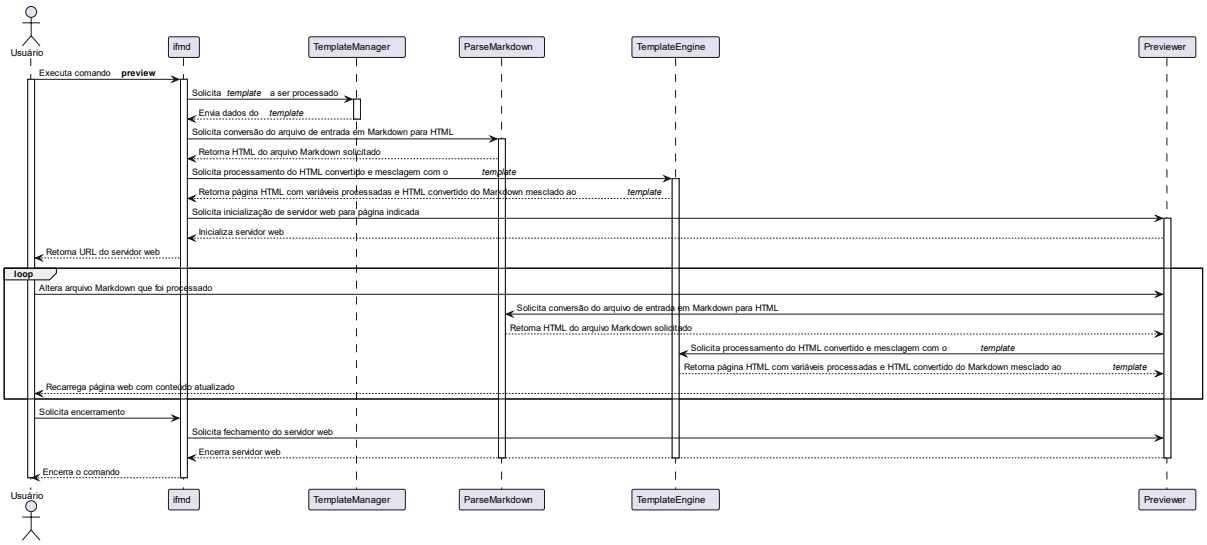
Figura 6 - Diagrama de Sequência do comando “compile”



Fonte: Elaborado pelo autor (2024).

O comando “compile” segue o mesmo princípio, conforme visto na Figura 6. Este comando utiliza um arquivo de controle chamado “Manifesto de compilação” para definir quais arquivos e em qual ordem devem ser concatenados para formar um arquivo único em PDF. Após uma execução para criar ou editar o manifesto de compilação, o comando pode ser executado novamente para aplicar as informações na produção do arquivo PDF final.

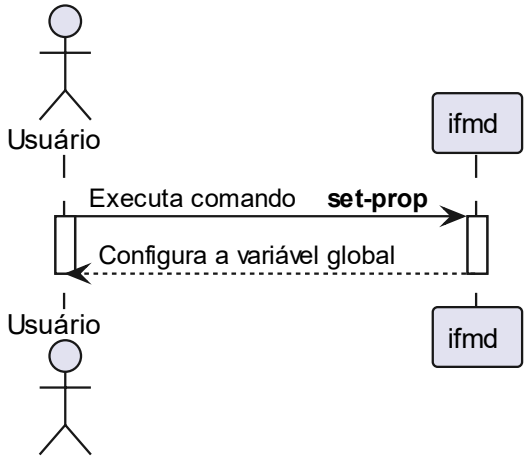
Figura 7 - Diagrama de Sequência do comando “preview”



Fonte: Elaborado pelo autor (2024).

Durante o processo de criação de uma documentação, pode ser necessário revisar múltiplas vezes o resultado do documento que está sendo escrito. Para ajudar neste processo, o comando “preview”, cujo diagrama de sequência está na Figura 7, foi concebido. Diferente da maioria dos outros comandos da aplicação, que normalmente retornam o controle ao usuário assim que terminam de realizar suas tarefas, este comando inicializa um servidor *web* com o resultado em HTML do documento sendo escrito, já no contexto do seu *template*. Sendo assim, qualquer alteração feita no documento pelo usuário é refletida poucos segundos depois no navegador, permitindo uma pré-visualização do conteúdo final.

Figura 8 - Diagrama de Sequência do comando “set-prop”

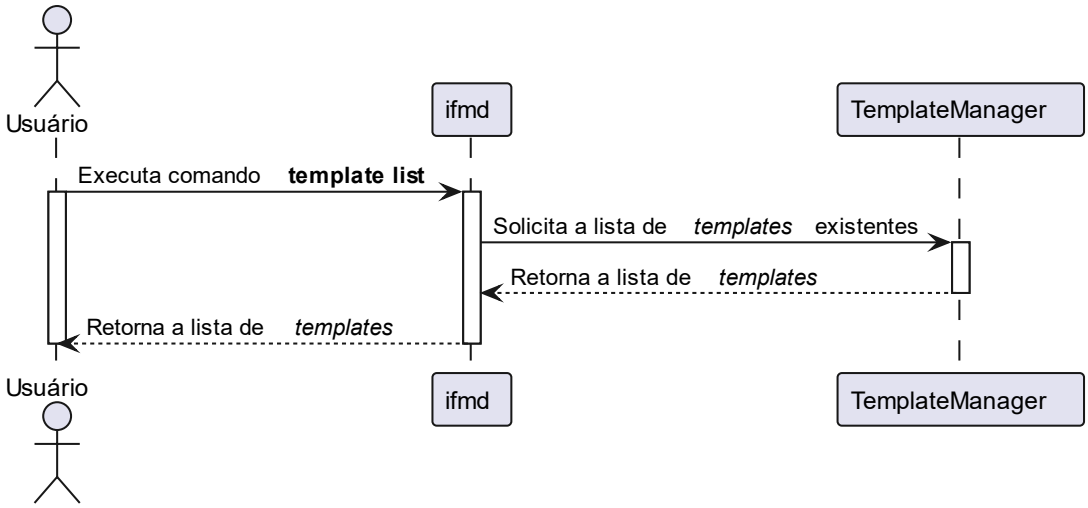


Fonte: Elaborado pelo autor (2024).

Ao gerar documentos, o produto conta com um sistema de macros, que podem ser utilizadas tanto pelos arquivos de documentação quanto pelos *templates*. Estas macros podem ler variáveis globais, que, por sua vez, podem ser configuradas através do comando “set-prop”, conforme mostrado na Figura 8. Este comando permite adicionar, editar e remover variáveis globais, que podem ser utilizadas por qualquer comando que utilize o serviço interno “TemplateEngine”. Mais detalhes sobre o uso de macros podem ser encontrados na seção 4.4.1 Macros deste capitulo.

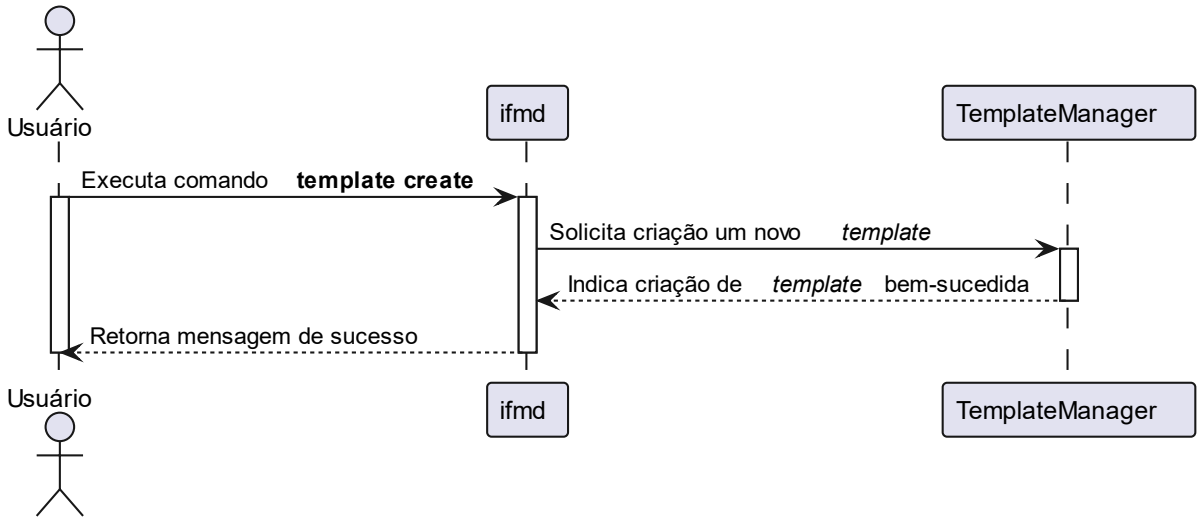
Entrando na parte em que o Designer de *templates* opera exclusivamente, teremos o comando “template”, que possui 7 operações. Como cada operação realiza tarefas bem distintas uma da outra, cada uma terá o seu diagrama de sequência.

Figura 9- Diagrama de Sequência do comando “template list”



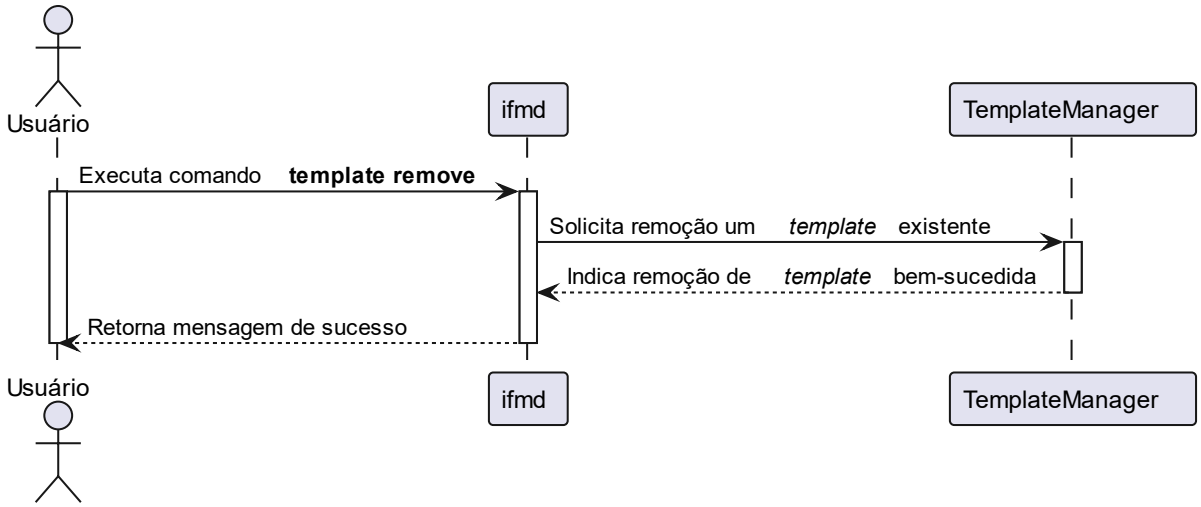
Fonte: Elaborado pelo autor (2024).

Figura 10 - Diagrama de Sequência do comando “template create”



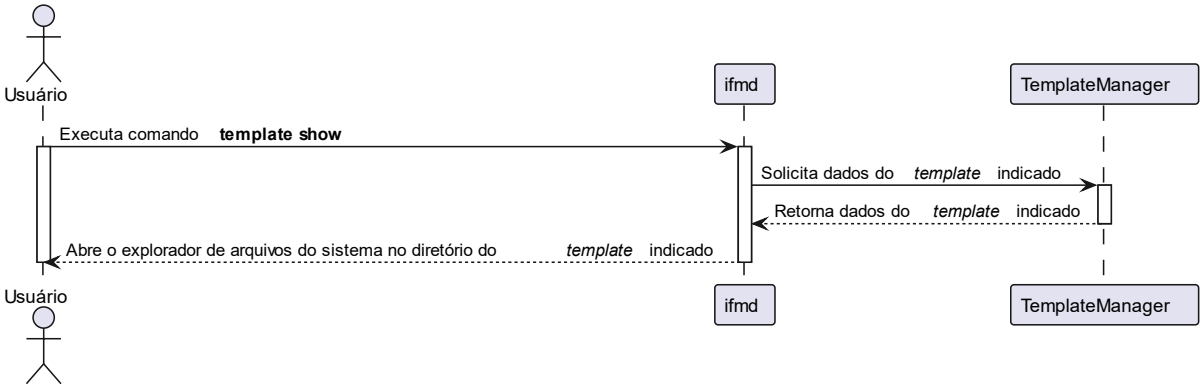
Fonte: Elaborado pelo autor (2024).

Figura 11 - Diagrama de Sequência do comando “template remove”



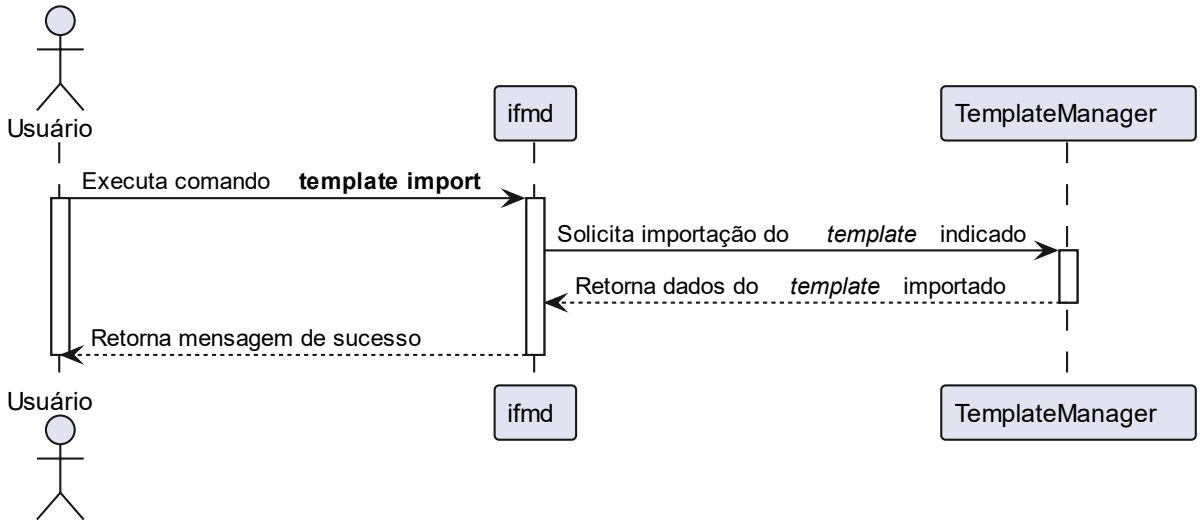
Fonte: Elaborado pelo autor (2024).

Figura 12 - Diagrama de Sequência do comando “template show”



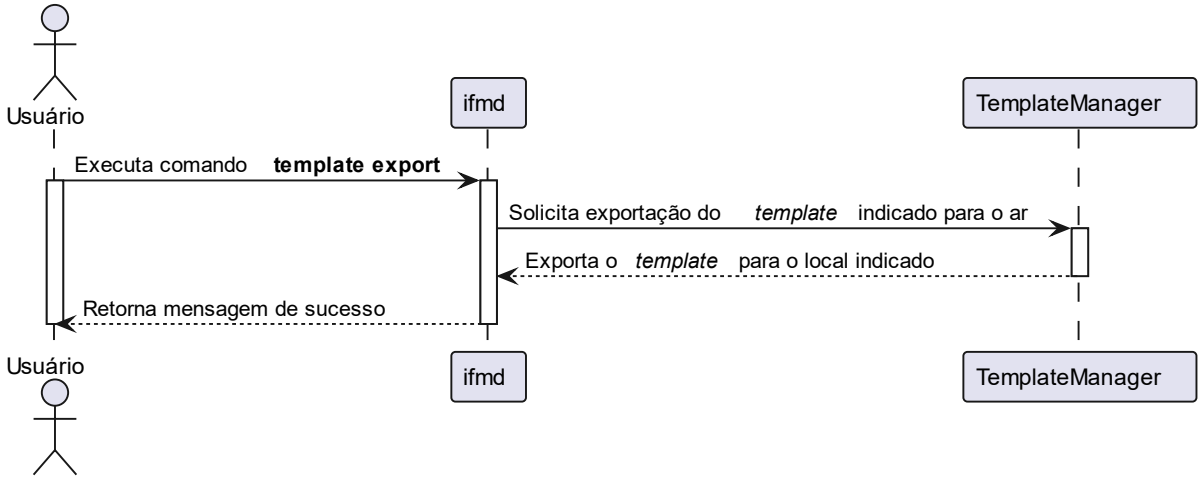
Fonte: Elaborado pelo autor (2024).

Figura 13 - Diagrama de Sequência do comando “template import”



Fonte: Elaborado pelo autor (2024).

Figura 14 - Diagrama de Sequência do comando “template export”

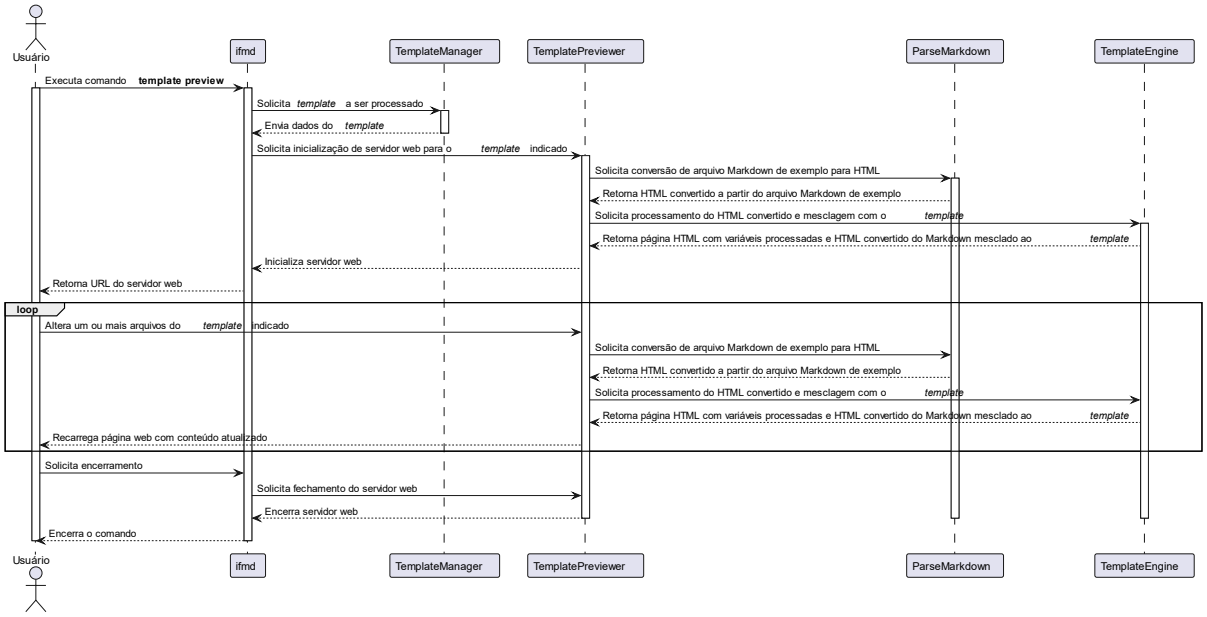


Fonte: Elaborado pelo autor (2024).

As Figuras Figura *9*, Figura *10* e Figura *11* mostram operações de listagem, criação e remoção de *templates*, permitindo que o usuário gerencie de forma eficiente os arquivos de cada template sem a necessidade de trabalhar com o explorador de arquivos de seu sistema. A Figura 12, por outro lado, demonstra a operação de visualização, ou seja, o usuário tem seu explorador de arquivos nativo do sistema aberto diretamente na pasta com os arquivos do *template* em questão, de forma a poder trabalhar nele de forma direta e com ferramentas de sua preferência.

Já a Figuras Figura *13* e Figura *14* envolvem o empacotamento de um *template*. Como os *templates* são pastas com arquivos salvos diretamente na máquina do usuário, transferi-los de forma simples e pratica poderia ser um desafio. Desta forma, as operações “import” e “export” lidarão com esta tarefa.

Figura 15 - Diagrama de Sequência do comando “template preview”



Fonte: Elaborado pelo autor (2024).

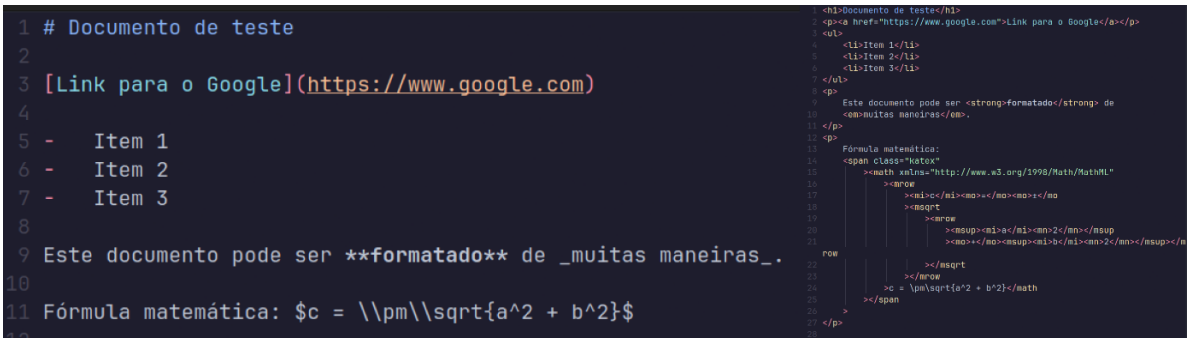
De maneira muito similar ao comando “preview”, visto na Figura 7, a operação “template preview”, com sua sequência de passos diagramada na Figura 15, serve para editar *templates* de forma mais fluida, com a possibilidade de ver as alterações rapidamente em um navegador *web*.

4.2 SERVIÇOS INTERNOS

Após definir o comportamento de cada comando que o sistema deve suportar, o desenvolvimento se deu a partir dos serviços internos da aplicação. Cada serviço e especializado em uma tarefa, tornando o processo de desenvolvimento mais desacoplado e independente.

A linguagem escolhida para o desenvolvimento foi o TypeScript, que adiciona um sistema de tipos ao JavaScript, fornecendo assim um melhor ambiente de desenvolvimento para o programador (TYPESCRIPT, 2024).

Figura 16 - Compilação de Markdown para HTML



Fonte: Elaborado pelo autor (2024).

O primeiro serviço a ser implementado foi o “ParseMarkdown”, que tem como função converter o arquivo de entrada para HTML e associar todas as referências externas para outros arquivos. Para atingir este objetivo, foi utilizada a biblioteca Marked, que foi construída com o objetivo de entregar alta velocidade de conversão de Markdown, além de possuir o mínimo de dependências possíveis em outros projetos (MARKED, 2024). Para suportar fórmulas matemáticas também foi utilizada uma biblioteca de extensão[5], que permite o uso da ferramenta Katex, que é a mais rápida biblioteca de composição matemática para a *web* (EISENBERG; ALPERT, 2024).

A Figura 16 demonstra o serviço em ação, contendo na esquerda um arquivo Markdown e na direita o resultado convertido para HTML, mantendo a estrutura e adicionando os recursos para a visualização de fórmulas matemáticas.

Figura 17 - Resultado da conversão do serviço “ParseMarkdown”

Fonte: Elaborado pelo autor (2024).

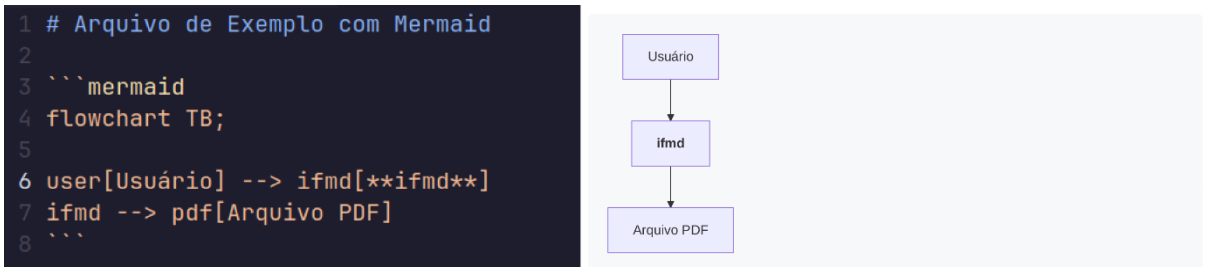
[5] Disponível em https://github.com/UziTech/marked-katex-extension. Acesso em 9 de novembro de 2024.

Com o HTML convertido, foi utilizada a biblioteca Cheerio, que manipula HTML com velocidade e flexibilidade (CHEERIO, 2024), para extrair o título do documento e referências a arquivos externos, como imagens e vídeos. O resultado da conversão pode ser visto na Figura 17, com o Markdown de origem no lado esquerdo, e um arquivo JSON representando o retorno do serviço no lado direito.

Há também, como serviço interno o “ParseMultiMarkdown”, que gerencia a conversão de múltiplos arquivos Markdown e os une. Ele utiliza o “ParseMarkdown” internamente.

A seguir, o serviço “TemplateEngine” foi criado, com o objetivo de mesclar dados de um *template* com o HTML proveniente de um arquivo Markdown. Ele altera qualquer uma referência a arquivos locais por uma URL que será servida via servidor HTTP. Esta medida evita que os arquivos referenciados por meio do arquivo Markdown sejam alterados no navegador, potencialmente acessando arquivos não intencionados pelo escritor da documentação. Como todos os arquivos locais são identificados e referenciados pelo “ParseMarkdown”, conforme visto no campo “localAssets” na Figura 17, nenhum arquivo de fora da lista poderá ser acessado pelo navegador utilizando o servidor HTTP fornecido. Também é neste serviço que o processamento de macros ocorre, permitindo maior dinamicidade a documentação gerada. Mais informações sobre o uso de Macros estão na seção 4.4.1 Macros.

Figura 18 - Exemplo de gráfico gerado com a ferramenta Mermaid



Fonte: Elaborado pelo autor (2024)

Além do processamento de macros e de arquivos locais, o serviço “TemplateEngine” adiciona referências a bibliotecas utilizadas para pré-visualizações, como o Socket.IO, biblioteca de comunicação bidirecional entre cliente e servidor com baixa latência (SOCKET.IO, 2024). Outra biblioteca adicionada e a MermaidJS, que permite a criação de gráficos e diagramas a partir de textos similares ao Markdown (MERMAID, 2024). Um exemplo de gráfico gerado pelo Mermaid pode ser visto na

Figura 18, com o Markdown de origem na esquerda e a seção do PDF com o gráfico gerado na direita.

O próximo serviço desenvolvido foi o “WebServer”, que, a partir de um diretório do sistema, cria um servidor *web* que serve os arquivos do diretório. Ele suporta a utilização de WebSockets, para, no caso de pré-visualizações, recarregar a página principal com o novo conteúdo escrito pelo usuário. Este serviço também gerencia os arquivos referenciados pelo Markdown original, servindo-os conforme são solicitados pelo navegador, a partir do caminho “/__dynamic_assets__/”.

Em seguida o serviço “PdfGenerator” foi concebido. Este serviço se encarrega de converter um *website* para PDF, utilizando um navegador *web*. Para atingir tal feito, a biblioteca Puppeteer foi utilizada, que permite executar comandos em navegadores *web* sem mostrar uma interface gráfica ao usuário (PUPPETEER, 2024). Utilizando a capacidade existente da biblioteca de imprimir páginas *web*, o PDF e criado com sucesso e retornado.

Figura 19 - Representação em JSON de um template

```json
{
  "name": "discurso",
  "path": "/home/giancarl021/.ifmd/templates/discurso",
  "createdAt": "2023-08-01T14:23:54.125Z",
  "isNative": false
}
```

Fonte: Elaborado pelo autor (2024).

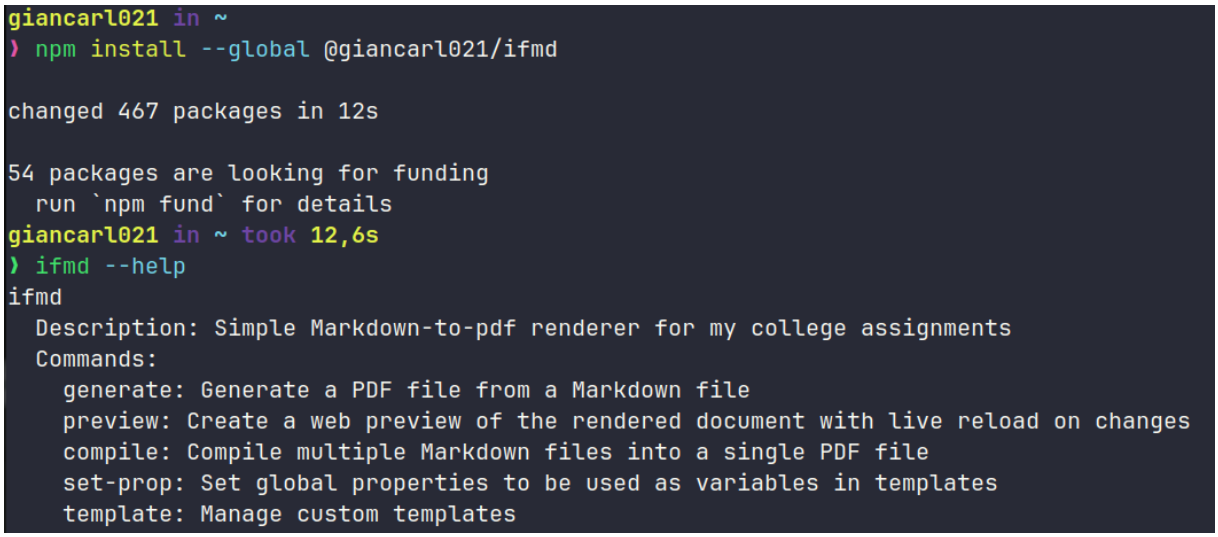
Para gerenciar os *templates*, o serviço “TemplateManager” foi criado, com a capacidade de criar, remover, carregar, importar e exportar *template*. Desta forma foi possível centralizar a gerência interna de arquivos de template em um único serviço, criando uma abstração para os demais componentes usarem. A representação de um *template* pode ser vista na Figura 19.

Para permitir a pré-visualização das documentações em um navegador *web*, os serviços “Previewer” e “TemplatePreviewer” foram criados. Eles gerenciam o estado do servidor *web*, fornecendo os arquivos a serem servidos de acordo com o *template* e documentação apresentados. A diferença entre eles é que o “TemplatePreviewer” lida com mais arquivos que o “Previewer”, visto ter que lidar com mais arquivos de entrada.

4.3 INSTALAÇÃO DO PRODUTO

Para ser possível utilizar o produto, é necessário que o computador do usuário já possua previamente instalado o NodeJS[6]. Este passo e necessário visto que aplicação esta publicada no gerenciador de pacotes do NodeJS, o NPM.

Figura 20 - Instalação do produto



Fonte: Elaborado pelo autor (2024).

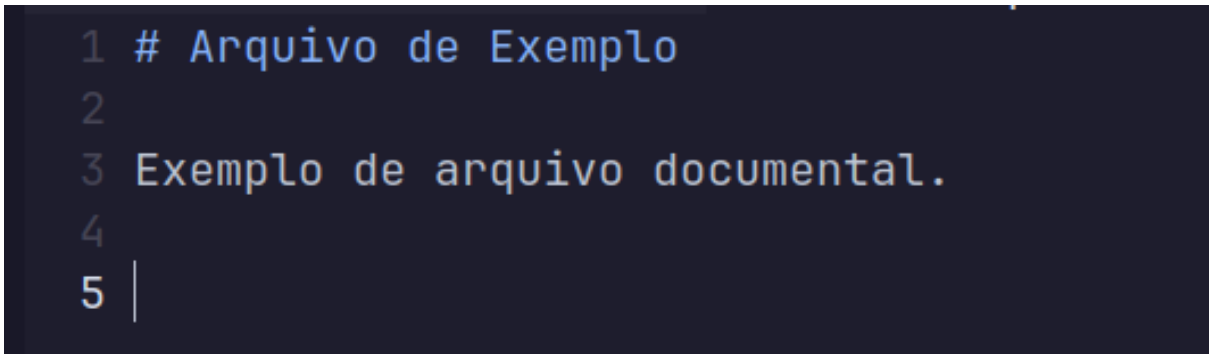
Após a instalação do NodeJS, é possível instalar o projeto utilizando o primeiro comando de terminal demonstrado na Figura 20, que irá baixar e instalar o projeto e todas as dependências necessárias para executá-lo no ambiente do usuário. Após a execução do primeiro comando, o segundo deverá ser executado para assegurar que o pacote foi instalado com sucesso.

4.4 ESCREVENDO DOCUMENTAÇÕES

Após a instalação bem sucedida será possível realizar qualquer operação supracitada. Nesta seção abordaremos o passo a passo para a geração de um documentações com a aplicação, utilizando o *template* padrão da aplicação. Na próxima seção serão abordados os comandos que envolvem o gerenciamento de *templates*, e como utilizá-los.

[6] Disponível para instalação em: https://nodejs.org/pt

Figura 21 - Conteúdo do arquivo “Exemplo.md”



Fonte: Elaborado pelo autor (2024).

Inicialmente será necessário produzir um arquivo em Markdown, que nesta seção se chamará “Exemplo.md” e terá o conteúdo apresentado na Figura 21. Executaremos então o comando “generate --help” e ver o que é permitido fazer com este comando.

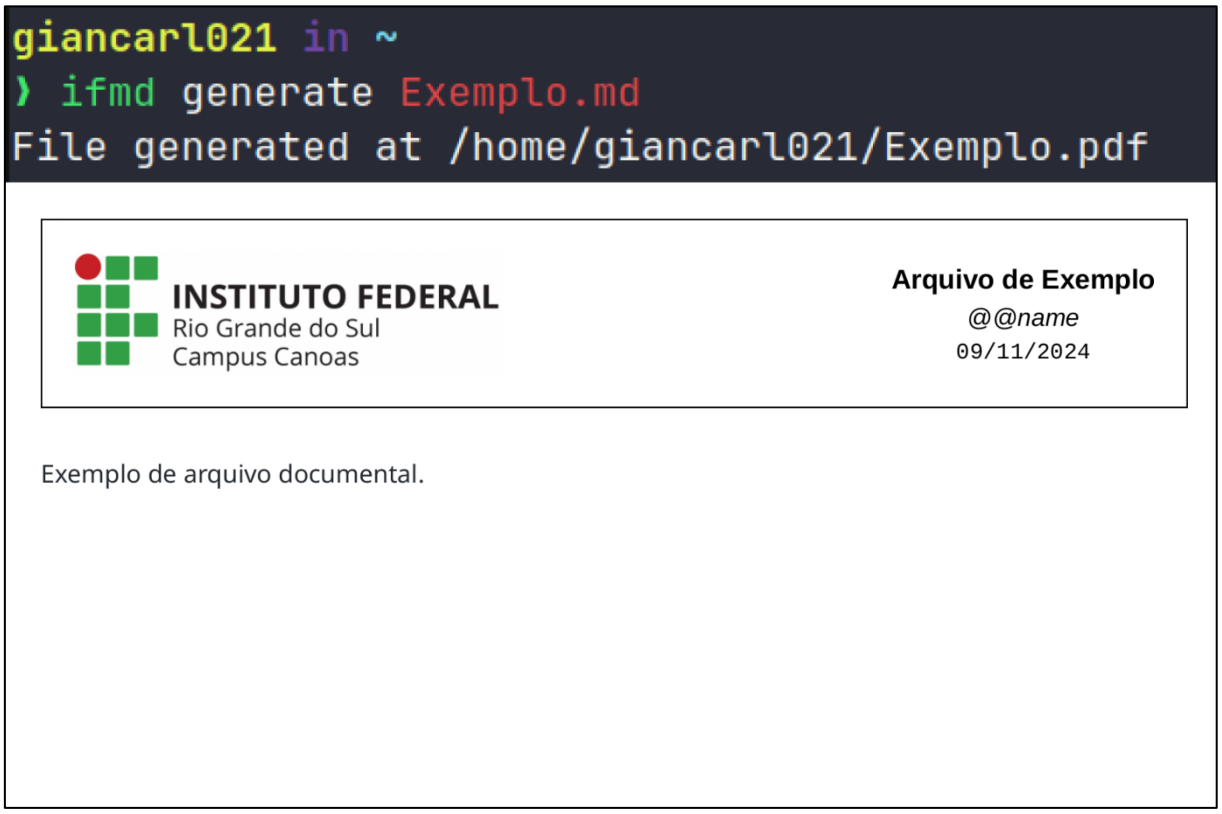
Figura 22 - Ajuda do comando “generate”

Fonte: Elaborado pelo autor (2024).

Na Figura 22 podemos ver que instruções devemos seguir para gerar um arquivo PDF. O único argumento requerido deve conter o caminho para o arquivo Markdown de entrada, além de termos 10 bandeiras disponíveis. Todas as operações do sistema possuem suporte a bandeira “--help”, permitindo que o usuário rapidamente aprenda o objetivo e como utilizar e quais valores de entrada são aceitos

e como são interpretados. Os detalhes de como utilizar e finalidade de cada bandeira podem ser encontrados também no repositório do projeto hospedado no Github[7].

Figura 23 - Demonstração do comando “generate”



Fonte: Elaborado pelo autor (2024).

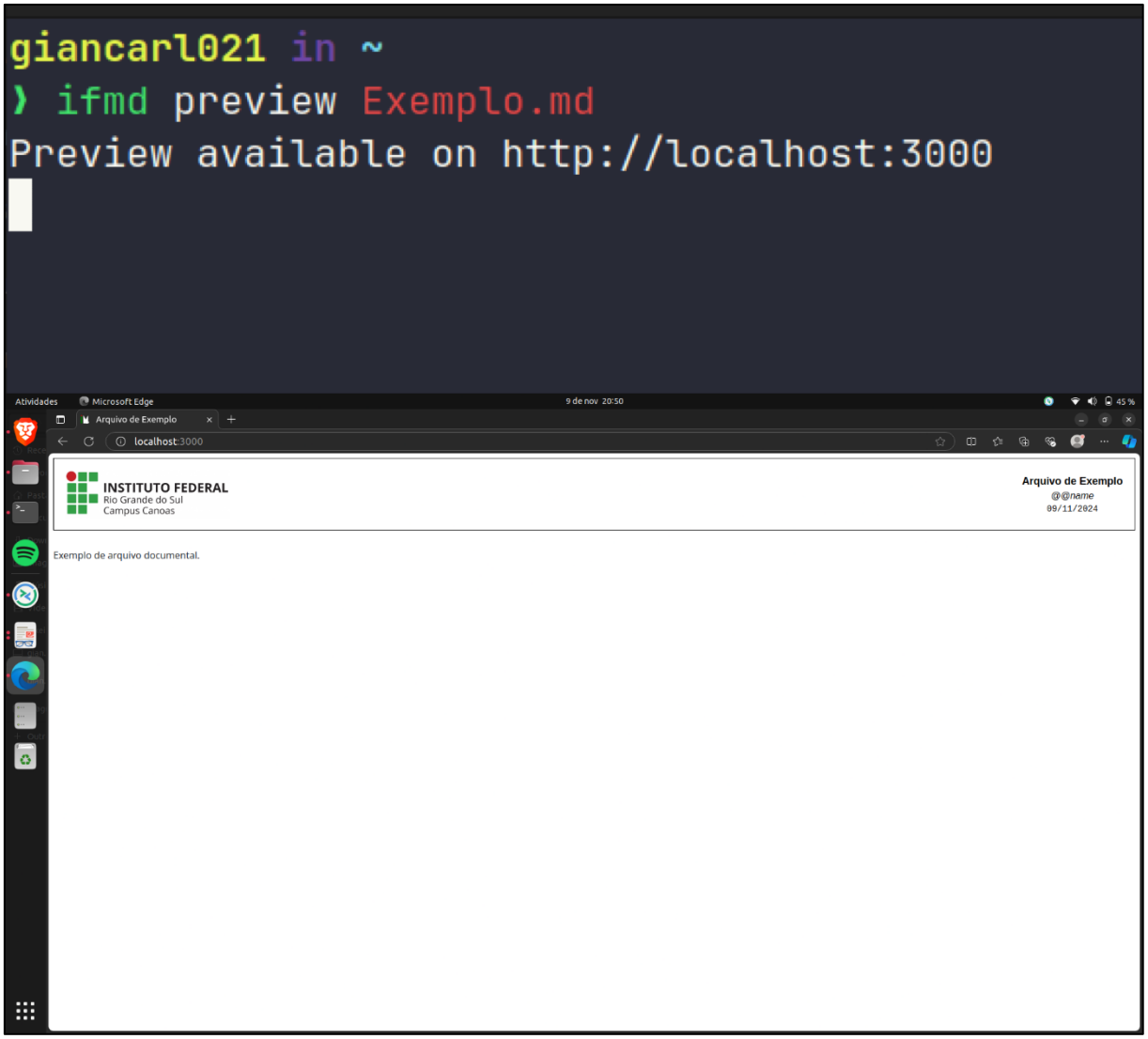
Depois de ler a documentação apresentada, executaremos o comando “ifmd generate Exemplo.md”, aguardando então a execução do código. O resultado pode ser visto na Figura 23, em que acima temos a execução do comando, e em baixo um recorte do documento PDF gerado.

É notável que um cabeçalho da instituição foi adicionado, além de ter o título e a data de criação do documento. Também é possível observar um campo com um valor não usual: “@@name”. Este campo representa uma macro, que veremos na seção 4.4.1 MACROS.

Para se ter uma ideia visual de como a documentação ficara após a geração, podemos executar o comando “ifmd preview Exemplo.md”, que inicializara um servidor *web* com o conteúdo processado do Markdown de entrada.

[7] Disponível em: https://github.com/Giancarl021/ifmd. Acesso em: 9 de novembro de 2024.

Figura 24 - Demonstração do comando “preview”



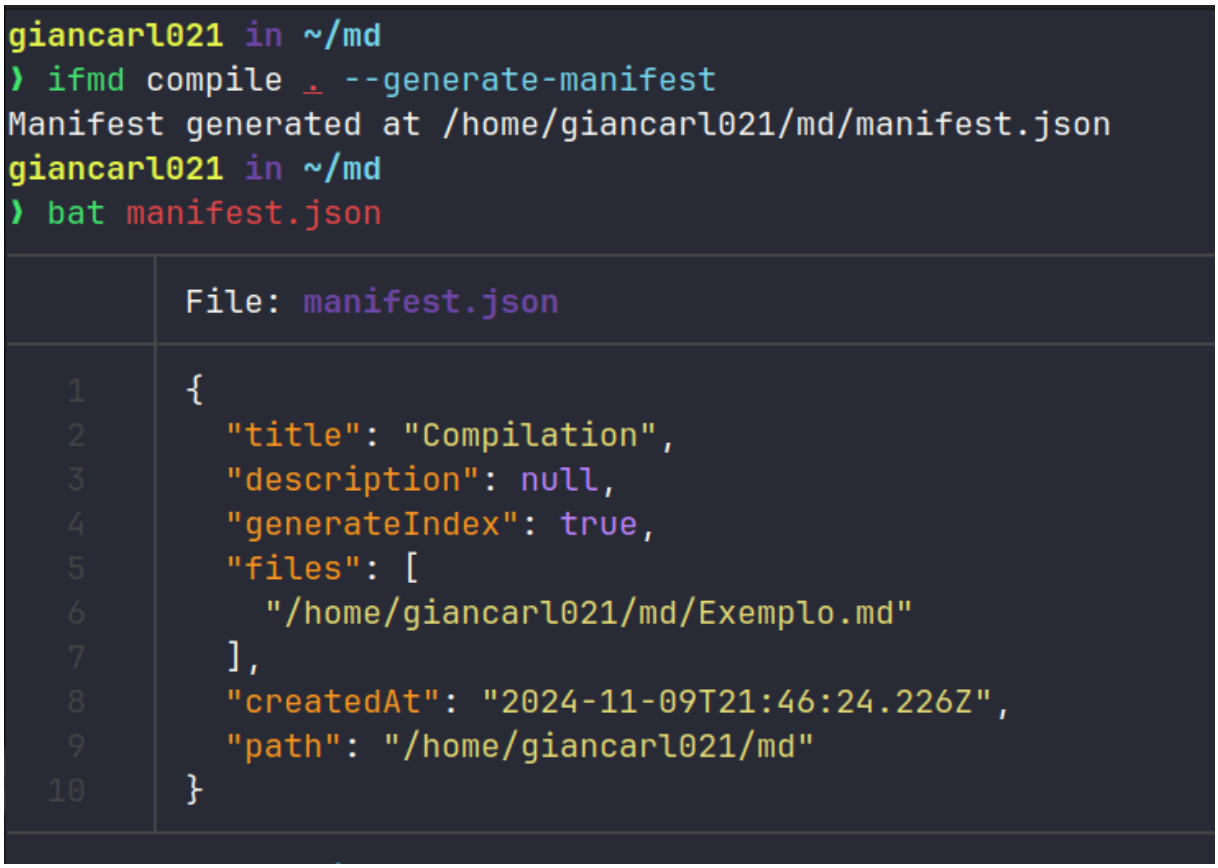
Fonte: Elaborado pelo autor (2024).

Figura 25 - Demonstração de encerramento do comando “preview”

Fonte: Elaborado pelo autor (2024).

Conforme mostrado na Figura 24, o comando iniciou um servidor *web* accessível por meio do endereço mostrado no terminal, na parte de baixo é possível ver o navegador com a página aberta. Nota-se que o comando não finalizou sua execução, para fazer isto é necessário que o usuário aperte a combinação de teclas “Ctrl” e “C”, encerrando assim o servidor web, conforme demonstrado na Figura 25.

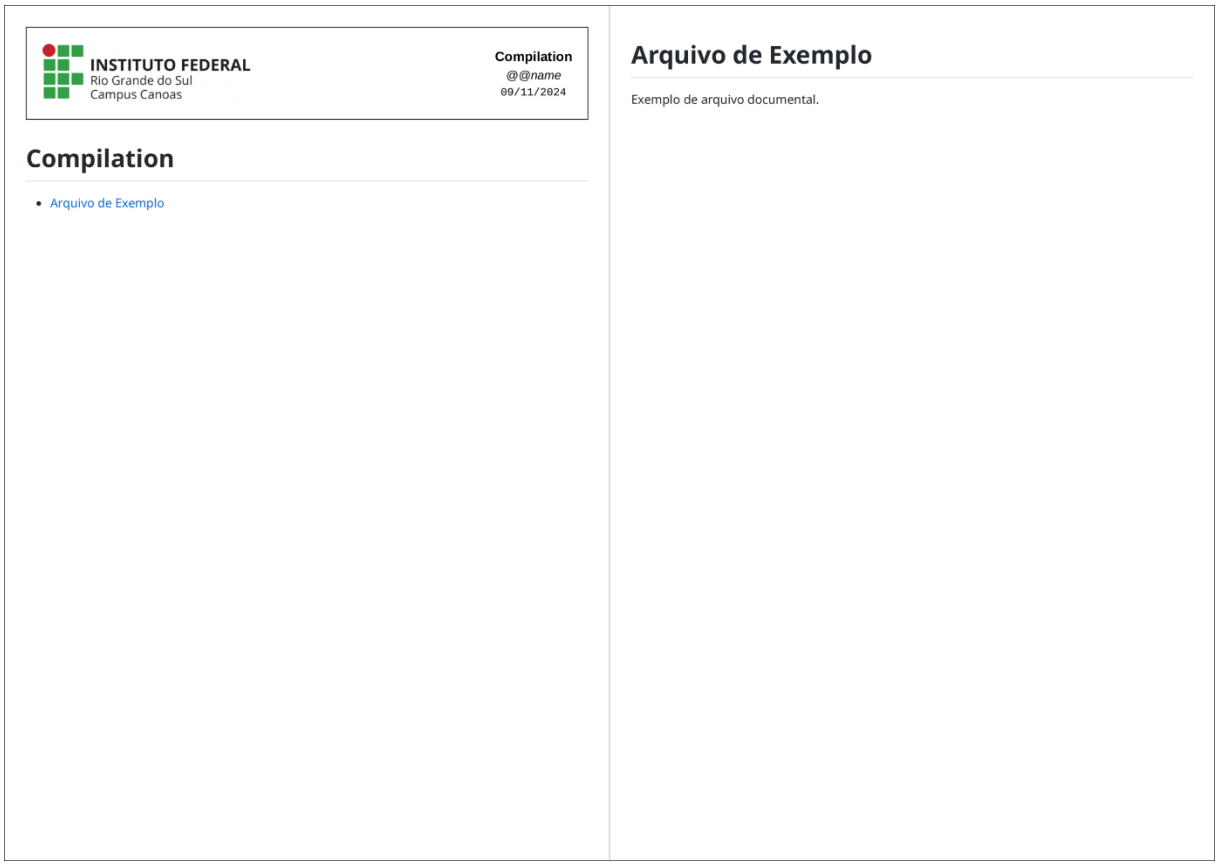
Figura 26 - Demonstração do comando “compile” para criação de um manifesto de compilação



Fonte: Elaborado pelo autor (2024).

Também pode ser de interesse ao usuário compilar diversos arquivos de documentação em um único arquivo PDF. Para tal operação o comando “compile” foi criado. Este comando precisa de duas etapas para ser executado. Sendo a primeira a geração de um arquivo de manifesto de compilação, conforme mostrado na Figura 26. Este arquivo contém instruções sobre como a compilação deve ocorrer, como se um índice deve ser criado, a descrição inicial do documento e a ordem na qual os arquivos encontrados devem ser processados.

Figura 27 - Documento PDF gerado pelo comando “compile”



Fonte: Elaborado pelo autor (2024).

Na Figura 27 temos as páginas do documento PDF com um índice e o conteúdo do arquivo “Exemplo.md” na segunda página.

Há também o comando “set-prop”, que permite a criação, edição e remoção de variáveis globais, para o uso em macros, que serão discutidas na seção 4.4.1 MACROS deste capitulo. Para adicionar ou editar variáveis globais, os argumentos de chave e valor devem estar presentes, respectivamente. Para remover uma variável global, a bandeira “--unset” deve estar presente e o único argumento necessário é o nome da variável.

4.4.1 Macros

As Macros são um componente importante na produção de documentações e *templates*. Todas iniciam com dois caracteres “@” em sequência seguidos de um identificador, que pode ser composto por letras, números e os caracteres “-” e “_” e, opcionalmente, ter parênteses logo após o identificador. Podem ser representadas

literalmente no texto utilizando o caractere “\” logo após as arrobas. Elas possuem 3 tipos: reservadas, funções e variáveis.

As macros reservadas tem um comportamento especifico delas, sendo únicas em relação as outras. As macros “title” e “content” fazem parte desta categoria. A macro “title” sempre terá o valor do primeiro elemento de título do Markdown, e “Trabalho” caso este elemento não exista, não sendo possível alterar este valor de outra maneira. Já a macro “content” só pode ser utilizada dentro de um template, resultando no texto “@@content” caso seja usada em um arquivo de documentação. A razão disto é que ela representa o conteúdo do Markdown a ser inserido no *template*, e, caso ele fosse substituído no próprio template, isto causaria uma recursão infinita que eventualmente causaria uma falha no programa.

Existe duas macros de função: a “date” e a “set”. A “date” por padrão, recebe o valor da data atual no formato “dia/mês/ano” caso o idioma do sistema do usuário seja o português brasileiro, seguindo o formato de data do idioma do sistema em outro caso, podendo ser alterado pela bandeira “--date” nos comandos compatíveis. Ela permite o uso de operações relativas para serem aplicadas ao resultado, com o uso de parênteses após o identificador. Com valores sendo representados como “<operação><valor><unidade>”, como, por exemplo, “+1w” representando “somar 1 semana”. As unidades permitidas são as utilizadas pela biblioteca timestring[8].

Já a macro de função “set” funciona para atribuição de macros de variáveis. Ela macro pode ser usada para configurar o valor de outra macro, salvo as reservadas e de função. Por exemplo: para se criar uma macro com o nome “dia” e valor “Hoje!”, poderia se utilizar a seguinte sintaxe: “@@set(dia, Hoje!)”. Desta forma, qualquer local da documentação e/ou do *template* que utilize a macro de variável “@@dia” teria o valor substituído por “Hoje!”.

Para as macros de variáveis, elas seguem o padrão “@@<nome>”. Elas são identificadoras para serem substituídos pelo projeto no momento de gerar o arquivo PDF. A precedência de valor e a seguinte: variáveis configuradas pelo usuário pela bandeira “—prop:<nome>”, variáveis globais configuradas através do comando “set-prop”, variáveis configuradas pelo arquivo de documentação e variáveis configuradas no arquivo “index.html” do *template*.

[8] Disponível em https://www.npmjs.com/package/timestring. Acesso em: 9 de novembro de 2024.

4.5 CRIANDO *TEMPLATES*

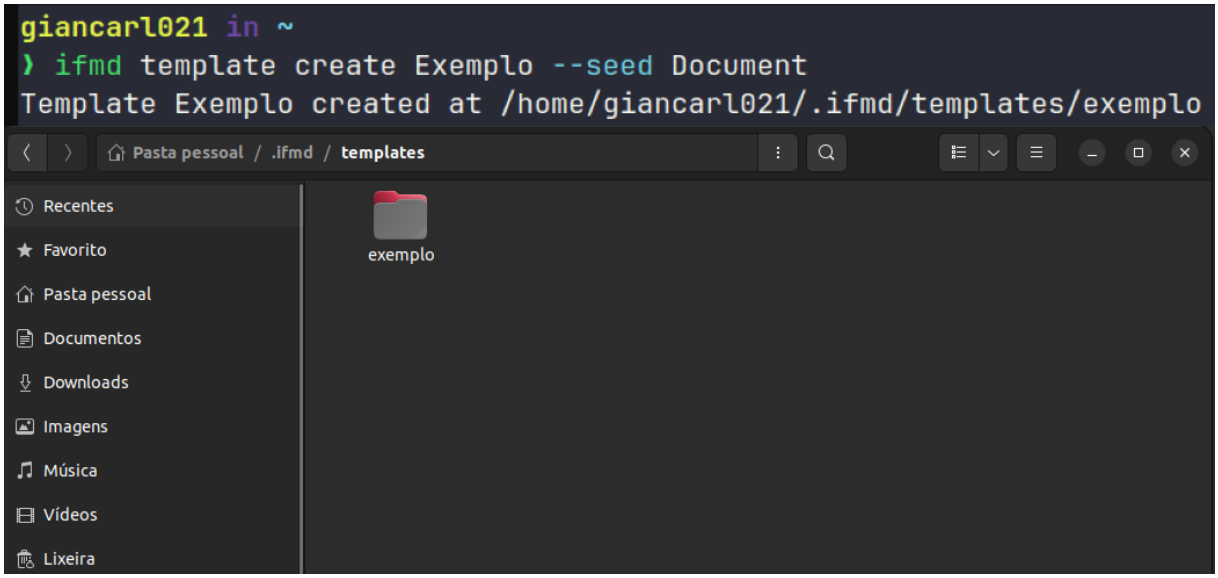
Nesta seção serão abordadas as operações para o gerenciamento dos *templates* da aplicação, envolvendo criação, edição, remoção e empacotamento. Os conceitos estudados na seção anterior serão reaproveitados nesta. As operações descritas serão todas sub-comandos do comando “template”. Existem 7 operações, sendo elas “list”, “show”, “create”, “remove”, “preview”, “import” e “export”.

Figura 28 - Demonstração do comando “template list”

Fonte: Elaborado pelo autor (2024).

A operação “list” mostra todos os *templates* disponíveis para uso no sistema de arquivos local do usuário, conforme demonstrado na Figura 28.

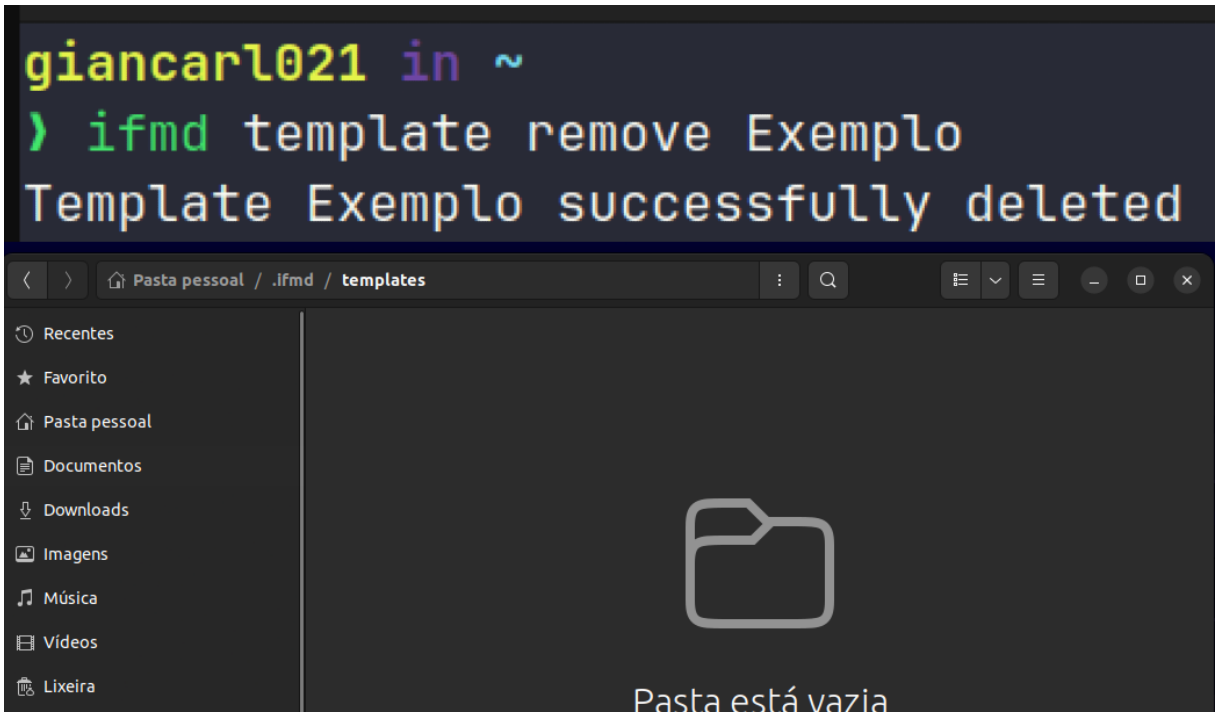
Figura 29 - Demonstração do comando “template create”



Fonte: Elaborado pelo autor (2024).

Já a operação “create” cria um novo *template*, podendo ser baseado em um já existente por meio da bandeira “--seed”. Ao executar este comando, um novo diretório é criado contendo os dados do novo *template*[9]. A Figura 29 demonstra esta operação, com o comando acima e o diretório onde os *templates* são armazenados após a execução do comando na parte inferior da imagem.

Figura 30 - Demonstração do comando “template remove”

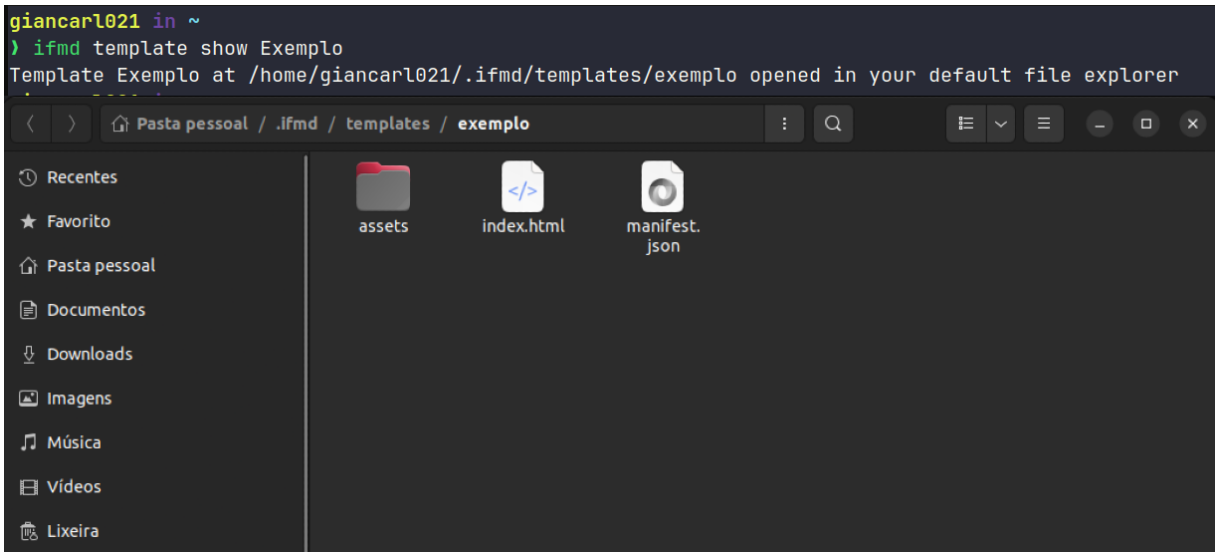


Fonte: Elaborado pelo autor (2024).

Enquanto isto, a operação “remove” faz o inverso, removendo o diretório do sistema de arquivos, conforme demonstração na Figura 30, com o comando acima e o diretório de *templates* abaixo.

[9] Este projeto utiliza como forma de armazenamento apenas o sistema de arquivos do usuário, dispensando a necessidade de um banco de dados.

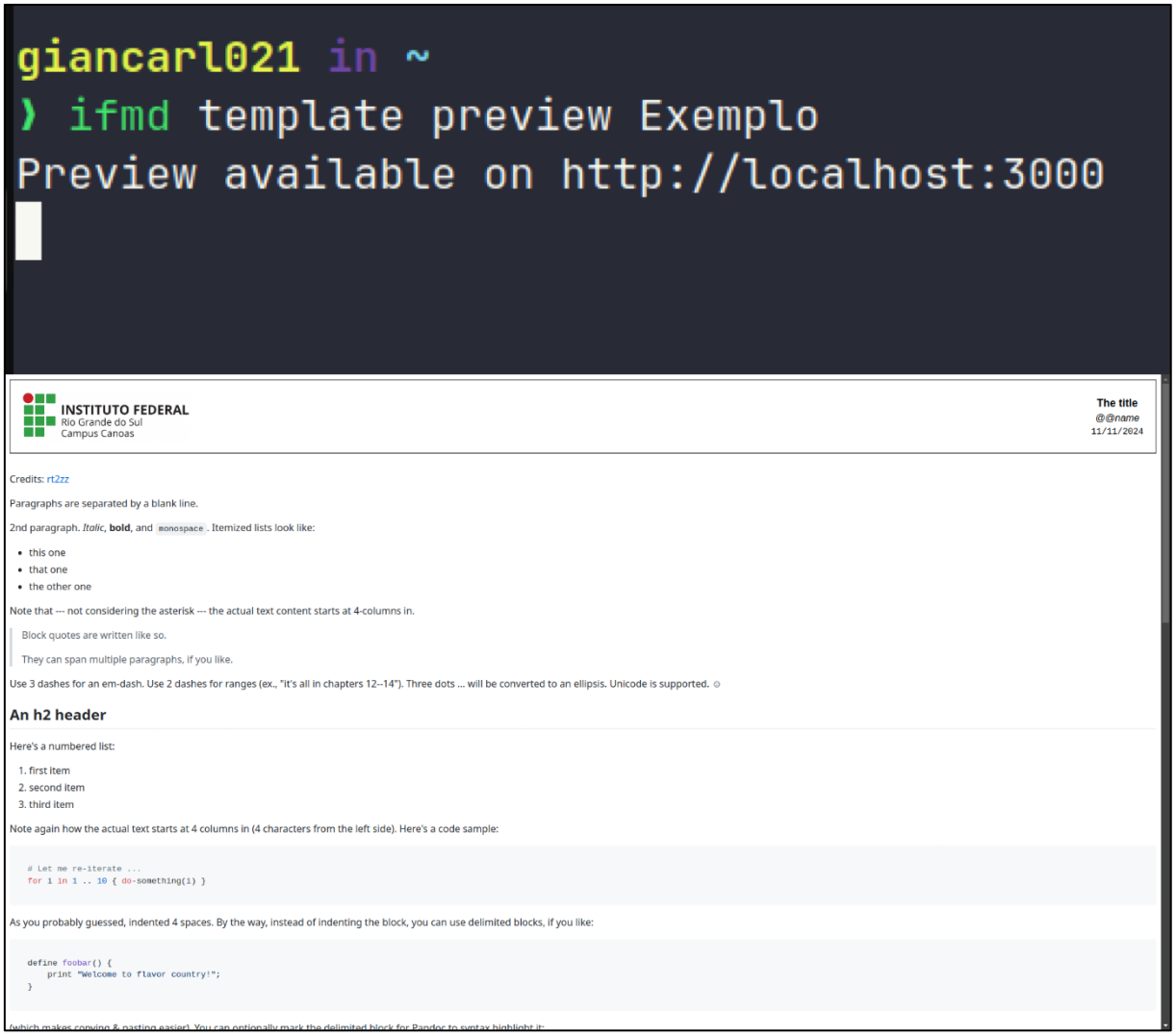
Figura 31 - Demonstração do comando “template show”



Fonte: Elaborado pelo autor (2024).

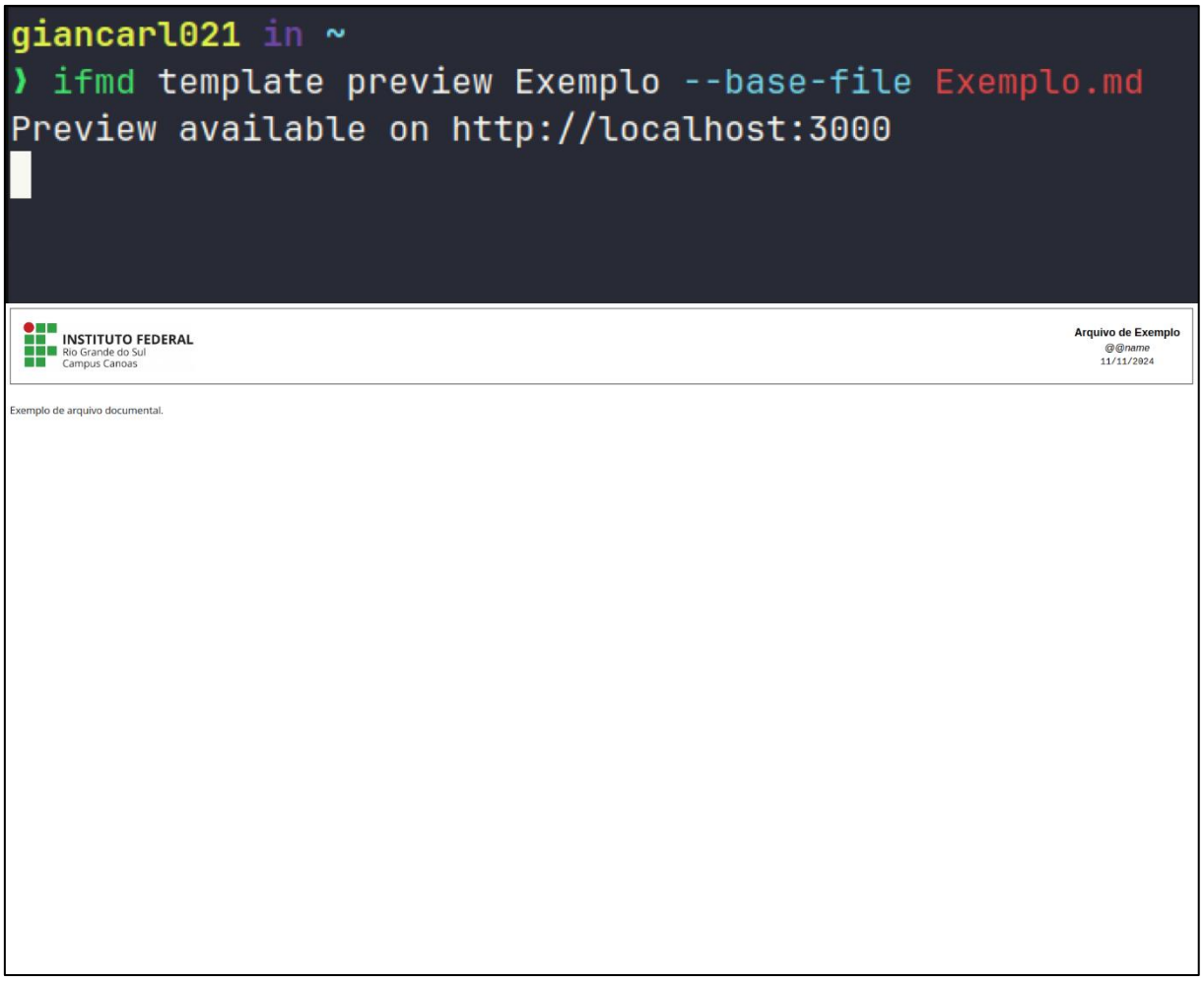
A operação “show” abre o explorador de arquivos do usuário na pasta do *template* escolhido, permitindo facilmente a edição dos arquivos, conforme mostrado na Figura 31, em que o comando está na parte superior da imagem, com o explorador de arquivos do sistema aberto na pasta do *template* na parte inferior.

Figura 32 - Demonstração do comando “template preview”



Fonte: Elaborado pelo autor (2024).

Figura 33 - Demonstração do comando “template preview” com um arquivo Markdown diferente

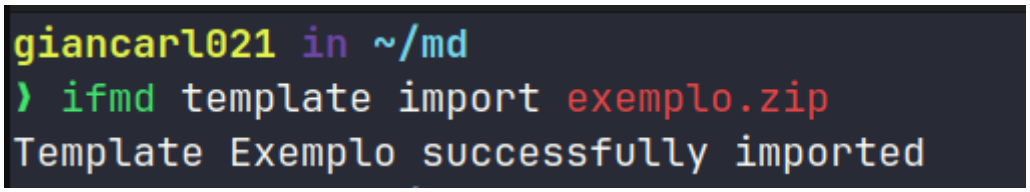


Fonte: Elaborado pelo autor (2024).

A operação “show” pode ser usada em conjunto com a operação “preview”, que funciona de forma similar ao comando “preview” visto anteriormente, porém desta vez agindo em todos os arquivos do diretório do *template*, conforme demonstrado na Figura 32, em que na esquerda temos o comando e na direita o navegador aberto com o *template* em pré-visualização. É possível alterar o arquivo Markdown usado para testar o template utilizando a bandeira “--base-file”, com o valor contendo o caminho do arquivo Markdown desejado, conforme mostrado na Figura 33.

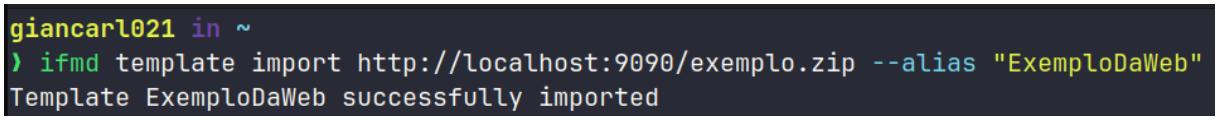
As operações “import” e “export” servem para empacotamento, utilizando o formato de arquivo ZIP, que contém um ou mais arquivos estruturados de forma a ocupar menos espaço (TECH TERMS, 2024).

Figura 34 - Demonstração do comando “template import” a partir de um arquivo



Fonte: Elaborado pelo autor (2024).

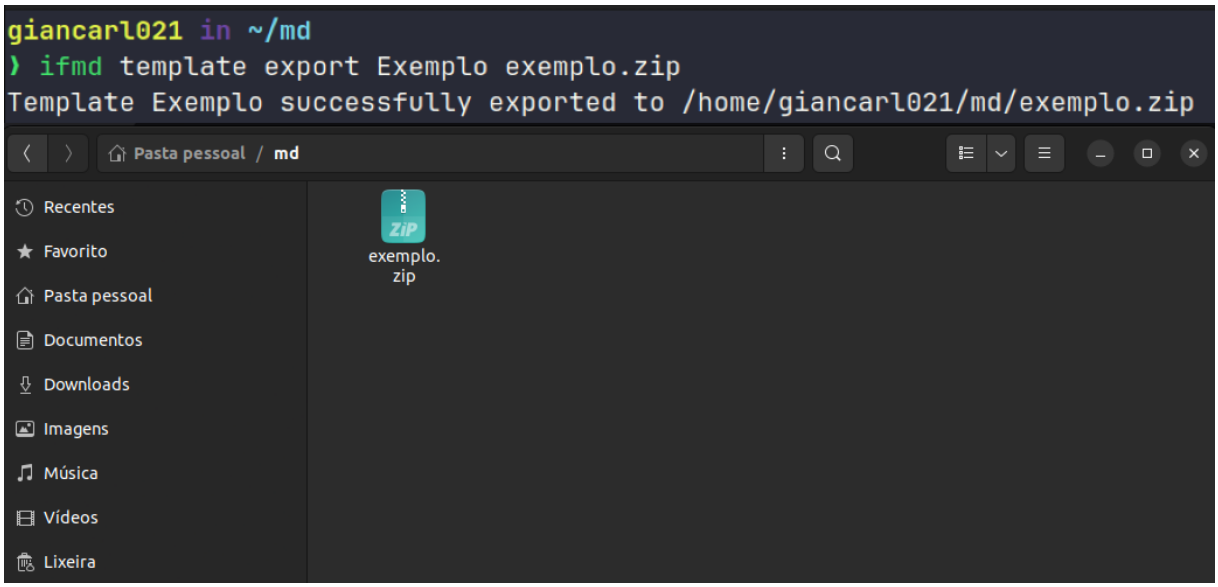
Figura 35 - Demonstração do comando “template import” a partir de uma URL com o uso da bandeira “--alias”



Fonte: Elaborado pelo autor (2024).

A operação “import” recebe como argumento o caminho para o arquivo a ser carregado ou uma URL que retorne um arquivo ZIP. Como resultado o *template* é adicionado a lista de *templates* disponíveis. Caso o nome do *template* sendo importado já exista na lista de templates a operação irá falhar. Para evitar que isto aconteça, e possível utilizar a bandeira “--alias”, contendo um nome diferente para ser utilizado pelo *template* importado. As Figuras Figura 34 e Figura 35 mostram, respectivamente, uma importação utilizando um arquivo e uma importação a partir de uma URL, sendo a última utilizando a bandeira “--alias”.

Figura 36 - Demonstração do comando “template export”



Fonte: Elaborado pelo autor (2024).

Por fim, a operação “export” recebe como argumentos, respectivamente, o nome do *template* a ser exportado como ZIP e o caminho para onde o arquivo deve ser criado. A Figura 36 demonstra esta operação, tendo na esquerda o comando e na direita o explorador de arquivos do sistema mostrando o arquivo gerado na pasta.

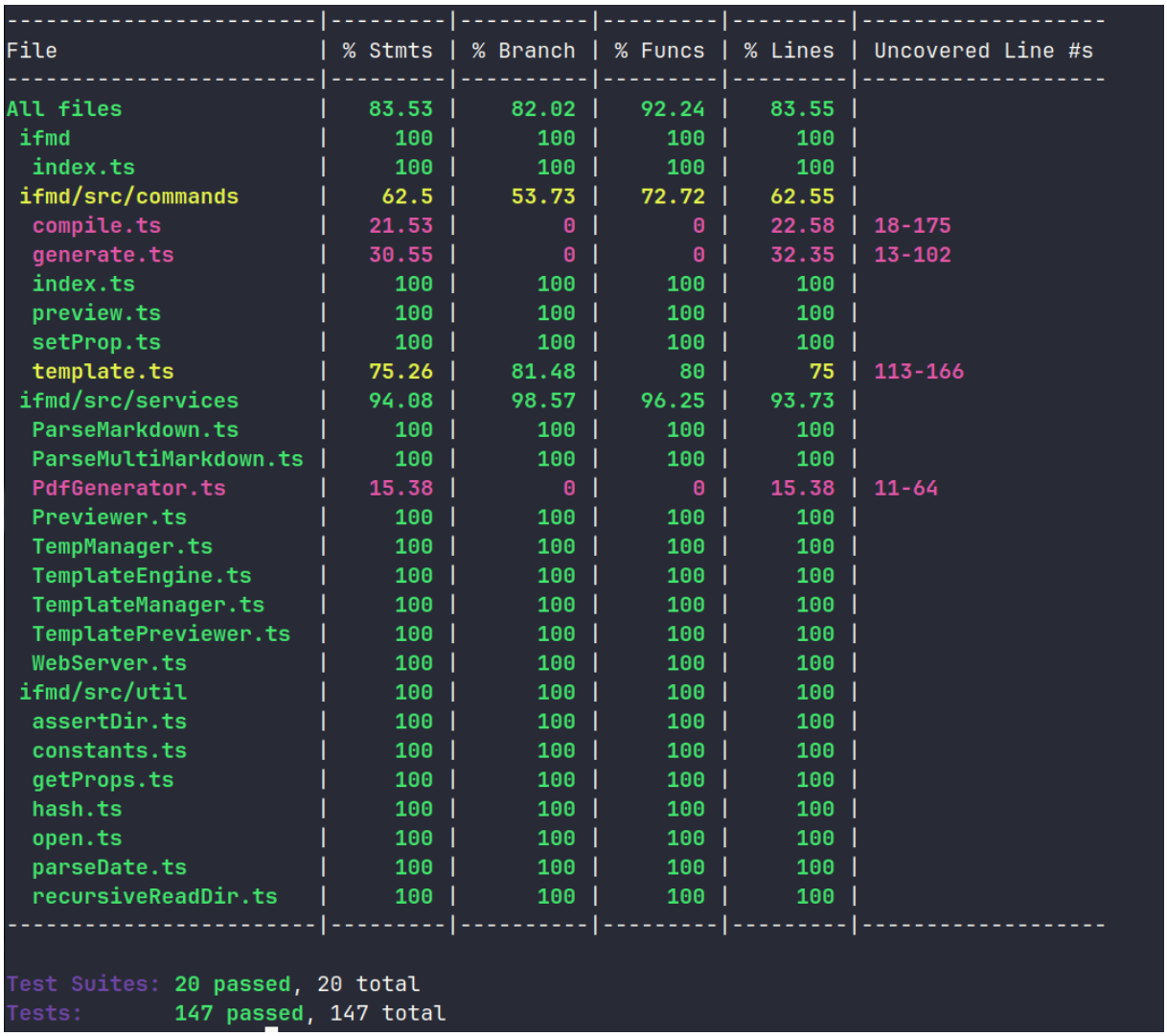
4.6 TESTES

Para verificar as funcionalidades da aplicação, foram realizados testes automatizados e testes com profissionais qualificados, os quais este capitulo abordara com mais detalhes.

4.6.1 Testes automatizados

Os testes automatizados foram feitos com a biblioteca Jest, uma biblioteca focada em simplicidade (JEST, 2024).

Figura 37 - Tabela de cobertura de testes do projeto



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	83.53	82.02	92.24	83.55	
ifmd	100	100	100	100	
index.ts	100	100	100	100	
ifmd/src/commands	62.5	53.73	72.72	62.55	
compile.ts	21.53	0	0	22.58	18-175
generate.ts	30.55	0	0	32.35	13-102
index.ts	100	100	100	100	
preview.ts	100	100	100	100	
setProp.ts	100	100	100	100	
template.ts	75.26	81.48	80	75	113-166
ifmd/src/services	94.08	98.57	96.25	93.73	
ParseMarkdown.ts	100	100	100	100	
ParseMultiMarkdown.ts	100	100	100	100	
PdfGenerator.ts	15.38	0	0	15.38	11-64
Previewer.ts	100	100	100	100	
TempManager.ts	100	100	100	100	
TemplateEngine.ts	100	100	100	100	
TemplateManager.ts	100	100	100	100	
TemplatePreviewer.ts	100	100	100	100	
WebServer.ts	100	100	100	100	
ifmd/src/util	100	100	100	100	
assertDir.ts	100	100	100	100	
constants.ts	100	100	100	100	
getProps.ts	100	100	100	100	
hash.ts	100	100	100	100	
open.ts	100	100	100	100	
parseDate.ts	100	100	100	100	
recursiveReadDir.ts	100	100	100	100	

Test Suites: 20 passed, 20 total
Tests: 147 passed, 147 total

Fonte: Elaborado pelo autor (2024).

A biblioteca fornece um relatório de cobertura do código, mostrando separadamente a porcentagem de instruções, ramificações e linhas cobertas pelos testes, conforme indicado na Figura 37. possível notar que a cobertura está acima dos 80%, com a maioria dos arquivos do projeto totalmente cobertos e 4 arquivos parcialmente cobertos[10]. Estes são testes unitários, que, segundo Sommervile (2013), e o processo de testar cada componente do programa e executá-los com diferentes valores de entrada para verificar sua funcionalidade.

Os 4 arquivos não testados são componentes e comandos que utilizam uma quantidade maior de recursos do sistema do usuário, logo, testá-los de forma automatizada tornou-se impraticável e acabou deixando o ambiente instável, levando a erros imprevisíveis nos testes. Desta forma, estes arquivos foram descartados dos testes unitários, de forma a serem validados com maior precisão pelos testes com usuários.

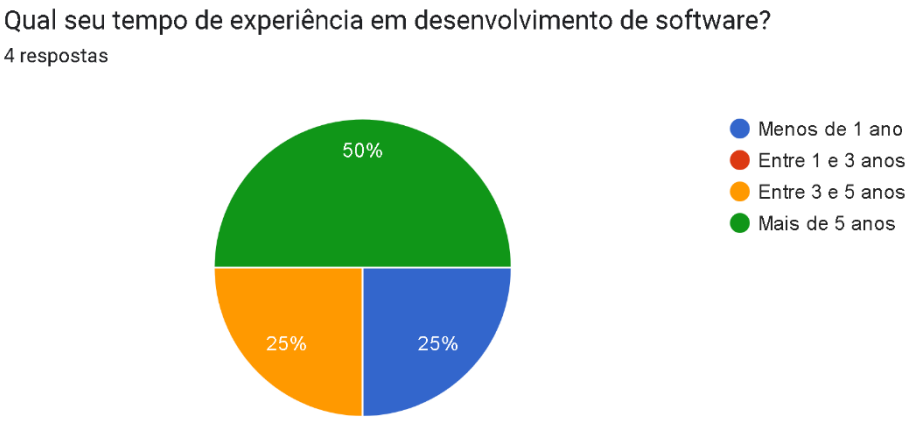
Os riscos de não testar estes arquivos são os de perder algum problema de performance ou de estabilidade nos componentes mais complexos, causando problemas aos usuários finais.

4.6.2 Testes com usuários

Para complementar os testes automatizados, foram realizados testes com usuários qualificados. Para realizar os testes, 4 profissionais qualificados foram escolhidos para instalar e executar o projeto.

[10] Para mais detalhes sobre os testes realizados, o relatório completo de testes está disponível no APÊNDICE D.

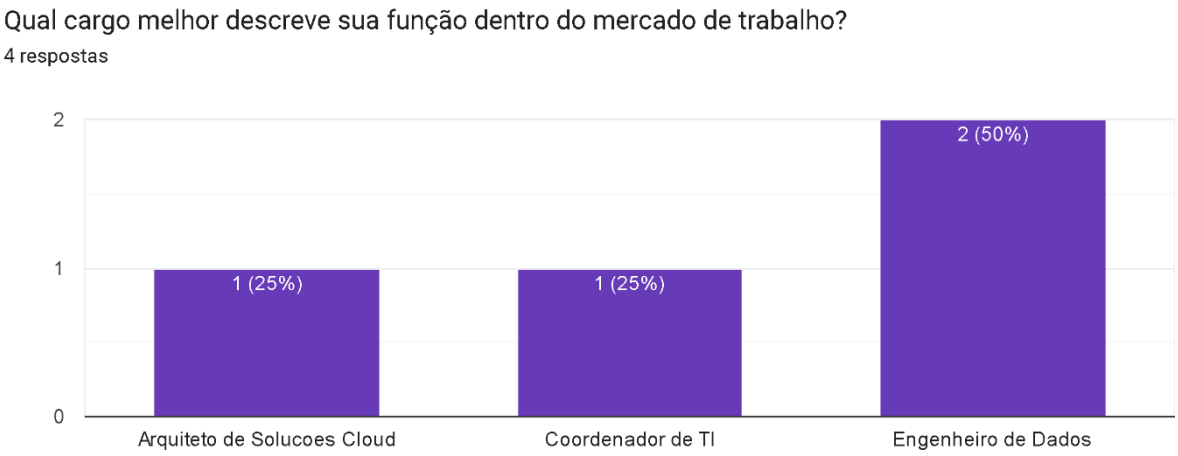
Figura 38 - Gráfico com respostas sobre a primeira pergunta do questionário



Fonte: Elaborado pelo autor (2024).

Conforme podemos notar na Figura 38, metade dos entrevistados possuem bastante experiência na área de desenvolvimento de software, enquanto a outra metade se divide em 3-5 anos e com menos de 1 ano. Sendo assim, a maioria dos entrevistados possuem relativamente bastante experiência com a área.

Figura 39 - Gráfico com respostas sobre a segunda pergunta do questionário



Fonte: Elaborado pelo autor (2024).

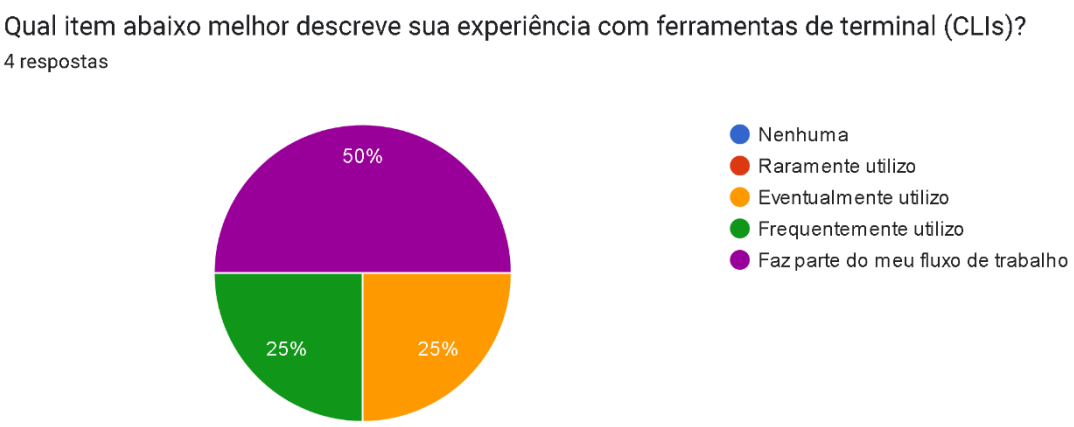
A segunda pergunta também teve resultados diversos, conforme podemos ver na Figura 39, sendo a maioria dos entrevistados engenheiros de dados. Também foram entrevistados um arquiteto de soluções Cloud (nuvem) e um coordenador de TI. Tal diversidade ajuda a entender de forma mais geral se a aplicação cumpre seu objetivo.

Figura 40 - Gráfico com respostas sobre a terceira pergunta do questionário

Fonte: Elaborado pelo autor (2024).

Podemos ver na Figura 40 que todos os entrevistados tem relativamente bastante experiência com as metodologias ágeis, o que atende o objetivo do projeto.

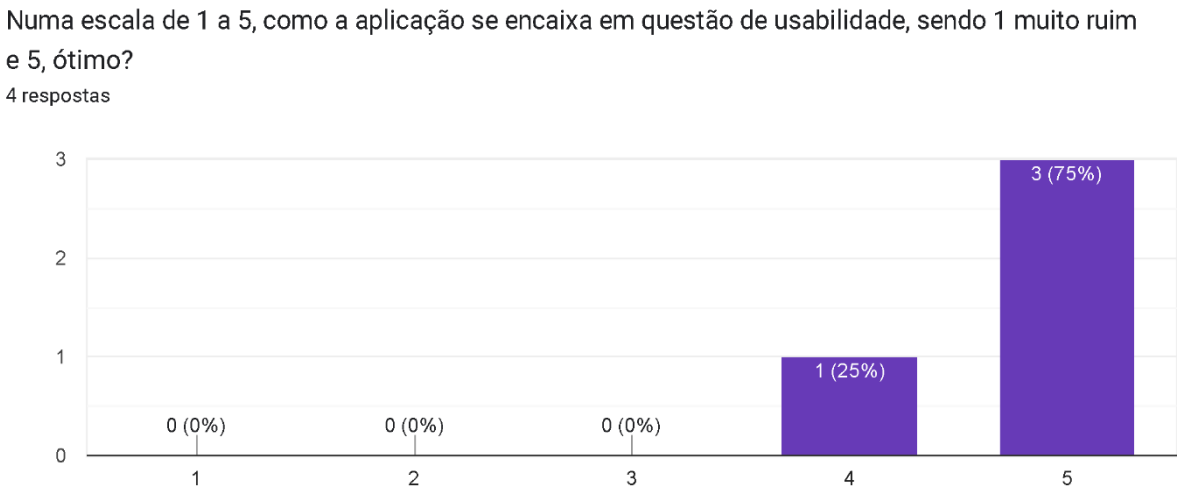
Figura 41 - Gráfico com respostas sobre a quarta pergunta do questionário



Fonte: Elaborado pelo autor (2024).

Conforme a Figura 41, três quartos dos entrevistados possuem bastante experiência com ferramentas de terminal, sendo o público-alvo adequado para a ferramenta.

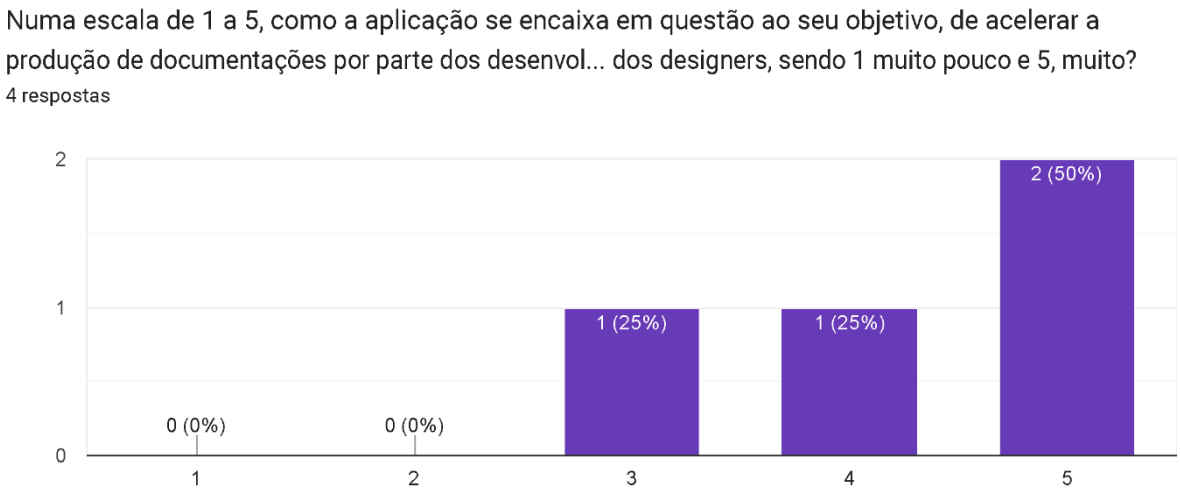
Figura 42 - Gráfico com respostas sobre a quinta pergunta do questionário



Fonte: Elaborado pelo autor (2024).

De acordo com a Figura 42, a maioria dos entrevistados achou ótima a usabilidade do projeto, com um entrevistado considerando bom. Desta forma a aplicação cumpriu seu objetivo dentro do seu público-alvo.

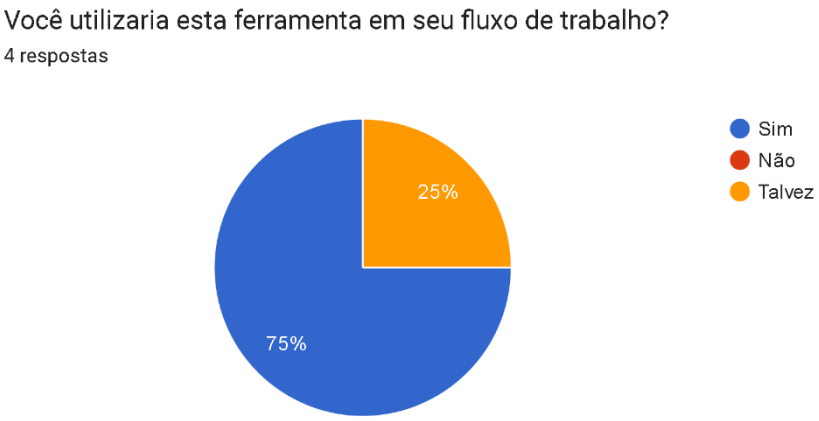
Figura 43 - Gráfico com respostas sobre a sexta pergunta do questionário



Fonte: Elaborado pelo autor (2024).

Como podemos ver na Figura 43, as opiniões sobre o objetivo do projeto e sua execução pendem positivamente, apesar de metade dos entrevistados não concluir que a aplicação entrega perfeitamente o seu objetivo, ainda assim, o resultado indica que a aplicação cumpre com o prometido.

Figura 44 - Gráfico com respostas sobre a sétima pergunta do questionário



Fonte: Elaborado pelo autor (2024).

Na Figura 44 vemos que 75% dos entrevistados utilizariam esta ferramenta em seus fluxos de trabalho, com apenas um deles indicando incerteza.

Um dos desafios encontrados na pesquisa foi o tamanho da amostra. Apesar de todos qualificados, os valores obtidos representam apenas uma fração da comunidade de desenvolvedores.

5 CONSIDERAÇÕES FINAIS

Este trabalho descreveu o desenvolvimento de uma ferramenta para o auxílio na criação de documentações, focada em usuários com conhecimento em programação, de forma a reduzir o tempo gasto para produzir documentações de qualidade, ao mesmo tempo permitindo a personalização de acordo com os interesses da empresa ou equipe envolvida.

A necessidade surgiu ao se analisar os requisitos de documentação em metodologias ágeis, considerando que a documentação, embora importante, não deve consumir mais tempo do que a produção de *software*. Para viabilizar a solução, estudos sobre soluções similares foram feitas, de forma a comparar as opções disponíveis com a proposta do trabalho, e avaliar a necessidade do mesmo, além disso, o perfil do usuário final foi traçado: um desenvolvedor com conhecimentos em desenvolvimento e documentação de *software*, e que possua experiência com aplicações de terminal.

O desenvolvimento foi feito em etapas, com funcionalidades sendo desenvolvidas aos poucos, seguindo o modelo de desenvolvimento cascata, com o objetivo final bem definido. A modelagem foi feita utilizando diagramas UML, de forma a facilitar a visualização dos componentes e suas funções no sistema. Foi utilizado o NodeJS. Desta forma o projeto progrediu de forma linear e teve um resultado previsível.

Para assegurar a qualidade do *software* desenvolvido, testes automatizados e com usuários foram feitos, de forma a dar uma alta cobertura no código em funcionalidades individuais, e ter a segurança que a aplicação cumpre seu objetivo.

Deste modo, considerando os resultados obtidos, se conclui que a proposta deste trabalho foi cumprida. A solução continuará hospedada no Github e no NPM, de forma que outros usuários possam utilizá-la.

Para melhorias futuras, a aplicação poderia ser expandida para possuir uma interface gráfica, de forma a ampliar o público-alvo. Também seria possível encapsular a solução em um container, de forma a aumentar a portabilidade e versatilidade do projeto. Além disso, conforme sugestões dos entrevistados, a ferramenta poderia ser expandida para suportar outros formatos de arquivos, ter integração com projetos em Git, sincronizando *templates* de forma automática, e possuir mais *templates* nativos. Outra sugestão seria integração nativa com editores código, por meio de extensões

ou outros recursos, deixando a experiência de edição mais pratica para os designers de *templates*.

REFERÊNCIAS

ADOBE. **PDF Reference**: Adobe Portable Document Format. Sexta Edição. Disponível em: https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/pdfreference1.7old.pdf. Acesso em: 16 de nov. 2023.

ADOBE. **Waterfall Methodolody**: A Complete Guide. Disponível em: https://business.adobe.com/blog/basics/waterfall. Acesso em: 7 de dez. 2023.

AMAZON. **O que é uma CLI?** Disponível em: https://aws.amazon.com/pt/what-is/cli/. Acesso em: 7 de dez. 2023.

BOOCH, G; RUMBAUCH, J; JACOBSON, I. **The Unified Software Development Process**: The complete guide to the Unified Process from the original designers. Massachusetts: Addison Wesley Longman, Inc, 1999.

CHEERIO. **cheerio**: The fast, flexible & elegant library for parsing and manipulating HTML and XML. Disponível em: https://cheerio.js.org/. Acesso em 9 de novembro de 2024.

COCKBURN, Alistair *et al*. **Manifesto para Desenvolvimento Ágil de Software**. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 7 de dez. de 2023.

EHE, Benjamim *et al*. **CommonMark**. Disponível em: https://commonmark.org/. Acesso em: 7 de dez. de 2023.

EISENBERG, Emily; ALPERT, Sophie. **KATEX**: The fastest math typesetting library for the web. Disponível em: https://katex.org/. Acesso em: 9 de novembro de 2024.

FREECODECAMP. **What Is a File System:** Types of Computer File Systems and How they Work – Explained with Examples, 2022. Disponível em: https://www.freecodecamp.org/news/file-systems-architecture-explained/. Acesso em: 28 de novembro de 2024.

GIL, Antonio C. **Métodos e técnicas de pesquisa social**. São Paulo: Atlas, 1991.

GITHUB. **About**. Disponível em: https://github.com/about. Acesso em: 2 de set. 2023.

JEST. **Jest**: Delightful JavaScript Testing. Disponível em: https://jestjs.io/pt-BR/. Acesso em: 9 de novembro de 2024.

LOSNAK, Giulia. **O que é metodologia ágil?** Disponível em: https://www.alura.com.br/artigos/o-que-e-metodologia-agil. Acesso em: 29 de fevereiro de 2024.

MARKED. **Marked Documentation**. Disponível em: https://marked.js.org/. Acesso em: 9 de novembro de 2024.

MERMAID. **Mermaid**: Diagramming and Charting tool. Disponível em: https://mermaid.js.org/. Acesso em: 10 de novembro de 2024.

MOZILLA. **Estruturando a web com HTML**: Aprendendo desenvolvimento web. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/HTML. Acesso em: 28 de novembro de 2024.

MOZILLA. **WebSockets**: APIs da Web. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/API/WebSockets_API. Acesso em: 28 de novembro de 2024.

NASCIMENTO, Francisco. **Classificação da Pesquisa**: Natureza, método ou abordagem metodológica, objetivos e procedimentos. Brasília: Thesaurus, 2016.

PUPPETEER. **Puppeteer**. Disponível em: https://pptr.dev/. Acesso em: 9 de novembro de 2024.

SOCKETIO. **Socket.IO**. Disponível em: https://socket.io/. Acesso em: 10 de novembro de 2024.

SOMMERVILE, I. **Engenharia de software**. 9ª Edição. São Paulo: Pearson Education do Brasil, dezembro de 2013.

TECH TERMS. **Zip Definition**. Disponível em: https://techterms.com/definition/zip. Acesso em 11 de novembro de 2024.

TYPESCRIPT. **TypeScript**: JavaScript with syntax for types. Disponível em: https://www.typescriptlang.org/. Acesso em: 9 de novembro de 2024.

APÊNDICE A - ESPECIFICAÇÃO DE CASOS DE USO

Pré-visualizar Markdown em página web

Descreve o fluxo que permite o usuário visualizar uma documentação em Markdown em um navegador *web*.

Funcionalidades:

- Visualizar um arquivo de documentação em Markdown em um navegador *web*
- Ter a página *web* recarregada sempre que houver uma alteração no conteúdo do arquivo Markdown

Atores:

- Desenvolvedor

Fluxo Principal (P):

P1. Usuário executa o comando de pré-visualização indicando o arquivo Markdown que deseja pré-visualizar

P2. O sistema inicializa um servidor *web* para servir uma página *web* com o conteúdo do arquivo

P3. O usuário pode, repetidamente, alterar o arquivo enquanto o comando está em execução (A)

P4. O usuário solicita encerramento do comando.

P5. Fim do caso de uso

Fluxo Alternativo (A) – Alteração no arquivo de documentação durante a execução do comando:

A1. O usuário altera o arquivo de documentação indicado

A2. O sistema detecta a alteração e recarrega a pagina no navegador do usuário

A3. Retorno ao fluxo principal

Converter Markdown para PDF

Descreve o fluxo que converte um arquivo de documentação em Markdown para PDF.

Funcionalidades:

- Converter arquivo Markdown para PDF

Atores:

- Desenvolvedor

Fluxo Principal (P):

P1. Usuario executa comando de geração de arquivo PDF indicando o arquivo Markdown que deseja utilizar

P2. O sistema cria um arquivo PDF a partir do arquivo Markdown indicado

P3. Fim do caso de uso

Compilar múltiplos arquivos Markdown para um único PDF

Descreve o fluxo que converte múltiplos arquivos Markdown para um único PDF.

Funcionalidades:

- Criar um arquivo de manifesto contendo os arquivos a serem compilados
- Compilar vários arquivos Markdown para um único arquivo PDF

Atores:

- Desenvolvedor

Fluxo Principal (P):

P1. Usuário executa comando de compilação de múltiplos arquivos Markdown para PDF, sendo possível gerar um manifesto de compilação (A1) ou gerar um arquivo PDF a partir de um manifesto de compilação (A2)

P2. Fim do fluxo

Fluxo Alternativo (A1) – Gerar manifesto de compilação

A1.1. Sistema cria um arquivo de manifesto de compilação contendo referências para os arquivos Markdown encontrado no diretório em que o comando executado.

A1.2 Retorno ao fluxo principal

Fluxo Alternativo (A2) – Gerar PDF a partir de manifesto de compilação

A2.1 Sistema carrega o arquivo de manifesto de compilação encontrado no diretório em que o comando foi executado

A2.2 Sistema gera arquivo PDF de acordo com os arquivos indicados no manifesto de compilação

A2.3 Retorno ao fluxo principal

Configurar variável global

Descreve o fluxo para configurar uma variável global no sistema

Funcionalidades:

- Atribuir uma variável global
- Remover uma variável global

Atores:

- Desenvolvedor
- Designer de *templates*

Fluxo Principal (P):

P1. Usuário executa comando de configuração de variável global

P2. O comando pode conter a bandeira de remoção de variável (A1) ou não (A2)

P3. Fim do caso de uso

Fluxo Alternativo (A1) – Remoção de variável:

A1.1 O sistema remove a variável com o nome indicado pelo usuário

A1.2 Retorno ao fluxo principal

Fluxo Alternativo (A2) – Atribuição de variável:

A2.1 O sistema atribui a variável com o nome indicado pelo usuário o valor indicado pelo usuário

A2.2 Retorno ao fluxo principal

Gerenciar templates

Descreve o fluxo que gerencia *templates* no sistema

Funcionalidades:

- Criar *template*
- Mostrar local de *template* especifico
- Listar *templates*
- Excluir *template*
- Pré-visualizar *template*
- Importar *template*
- Exportar *template*

Atores:

- Designer de *templates*

Fluxo principal (P):

P1. Usuário executa comando para gerenciar *templates* com operação desejada

P2. Sistema ramifica de acordo com a operação indicada: Criação de *template* (A1), mostrar local de *template* (A2), listar *templates* (A3), excluir *template* (A4), pré-visualizar template (A5), importar *template* (A6) ou exportar *template* (A7)

P3. Fim do caso de uso

Fluxo Alternativo (A1) – Criação de *template*:

A1.1 Sistema cria template de acordo o nome indicado pelo usuário

A1.2 Retorno ao fluxo principal

Fluxo Alternativo (A2) – Mostrar local do template:

A2.1 Sistema abre o gerenciador de arquivos do usuário no diretório do *template* indicado pelo usuário

A2.2 Retorno ao fluxo principal

Fluxo Alternativo (A3) – Listar *templates*:

A3.1 Sistema retorna uma lista com todos os templates disponíveis para uso para o usuário

A3.2 Retorno ao fluxo principal

Fluxo Alternativo (A4) – Excluir *template*

A4.1 Sistema remove diretório contendo *template* indicado pelo usuário

A4.1 Retorno ao fluxo principal

Fluxo Alternativo (A5) – Pré-visualizar *template*:

A5.1. O sistema inicializa um servidor *web* para servir uma página *web* com o conteúdo do *template*

A5.2. O usuário pode, repetidamente, alterar os arquivos no diretório do *template* enquanto o comando está em execução (A8)

A5.3. O usuário solicita encerramento do comando.

A5.4. Retorno ao fluxo principal

Fluxo Alternativo (A6) – Importar *template*:

A6.1 Sistema carrega arquivo ZIP com local ou URL indicado pelo usuário

A6.2 Sistema adiciona novo *template* no diretório de *templates*

A6.3 Retorno ao fluxo principal

Fluxo Alternativo (A7) – Exportar *template*:

A7.1 Sistema cria arquivo ZIP do *template* indicado pelo usuário no local indicado pelo usuário

A7.2 Retorno ao fluxo principal

Fluxo Alternativo (A8) – Alteração em arquivo no diretório do *template* durante a execução da operação de pré-visualização:

A8.1. O usuário altera o arquivo no diretório do *template* indicado

A8.2. O sistema detecta a alteração e recarrega a página no navegador do usuário

A3. Retorno ao fluxo principal

APÊNDICE B – FORMULÁRIO DE TESTES COM USUÁRIOS

Seguem abaixo a exportação na íntegra das perguntas e respostas do formulário utilizado com os usuários.

Avaliação de uso do IFMD

Este formulário visa pesquisar com usuários qualificados na área de desenvolvimento de software as funcionalidades e objetivos do Trabalho de Conclusão de Custo de Giancarlo Fontela da Luz (gian.f.luz@hotmail.com) a fim de validar o objetivo do trabalho e suas principais funcionalidades.

O trabalho visa principal desenvolver uma solução que automatize a geração de documentações em formato PDF que se originam de arquivos Markdown para conteúdo e tecnologias *web* (HTML, CSS, JavaScript) para a estilização. Desta forma seria possível desacoplar a estilização das documentações com o seu conteúdo, permitindo maior otimização de tempo por parte do desenvolvedor, ao mesmo tempo que permitiria ampla personalização por parte de *designers*.

Instruções instalação e uso: https://giancarl021disc.blob.core.windows.net/cdn/Instrucoes de Instalacao e Uso - IFMD.pdf
Código-fonte: https://github.com/Giancarl021/ifmd

Antes de responder às perguntas, por favor instale e execute a aplicação conforme as instruções, experimentando livremente suas funcionalidades.

Desde já apreciamos sua participação.

- **Observação:** A aplicação também suporta o uso da ferramenta Mermaid (https://mermaid.js.org/) na geração de documentação, bastando utilizar a linguagem "mermaid" em um bloco de código.

Qual seu tempo de experiência em desenvolvimento de *software?* *

- ○ Menos de 1 ano
- ○ Entre 1 e 3 anos
- ○ Entre 3 e 5 anos
- ◉ Mais de 5 anos

Qual cargo melhor descreve sua função dentro do mercado de trabalho? *

Engenheiro de Dados

Qual item abaixo melhor descreve sua experiência com Metologias Ágeis? *

- ○ Nenhuma
- ○ Consumi conteúdo sobre mas nunca utilizei
- ○ Já trabalhei em alguns projetos
- ○ Frequentemente utilizo
- ◉ É meu principal meio de produzir software

Quanto ao projeto

Esta seção visa abordar a sua experiência utilizando o projeto IFMD.

Qual item abaixo melhor descreve sua experiência com ferramentas de terminal (CLIs)? *

- ○ Nenhuma
- ○ Raramente utilizo
- ○ Eventualmente utilizo
- ○ Frequentemente utilizo
- ◉ Faz parte do meu fluxo de trabalho

Numa escala de 1 a 10, como a aplicação se encaixa em questão de usabilidade, sendo 1 muito ruim e 10 ótimo? *

	1	2	3	4	5	
Muito ruim	○	○	○	○	◉	Muito bom

Numa escala de 1 a 10, como a aplicação se encaixa em questão ao seu objetivo, de acelerar a produção de documentações por parte dos desenvolvedores, ao mesmo tempo que permite personalização por parte dos *designers*, sendo 1 muito pouco e 10 muito? *

	1	2	3	4	5	
Muito pouco	○	○	○	○	◉	Muito

Você utilizaria esta ferramenta em seu fluxo de trabalho? *

- (●) Sim
- ( ) Não
- ( ) Talvez

Você daria alguma sugestão para melhorar a ferramenta?

Achei uma ferramenta boa no que se propoẽ a fazer.

Avaliação de uso do IFMD

Este formulário visa pesquisar com usuários qualificados na área de desenvolvimento de software as funcionalidades e objetivos do Trabalho de Conclusão de Custo de Giancarlo Fontela da Luz (gian.f.luz@hotmail.com) a fim de validar o objetivo do trabalho e suas principais funcionalidades.

O trabalho visa principal desenvolver uma solução que automatize a geração de documentações em formato PDF que se originam de arquivos Markdown para conteúdo e tecnologias *web* (HTML, CSS, JavaScript) para a estilização. Desta forma seria possível desacoplar a estilização das documentações com o seu conteúdo, permitindo maior otimização de tempo por parte do desenvolvedor, ao mesmo tempo que permitiria ampla personalização por parte de *designers*.

Instruções instalação e uso: https://giancarl021disc.blob.core.windows.net/cdn/Instrucoes de Instalacao e Uso - IFMD.pdf
Código-fonte: https://github.com/Giancarl021/ifmd

Antes de responder às perguntas, por favor instale e execute a aplicação conforme as instruções, experimentando livremente suas funcionalidades.

Desde já apreciamos sua participação.

- **Observação:** A aplicação também suporta o uso da ferramenta Mermaid (https://mermaid.js.org/) na geração de documentação, bastando utilizar a linguagem "mermaid" em um bloco de código.

Qual seu tempo de experiência em desenvolvimento de *software?* *

- ○ Menos de 1 ano
- ○ Entre 1 e 3 anos
- ○ Entre 3 e 5 anos
- ◉ Mais de 5 anos

Qual cargo melhor descreve sua função dentro do mercado de trabalho? *

Coordenador de TI

Qual item abaixo melhor descreve sua experiência com Metologias Ágeis? *

- ○ Nenhuma
- ○ Consumi conteúdo sobre mas nunca utilizei
- ○ Já trabalhei em alguns projetos
- ○ Frequentemente utilizo
- ◉ É meu principal meio de produzir software

Quanto ao projeto

Esta seção visa abordar a sua experiência utilizando o projeto IFMD.

Qual item abaixo melhor descreve sua experiência com ferramentas de terminal (CLIs)? *

- ○ Nenhuma
- ○ Raramente utilizo
- ◉ Eventualmente utilizo
- ○ Frequentemente utilizo
- ○ Faz parte do meu fluxo de trabalho

Numa escala de 1 a 10, como a aplicação se encaixa em questão de usabilidade, sendo 1 muito ruim e 10 ótimo? *

	1	2	3	4	5	
Muito ruim	○	○	○	○	◉	Muito bom

Numa escala de 1 a 10, como a aplicação se encaixa em questão ao seu objetivo, de acelerar a produção de documentações por parte dos desenvolvedores, ao mesmo tempo que permite personalização por parte dos *designers*, sendo 1 muito pouco e 10 muito? *

	1	2	3	4	5	
Muito pouco	○	○	○	◉	○	Muito

Você utilizaria esta ferramenta em seu fluxo de trabalho? *

- ◉ Sim
- ○ Não
- ○ Talvez

Você daria alguma sugestão para melhorar a ferramenta?

expandir a ferramenta para outros formatos de arquivos.

Avaliação de uso do IFMD

Este formulário visa pesquisar com usuários qualificados na área de desenvolvimento de software as funcionalidades e objetivos do Trabalho de Conclusão de Custo de Giancarlo Fontela da Luz (gian.f.luz@hotmail.com) a fim de validar o objetivo do trabalho e suas principais funcionalidades.

O trabalho visa principal desenvolver uma solução que automatize a geração de documentações em formato PDF que se originam de arquivos Markdown para conteúdo e tecnologias *web* (HTML, CSS, JavaScript) para a estilização. Desta forma seria possível desacoplar a estilização das documentações com o seu conteúdo, permitindo maior otimização de tempo por parte do desenvolvedor, ao mesmo tempo que permitiria ampla personalização por parte de *designers*.

Instruções instalação e uso: https://giancarl021disc.blob.core.windows.net/cdn/Instrucoes de Instalacao e Uso - IFMD.pdf
Código-fonte: https://github.com/Giancarl021/ifmd

Antes de responder às perguntas, por favor instale e execute a aplicação conforme as instruções, experimentando livremente suas funcionalidades.

Desde já apreciamos sua participação.

- **Observação:** A aplicação também suporta o uso da ferramenta Mermaid (https://mermaid.js.org/) na geração de documentação, bastando utilizar a linguagem "mermaid" em um bloco de código.

Qual seu tempo de experiência em desenvolvimento de *software?* *

- (•) Menos de 1 ano
- ( ) Entre 1 e 3 anos
- ( ) Entre 3 e 5 anos
- ( ) Mais de 5 anos

Qual cargo melhor descreve sua função dentro do mercado de trabalho? *

Arquiteto de Solucoes Cloud

Qual item abaixo melhor descreve sua experiência com Metologias Ágeis? *

- ( ) Nenhuma
- ( ) Consumi conteúdo sobre mas nunca utilizei
- ( ) Já trabalhei em alguns projetos
- (•) Frequentemente utilizo
- ( ) É meu principal meio de produzir software

Quanto ao projeto

Esta seção visa abordar a sua experiência utilizando o projeto IFMD.

Qual item abaixo melhor descreve sua experiência com ferramentas de terminal (CLIs)? *

- ○ Nenhuma
- ○ Raramente utilizo
- ○ Eventualmente utilizo
- ○ Frequentemente utilizo
- ◉ Faz parte do meu fluxo de trabalho

Numa escala de 1 a 10, como a aplicação se encaixa em questão de usabilidade, sendo 1 muito ruim e 10 ótimo? *

	1	2	3	4	5	
Muito ruim	○	○	○	◉	○	Muito bom

Numa escala de 1 a 10, como a aplicação se encaixa em questão ao seu objetivo, de acelerar a produção de documentações por parte dos desenvolvedores, ao mesmo tempo que permite personalização por parte dos *designers*, sendo 1 muito pouco e 10 muito? *

	1	2	3	4	5	
Muito pouco	○	○	◉	○	○	Muito

Você utilizaria esta ferramenta em seu fluxo de trabalho? *

- ○ Sim
- ○ Não
- ◉ Talvez

Você daria alguma sugestão para melhorar a ferramenta?

Melhoria no tutorial para utilizacao da ferramenta

Avaliação de uso do IFMD

Este formulário visa pesquisar com usuários qualificados na área de desenvolvimento de software as funcionalidades e objetivos do Trabalho de Conclusão de Custo de Giancarlo Fontela da Luz (gian.f.luz@hotmail.com) a fim de validar o objetivo do trabalho e suas principais funcionalidades.

O trabalho visa principal desenvolver uma solução que automatize a geração de documentações em formato PDF que se originam de arquivos Markdown para conteúdo e tecnologias *web* (HTML, CSS, JavaScript) para a estilização. Desta forma seria possível desacoplar a estilização das documentações com o seu conteúdo, permitindo maior otimização de tempo por parte do desenvolvedor, ao mesmo tempo que permitiria ampla personalização por parte de *designers*.

Instruções instalação e uso: https://giancarl021disc.blob.core.windows.net/cdn/Instrucoes de Instalacao e Uso - IFMD.pdf
Código-fonte: https://github.com/Giancarl021/ifmd

Antes de responder às perguntas, por favor instale e execute a aplicação conforme as instruções, experimentando livremente suas funcionalidades.

Desde já apreciamos sua participação.

- **Observação:** A aplicação também suporta o uso da ferramenta Mermaid (https://mermaid.js.org/) na geração de documentação, bastando utilizar a linguagem "mermaid" em um bloco de código.

Qual seu tempo de experiência em desenvolvimento de *software?* *

- ○ Menos de 1 ano
- ○ Entre 1 e 3 anos
- ◉ Entre 3 e 5 anos
- ○ Mais de 5 anos

Qual cargo melhor descreve sua função dentro do mercado de trabalho? *

Engenheiro de Dados

Qual item abaixo melhor descreve sua experiência com Metologias Ágeis? *

- ○ Nenhuma
- ○ Consumi conteúdo sobre mas nunca utilizei
- ○ Já trabalhei em alguns projetos
- ○ Frequentemente utilizo
- ◉ É meu principal meio de produzir software

Quanto ao projeto

Esta seção visa abordar a sua experiência utilizando o projeto IFMD.

Qual item abaixo melhor descreve sua experiência com ferramentas de terminal (CLIs)? *

- ○ Nenhuma
- ○ Raramente utilizo
- ○ Eventualmente utilizo
- ◉ Frequentemente utilizo
- ○ Faz parte do meu fluxo de trabalho

Numa escala de 1 a 10, como a aplicação se encaixa em questão de usabilidade, sendo 1 muito ruim e 10 ótimo? *

	1	2	3	4	5	
Muito ruim	○	○	○	○	◉	Muito bom

Numa escala de 1 a 10, como a aplicação se encaixa em questão ao seu objetivo, de acelerar a produção de documentações por parte dos desenvolvedores, ao mesmo tempo que permite personalização por parte dos *designers*, sendo 1 muito pouco e 10 muito? *

	1	2	3	4	5	
Muito pouco	○	○	○	○	◉	Muito

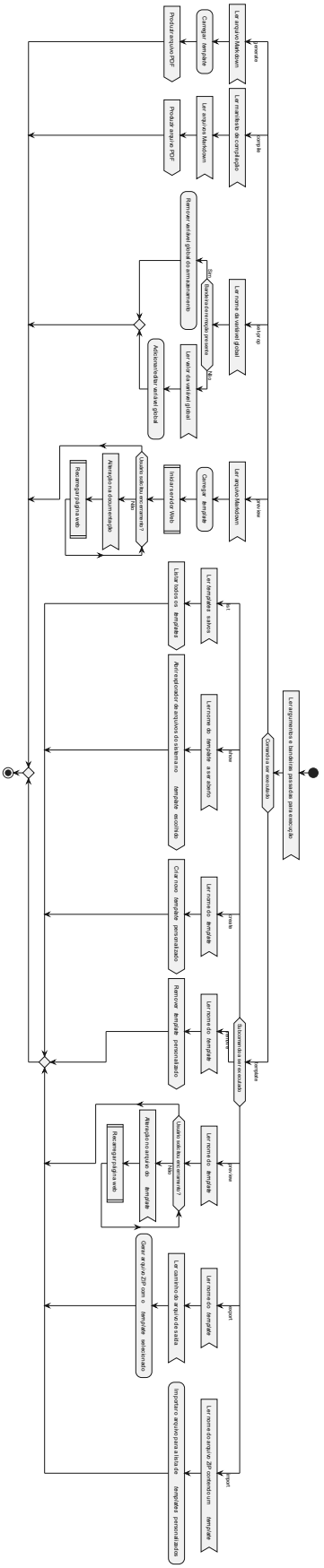
Você utilizaria esta ferramenta em seu fluxo de trabalho? *

- ◉ Sim
- ○ Não
- ○ Talvez

Você daria alguma sugestão para melhorar a ferramenta?

Gostei muito da ideia dos templates, imagino que somente a adição de mais alguns tipos já ficaria bem prático para a utilização de pessoas que não querem entrar em detalhes aprofundados de customização. Não cheguei a testar mas alguma possível integração nativa com repositórios git ajudaria a manter documentações de projetos em dia. Mas seriam melhorias no geral está muito bom!

APÊNDICE C – DIAGRAMA DE ATIVIDADE EM ALTA RESOLUÇÃO



APÊNDICE D – RELATÓRIO DE TESTES

```
PASS tests/index.test.ts (5.053 s)
  app/runner
    + Run in normal mode (2707 ms)
    + Run in debug mode (236 ms)

PASS tests/services/TemplateManager.test.ts
  services/TemplateManager
    + Initialization (81 ms)
    + getDefaultTemplates (66 ms)
    + getCustomTemplates (62 ms)
    + getAllTemplates (67 ms)
    + getTemplate (64 ms)
    + Get invalid template (69 ms)
    + createTemplate (69 ms)
    + Create template with invalid seed (65 ms)
    + deleteTemplate (72 ms)
    + Delete Template invalid template (63 ms)
    + Import file template (82 ms)
    + Import file template on inexistent file (60 ms)
    + Import file template with existing name (71 ms)
    + Import invalid ZIP file (82 ms)
    + Import file template without alias (93 ms)
    + Import URL template (89 ms)
    + Import URL template with non-compliant server (64 ms)
    + Export custom template (71 ms)
    + Export native template (64 ms)
    + Export inexistent template (61 ms)

PASS tests/services/WebServer.test.ts
  services/WebServer
    + Initialization (19 ms)
    + Close without start (14 ms)
    + Start without assets (46 ms)
    + Start with assets (51 ms)
    + Start with sockets (74 ms)

PASS tests/services/Previewer.test.ts
  services/Previewer
    + Initialization (16 ms)
    preview operation
      + Valid template (74 ms)
    update operation
      + Update files (74 ms)

PASS tests/services/TempManager.test.ts
  services/TempManager
    + Initialization (2 ms)
```

```
    + getRootPath (2 ms)
    + create (6 ms)
    + remove (2 ms)
    + clear (3 ms)
    + write (3 ms)
    + read (3 ms)
    + getFilePath (2 ms)
    + fill (14 ms)
    + createDir (3 ms)

PASS tests/commands/setProp.test.ts
  commands/setProp
    + Set property (1 ms)
    + Set invalid key property (4 ms)
    + Set disallowed key property (1 ms)
    + Set empty key property (5 ms)
    + Set empty value property (1 ms)
    + Set property with unset flag (1 ms)
    + Set property with u flag (1 ms)

PASS tests/util/recursiveReadDir.test.ts
  util/recursiveReadDir
    + Empty directory (4 ms)
    + Without filter (3 ms)
    + With filter for .a extension (3 ms)
    + With filter for .b extension (3 ms)
    + Invalid directory (4 ms)

PASS tests/util/assertDir.test.ts
  util/assertDir
    + Existing folder, unaltered (1 ms)
    + Empty folder, create new one (1 ms)
    + Invalid input (4 ms)

PASS tests/util/hash.test.ts
  util/hash
    + Same input, same output (1 ms)
    + Different input, different output
    + Similar input, different output (1 ms)
    + Empty input (1 ms)
    + Invalid input (2 ms)

PASS tests/util/getProps.test.ts
  util/getProps
    + No flags / Default saved props (2 ms)
    + No flags / null saved props (1 ms)
    + With flags / Default saved props (1 ms)
    + No flags / No saved props
    + With flags / No saved props (1 ms)
    + No flags / With saved props (1 ms)
```

```
    + With flags / With saved props [no overwriting] (1 ms)
    + With flags / With saved props [with overwriting]

PASS tests/util/open.test.ts
  util/open
    + Start process on Linux (1 ms)
    + Start process on Windows (1 ms)
    + Start process on MacOS (1 ms)
    + Start process without defining platform (1 ms)

PASS tests/util/constants.test.ts
  util/constants functions
    + compilation/getDefaultManifest (1 ms)
    + webServer/defaultPort

PASS tests/commands/template.test.ts
  commands/template
    list operation
      + Empty execution (36 ms)
      + Verbose execution with v flag (26 ms)
      + Verbose execution with verbose flag (28 ms)
      + JSON execution with json flag (26 ms)
      + JSON execution with j flag (25 ms)
    create operation
      + Empty execution (23 ms)
      + Existing template name (28 ms)
      + Valid execution (38 ms)
      + With custom valid seed on flag s (37 ms)
      + With custom valid seed on flag seed (37 ms)
      + With custom invalid seed on flag s (30 ms)
      + With custom invalid seed on flag seed (32 ms)
    remove operation
      + Empty execution (20 ms)
      + Non-existing template (22 ms)
      + Valid execution (40 ms)
    show operation
      + Empty execution (20 ms)
      + Non-existing template (25 ms)
      + Native template (26 ms)
      + Successful execution (42 ms)
    import operation
      + Empty execution (20 ms)
      + Succesful importing (44 ms)
      + With custom name on flag a (44 ms)
      + With custom name on flag as (45 ms)
      + With custom name on flag alias (43 ms)
    export operation
      + Empty execution (20 ms)
      + Without output file (20 ms)
      + Succesful exporting with filename (42 ms)
```

```
      + Successful export with folder (43 ms)

PASS tests/commands/preview.test.ts
  commands/preview
    + Empty execution (7 ms)
    + Inexistent file (19 ms)
    + With invalid port (121 ms)
    + With default port (72 ms)
    + Default template (77 ms)
    + Custom template (91 ms)
    + Update document (2149 ms)

PASS tests/util/parseDate.test.ts
  util/parseDate
    + Invalid date input (12 ms)
    + Invalid operand input (5 ms)
    + Addition to date (18 ms)
    + Subtraction to date (2 ms)

PASS tests/util/help.test.ts
  util/help
    + help commands matches actual commands (1 ms)

PASS tests/services/TemplatePreviewer.test.ts
  services/TemplatePreviewer
    + Initialization (19 ms)
    preview operation
      + Invalid template (26 ms)
      + Valid template (73 ms)
    update operation
      + Update files (91 ms)

PASS tests/services/TemplateEngine.test.ts
  services/TemplateEngine
    + Initialization (2 ms)
    + generate/Empty content (7 ms)
    + generate/Escaped prop (4 ms)
    + generate/Recursive content (4 ms)
    + generate/All props (4 ms)
    + generate/With mermaid block (8 ms)
    + generate/Unfilled value macro (3 ms)
    + generate/Filled value macro (3 ms)
    + generate/Date displacement macro (3 ms)
    + generate/Date displacement macro with invalid date (5 ms)
    + generate/Set on content (4 ms)
    + generate/Set on content overwritten by props (2 ms)
    + generate/Set on base HTML (2 ms)
    + generate/Set on base HTML overwritten by content (6 ms)
    + generate/Set on base HTML overwritten by props (3 ms)
    + generate/With assets (3 ms)
```

```
    + generateForPreview/Empty content (3 ms)

PASS tests/services/ParseMultiMarkdown.test.ts
  services/ParseMultiMarkdown
    + Initialization (1 ms)
    + compile with no files (1 ms)
    + compile with files (20 ms)
    + compile with no description (1 ms)

PASS tests/services/ParseMarkdown.test.ts
  services/ParseMarkdown
    + Initialization (1 ms)
    + convert/Simple conversion (3 ms)
    + convert/No title conversion (2 ms)
    + convert/Conversion with local assets (4 ms)
    + convert/With math code (10 ms)
    + convertWithMetadata/Simple conversion (2 ms)
    + convertWithMetadata/No title conversion (2 ms)
    + convertWithMetadata/With Title ID (3 ms)

Test Suites: 20 passed, 20 total
Tests:       147 passed, 147 total
Snapshots:   0 total
Time:        15.436 s
Ran all test suites.
```

APÊNDICE E – GUIA RÁPIDO

IFMD - Guia rápido

Giancarlo Luz

17/12/2024

Guia rápido de instalação e utilização do IFMD.

> **Importante:** Mais detalhes de utilização e todas as capacidades do projeto podem ser encontrados no corpo do texto.

Instalação

Para utilizar a aplicação e necessário possuir o NodeJS instalado no dispositivo. As versões do Node e NPM utilizadas no projeto são, respectivamente: 22 LTS e 10, qualquer versão superior deve suportar o projeto.

Para requisitos de *hardware*, deve-se considerar um dispositivo capaz de executar sem dificuldade o navegador web Chrome.

Apos isto, execute o seguinte comando no terminal:

```
npm install --global @giancarl021/ifmd
```

Este comando ira baixar e instalar o projeto no dispositivo, para validar a instalação, execute:

```
ifmd --help
```

O resultado deve ser semelhante a este:

```
ifmd
  Description: Simple Markdown-to-pdf renderer for my college assignments
  Commands:
    generate: Generate a PDF file from a Markdown file
    preview: Create a web preview of the rendered document with live reload on change
    compile: Compile multiple Markdown files into a single PDF file
    set-prop: Set global properties to be used as variables in templates
    template: Manage custom templates
```

Caso tenha interesse, o código-fonte se encontra no GitHub, e o pacote instalado no NPM.

Utilização

Antes de executar alguns comandos do projeto, crie um arquivo Markdown no mesmo diretório em que o terminal esta aberto, como o abaixo (nomeado `Exemplo.md`):

```
# Documento de exemplo

Este documento exemplifica o uso do IFMD.
```

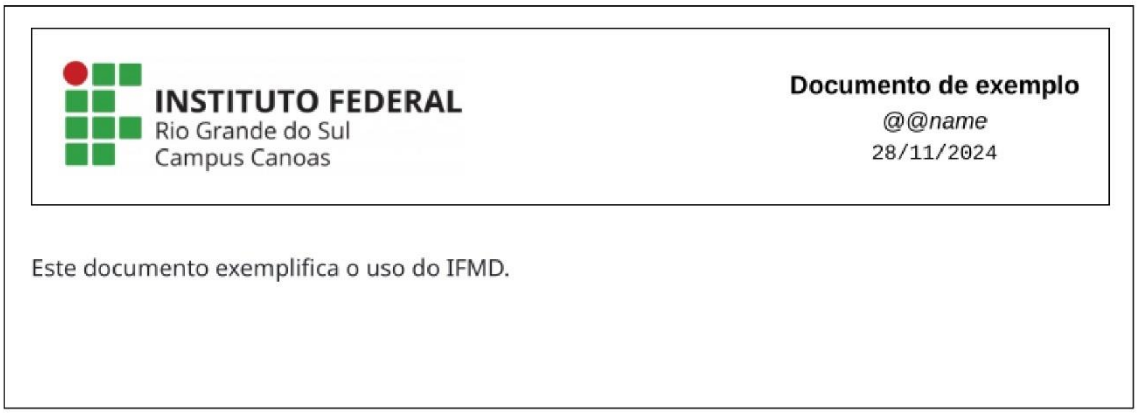
Para criar um PDF a partir deste arquivo, execute o seguinte comando:

```
ifmd generate Exemplo.md
```

A saída deve ser similar a esta:

```
File generated at /home/giancarl021/tcc/Exemplo.pdf
```

O arquivo PDF deve ser similar a esta imagem:



Para preencher o valor de `@@name`, vamos executar o seguinte comando:

```
ifmd set-prop name "Giancarlo Luz"
```

E executar o comando anterior novamente.

O novo PDF agora deve ter o valor `Giancarlo Luz` onde antes ficava `@@name`.