INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL
CAMPUS RIO GRANDE
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

VINÍCIUS PATZDORF GOMES

Markdemic: Um editor Markdown colaborativo para a produção de documentos científicos

Trabalho de Conclusão de Curso
apresentado como requisito parcial
para a obtenção do grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas

Prof. André Vinícius dos Santos
Orientador

Rio Grande, dezembro de 2019

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Patzdorf Gomes, Vinícius

Markdemic: Um editor Markdown colaborativo para a produção de documentos científicos / Vinícius Patzdorf Gomes. – Rio Grande: Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, 2019.

72 f.: il.

Trabalho de Conclusão de Curso (tecnólogo) – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Rio Grande, BR–RS, 2019. Orientador: André Vinícius dos Santos.

1. Transpiladores. 2. Markdown. 3. Latex. 4. Edição Colaborativa. I. dos Santos, André Vinícius. II. Título.

FOLHA DE APROVAÇÃO

Monografia sob o título *"Markdemic: Um editor Markdown colaborativo para a produção de documentos científicos"*, defendida por Vinícius Patzdorf Gomes e aprovada em 10 de dezembro de 2019, em Rio Grande, RS, pela banca examinadora constituída pelos professores:

Prof. André Vinícius dos Santos
Orientador

Prof. Luciano Vargas Gonçalves
IFRS - Campus Rio Grande

Prof. Rafael Betito
IFRS - Campus Rio Grande

"Não deixem que lhe façam pensar que você não é capaz de fazer algo porque essa pessoa não consegue fazer. Se você deseja alguma coisa, se quer realmente, lute por isso e ponto final."

— A Procura da Felicidade

AGRADECIMENTOS

Agradeço a minha família, amigos e colegas por estarem presentes em toda a trajetória do curso, dando apoio e me incentivando. Agradeço especialmente aos meus pais, Isabel Cristina Damasceno Patzdorf e Paulo Renato Mattos Gomes e à minha namorada Vitória Santos Avila, que desde o início torceram muito para que eu chegasse até aqui. Agradeço aos meus professores por me passarem um pouco de seus conhecimentos e experiências, o que contribuiu de forma muito significativa para o meu desenvolvimento pessoal e profissional. Um agradecimento em especial a meu orientador André Vinícius dos Santos, que me incentivou e ajudou no decorrer deste trabalho com tudo que esteve ao seu alcance.

RESUMO

Com a evolução das ciências e tecnologias e o aumento no número de trabalhos nas áreas de pesquisa, surgiu a necessidade de uma escrita mais ágil e eficiente, que fosse capaz de facilitar a aplicação de regras de formatação em documentos. Atrelado a isso, veio também a necessidade de se trabalhar cooperativamente na produção acadêmica, trazendo os autores para um ambiente de edição colaborativa em tempo real. Editores LaTeX colaborativos cumprem esse papel, no entanto, muitos usuários – em sua maioria pessoas de fora das áreas de tecnologia – os abandonam ao esbarrarem na dificuldade encontrada no aprendizado da linguagem, optando por um sistema mais convencional. Dessa forma, este trabalho tem por objetivo desenvolver um sistema que utilize como base para a escrita, uma linguagem de marcação com uma menor curva de aprendizado, como o Markdown – considerada simples e popular – sem abrir mão dos recursos de edição colaborativa síncrona, tornando-se uma alternativa às ferramentas LaTeX e aos editores colaborativos convencionais existentes.

Palavras-chave: Transpiladores. Markdown. Latex. Edição Colaborativa.

LISTA DE SIGLAS E ABREVIATURAS

ABNT	Associação Brasileira de Normas Técnicas
API	Interface de Programação de Aplicativos (*Application Programming Interface*)
ASCII	Código Padrão Americano para o Intercâmbio de Informação (*American Standard Code for Information Interchange*)
CRUD	Criação, Leitura, Atualização e Remoção (*Create, Read, Update and Delete*)
CSCL	Aprendizagem Colaborativa Suportada por Computador (*Computer Supported Collaborative Learning*)
DTO	Objeto de Transferência de Dados (*Data Transfer Object*)
HMAC	Código de Autenticação de Mensagens Baseado em hash (*Hash-based Message Authentication Code*)
HTML	Linguagem de Marcação de Hipertexto (*Hypertext Markup Language*)
HTTP	Protocolo de Transferência de Hipertexto (*Hyper Text Transfer Protocol*)
HTTPS	Protocolo de Transferência de Hipertexto Seguro (*Hyper Text Transfer Protocol Secure*)
JSON	Notação de Objeto JavaScript (*JavaScript Object Notation*)
JWT	JSON Web Token
MVP	Produto Mínimo Viável (*Minimum Viable Product*)
NPM	Gerenciador de Pacotes do Node(*Node Package Manager*)
ODT	Documento de Texto Aberto (*Open Document Text*)
PDF	Formato de Documento Portátil (*Portable Document Format*)
POSIX	Interface Portável Entre Sistemas Operacionais (*Portable Operating System Interface*)
REGEX	Expressão Regular (*Regular Expression*)
REST	Transferência Representacional de Estado (*Representational State Transfer*)
SaaS	Software como um Serviço (*Software as a Service*)
SGBD	Sistemas de Gestão de Banco de Dados (*Data Base Management System*)
URL	Localizador Padrão de Recursos (*Uniform Resource Locator*)
WYSIWYG	O que você vê é o que você obtém (*What You See Is What You Get*)

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com o passar dos anos e a evolução das ciências e tecnologias, cresceu exponencialmente o número de publicações científicas de pesquisa. Atrelado a isso veio a necessidade de se obter maior agilidade durante o processo de escrita, portanto, novas ferramentas começaram a surgir. Processadores de texto cada vez mais poderosos e com novos recursos foram aparecendo, tornando o processo de produção científica algo muito mais desembaraçado. Ademais, linguagens como o LaTeX se tornaram populares para este fim, pelo fato de abstraírem os autores de cuidados com a formatação de seus trabalhos.

E como a elaboração de documentos acadêmicos geralmente possui mais de uma pessoa envolvida – ao menos um autor e um orientador – nasceu também a necessidade da edição síncrona em conjunto. Com isso, seguindo a onda de crescimento da *web*, foram sendo apresentados novos processadores de texto colaborativos, como o Overleaf – editor LaTeX web-colaborativo – software que une a agilidade do LaTeX com a interação social cooperativa.

No entanto, nem todas as pessoas são capazes de escrever textos em linguagem LaTeX, visto que para tal, exige um certo grau de conhecimento em programação e, infelizmente, existem poucas alternativas a essa linguagem no mercado.

1.1 Motivação

Conforme comentado no último parágrafo, existe uma carência no mercado por ferramentas colaborativas, voltadas para a produção de documentos científicos, que utilizem como base uma linguagem de marcação diferente do LaTeX. Em vista disso, criou-se o fator motivacional de desenvolver uma plataforma semelhante às atualmente existentes, mas que, por outro lado, tenha como base uma linguagem de marcação considerada mais simples, mas ao mesmo tempo, popular.

Sendo assim, optou-se por utilizar o Markdown como sendo a linguagem base do sistema, visto que esta possui todas as características que motivaram a elaboração do presente trabalho.

1.2 Objetivos

O objetivo geral deste trabalho é estudar, projetar e desenvolver um protótipo capaz de gerar documentos científicos formatados, a partir da edição colaborativa, em tempo real, de arquivos somente texto, em linguagem Markdown, sendo desmembrado nos seguintes objetivos específicos:

- Estudar a estrutura de trabalhos científicos;

- Pesquisar os aspectos humanos e tecnológicos, relacionados à edição colaborativa;
- Estudar implementações de escrita colaborativa;
- Estudar todas as características do Markdown;
- Pesquisar sobre os sistemas existentes relacionados;
- Projetar o sistema;
- Criar uma API de transpilação de linguagem Markdown para linguagem LaTeX;
- Desenvolver extensão para o editor de textos Visual Studio Code;
- Integrar os módulos desenvolvidos;
- Realizar um estudo de caso utilizando este próprio TCC;
- Concluir o trabalho e expor os resultados obtidos.

1.3 Metodologia

A metodologia utilizada para o desenvolvimento deste trabalho consiste em:

1. Leitura de artigos, livros e publicações, a fim de absorver conhecimento a cerca dos temas relacionados ao presente trabalho;
2. Leitura de materiais disponibilizados na web, como guias, manuais e documentações oficiais de sistemas existentes e tecnologias necessárias;
3. Desenvolvimento de um estudo sobre aplicações existentes;
4. Levantamento de requisitos e possíveis limitações;
5. Definição de tecnologias a serem utilizadas, que possam atender às necessidades;
6. Elaboração do *backlog* de atividades a serem executadas durante o processo de desenvolvimento;
7. Definição de cronograma de tarefas, a fim de cumprir com todas as atividades do *backlog*;
8. Processo de implementação do sistema;
9. Avaliação dos resultados através do estudo de caso.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata dos conceitos teóricos para o entendimento do trabalho, com intuito de expor uma visão geral acerca do processo formatativo de textos científicos, da edição colaborativa e da linguagem de marcação Markdown, bem como expor seus benefícios, seus contrapontos e suas técnicas.

2.1 Estrutura e formatação de documentos científicos

O texto final de um trabalho científico, além de seguir as normas técnicas da ABNT[1], têm estrutura e formatação em conformidade com os padrões gerais de todo trabalho científico, complementadas por eventuais diretrizes próprias da instituição de ensino do curso (SEVERINO, 2017). De um modo geral, um trabalho científico contém:

- Capa inicial;
- Página de rosto;
- Sumário;
- Lista de tabelas e figuras;
- Núcleo do trabalho;
  - Introdução;
  - Desenvolvimento;
  - Conclusão.
- Apêndices e anexos;
- Bibliografia;
- Capa final ou quarta capa.

Os microcomputadores são hoje ferramentas fundamentais para agilizar o processo de formatação dos trabalhos acadêmicos, pois, além de auxiliarem no processo de pesquisa, com eles é possível realizar uma configuração prévia da formatação do documento, de modo a se ganhar um tempo considerável, antes despendido com cuidados com tipologia, espaçamentos, margens, entrelinhamentos, numeração de páginas, entre outros (SEVERINO, 2017).

[1] Associação Brasileira de Normas Técnicas

Com o passar dos anos e o crescimento do uso dos microcomputadores no processo da produção científica acadêmica, novas necessidades começaram a surgir, como por exemplo a necessidade da edição de documentos de forma cooperativa, pois estudos apontam que aproximadamente oitenta e cinco porcento dos documentos produzidos em universidades têm ao menos dois autores (EDE; EDE; LUNSFORD, 1990).

2.2 Edição Colaborativa

Naturalmente com o aumento da utilização dos computadores pessoais para escrita e edição de textos científicos, a necessidade de que se fosse possível editar tais documentos de maneira mais dinâmica e plural também se fez necessária. Junto da popularização da internet novas maneiras de trabalhar surgiram, e uma delas é a edição colaborativa.

Mas de fato o que vem a ser edição colaborativa e onde ela se encaixa? A edição colaborativa, da forma como é concebida hoje, basicamente é um arquivo digital que está hospedado em um servidor e que pode ser acessado através de uma aplicação web permitindo a sua edição por vários colaboradores. Em uma definição mais técnica, pode-se dizer que a edição colaborativa é a designação atribuída a um conjunto de aplicações que permite a várias pessoas editar um arquivo em computadores diferentes (MARTINS et al., 2008). Atualmente existe uma vasta gama de provedoras desse serviço, dentre elas pode-se citar: Google Docs, Overleaf, HackMD, Dropbox Paper, Word Online, entre muitos outros de menor expressão. É importante salientar que a edição colaborativa já se estende para além da edição de textos, artigos, planilhas e apresentações, pois já existem diversas outras empresas de outros segmentos que estão aderindo a utilização da edição colaborativa. Em um artigo de Kaio Sermann (2017) são citadas duas plataformas de segmentos distintos que permitem isso, uma delas é o Spotify, que permite que seus usuários criem *playlists*[2] colaborativamente, outra é um editor de vídeos chamado "DaVinci Resolve", da empresa Black Magic Design, que permite aos usuários edição profissional, correção de cores, efeitos visuais e pós-produção de áudio em 8K.

É possível perceber que o conceito de edição colaborativa pode se aplicar a incontáveis aplicações além das citadas neste trabalho, porém devido ao escopo do trabalho será aprofundado apenas o conceito da escrita colaborativa.

2.2.1 Escrita Colaborativa

Desde que os computadores começaram a ser utilizados para processamento de texto, a escrita colaborativa tem recebido certa atenção, com foco em duas áreas de pesquisa: a pesquisa que estuda a escrita colaborativa em termos de processos de aprendizagem em grupo, concentrada em temas como a construção de conhecimento e resultados de aprendizagem; e a pesquisa que analisa a escrita colaborativa em termos de processos de trabalho em grupo, com foco em questões como produtividade e qualidade dos resultados. Na primeira linha de pesquisa, a escrita colaborativa por meio de CSCL[3] é vista como um meio de aprofundar o envolvimento dos autores com as ideias e a construção do conhecimento (EDE; EDE; LUNSFORD, 1990).

[2] Coleções de músicas.

[3] Aprendizagem colaborativa suportada por computador.

2.2.2 Escrita com foco no processo

Escrita colaborativa é um processo interativo, social, cognitivo e organizacionalmente exigente que envolve um grupo de pessoas focadas em um objetivo comum, trabalhando de forma coordenada e se comunicando durante a criação de um documento. Como uma forma distinta de trabalho em grupo que envolve uma ampla gama de atividades de grupo, várias funções e subtarefas. (LOWRY; CURTIS; LOWRY, 2004).

Martins et al. (2008) a define como a designação atribuída ao ato de uma ou mais aplicações permitirem que várias pessoas editem um mesmo documento em computadores diferentes.

Além disso, quando realizado por grupos que se dialogam através dos meios de comunicação, o processo envolve, tipicamente, várias ferramentas com diferentes características de utilização, como por exemplo, telefone, correio eletrônico, mensagens instantâneas ou sistemas de gestão de documentos. A partir de uma perspectiva cognitiva, a escrita tem sido descrita como um tipo de problema "mal estruturado", o que significa que a tarefa de escrita tem de ser esclarecida pelos autores antes de se envolver na resolução de qualquer problema mais direcionado (LOWRY; CURTIS; LOWRY, 2004).

Quando realizada em um contexto educacional, o professor normalmente fornece a tarefa a ser escrita, as ferramentas de escrita e comunicação e a composição do grupo, para que as equipes possam se concentrar em planejamento de equipe e na produção do documento em si. Essas etapas são complexas, pois envolvem a decomposição de tarefas, definição de funções, alocação das tarefas, planejamento, debate, elaboração, revisão e edição do texto. Esses passos não são requisitos formais de todas as atividades de escrita colaborativa e muitas vezes não são geridos pelo professor, mas diretamente pelos alunos (LOWRY; CURTIS; LOWRY, 2004).

As aplicações que suportam escrita colaborativa dividem-se sem dois grupos: as que permitem a edição colaborativa em tempo real, conhecida como escrita colaborativa síncrona; e as que não permitem essa interação imediata, que são as que utilizam da escrita colaborativa assíncrona (MARTINS et al., 2008).

2.2.3 Escrita colaborativa síncrona

A escrita colaborativa síncrona é aquela que ocorre em tempo real, permitindo que um autor possa visualizar o que outro está escrevendo enquanto ele escreve. Isso torna a escrita em conjunto algo muito mais interativo, melhora o entendimento e reduz os riscos de autores interferirem na escrita uns dos outros (MARTINS et al., 2008). Alguns exemplos de sistemas que suportam colaboração síncrona de escrita são Google Docs e Word Online.

2.2.4 Escrita colaborativa assíncrona

Por outro lado, ao contrário da que ocorre em tempo real, na escrita colaborativa assíncrona não é possível visualizar o que outros autores escrevem até que eles realizem o envio dessas informações. Além disso, ela possui uma menor interatividade social, mas não quer dizer que não ocorra interação. Ferramentas que suportam esse modelo de escrita geralmente dependem de um sistema de controle de versão para gerenciar trechos conflitantes (MARTINS et al., 2008). Um exemplo de sistema que permite edição colaborativa assíncrona é o Wikipedia.

2.2.5 Aprendizagem colaborativa

A aprendizagem colaborativa vem cada vez mais sendo defendida no cenário acadêmico, visto que possui uma grande capacidade de promover estímulos ao pensamento crítico e à capacidade de interação que, por sua vez, geram uma aprendizagem mais ativa. Tal forma de adquirir e prover conhecimento, segundo seus defensores, aumenta o engajamento dos alunos por sua aprendizagem, de forma a terem mais autonomia para assimilar conceitos, trazendo essencialmente ideias sobre o que é ensino, aprendizagem e conhecimento (TORRES; IRALA, 2014).

Dillenbourg (1999) apresenta de uma forma simples que aprendizagem colaborativa é uma circunstância em que duas ou mais pessoas tentam aprender algo em conjunto. De acordo com o autor, esse conceito pode ter múltiplas interpretações, visto que aprender ou tentar aprender algo é um conceito muito amplo, pois pode, por exemplo, significar a participação em atividades como a resolução de um problema, ou ainda o simples acompanhamento de um curso. Além disso, o aprender "em conjunto" também pode sofrer diversas interpretações, como resolver tarefas totalmente em conjunto ou resolver com a divisão dessas tarefas; situações de aprendizagem virtuais ou presenciais; entre outras.

Por outro lado, em uma visão mais abrangente, aprender coletivamente significa que o aprendizado é o resultado de uma interação entre pessoas que trabalham na realização de tarefas em sistemas interdependentes. Esse tipo de aprendizagem se mostra mais eficiente, visto que é colaborativa e social, em vez competitiva e isolada. As constantes trocas de ideias entre as pessoas amplia o entendimento e aprofunda o pensamento (WIERSEMA, 2002).

Em um contexto escolar, aprendizagem colaborativa pode significar um grupo de duas ou mais pessoas com objetivos em comum, ajudando-se mutuamente na construção do conhecimento (TORRES; IRALA, 2014).

2.3 Markdown

Existem inúmeras linguagens de marcação diferentes para descrever documentos. O LaTeX é uma delas e, apesar de ser extremamente poderosa e muito difundida principalmente no meio acadêmico, possui desvantagens. Dentre essas desvantegens estão o longo período que leva o seu aprendizado e a dificuldade em que as ferramentas têm para gerar algo diferente do formato PDF[4], como, por exemplo, o HTML[5]. No entanto, alternativas mais simples ao LaTeX surgiram ao longo dos anos, como o Markdown (ARLOING et al., 2016).

O Markdown é uma linguagem de marcação leve, criada em 2004 por John Gruber com a contribução de Aaron Swartz. Sua sintaxe de formatação foi projetada para ser simples e legível, pois é basicamente formada por pontuação e caracteres (HERRERO, 2013).

Segundo Herrero, o Markdown foi descrito por John Gruber como:

> "Uma ferramenta de conversão de texto em HTML para escritores da web. O Markdown permite escrever usando um formato de texto plano, de fácil leitura e escrita, e depois convertê-lo em HTML estruturalmente válido."

[4]Formato de Documento Portátil.

[5]Linguagem de marcação utilizada na construção de páginas na Web.

Markdown é uma maneira rápida e fácil de fazer anotações, criar conteúdo para um website, escrever um livro, criar apresentações, escrever um e-mail e produzir documentos prontos para impressão (ALAM, 2018).

Além disso, quase todas as aplicações do Markdown suportam a sintaxe básica delineada no documento de projeto original de John Gruber (ver tabela 2.1). Porém, é importante salientar que existem pequenas variações e discrepâncias entre processadores Markdown (ALAM, 2018).

Tabela 2.1: Principais comandos Markdown. Fonte: (GITTER, 2015)

Nome	Comando
Negrito	**texto** ou __texto__
Itálico	*texto* ou _texto_
Riscado	∼∼texto∼∼
Cabeçalhos	# H1 ## H2 ### H3
Lista	- item ou * item
Bloco de citação	> texto
Menção	@alguém
Link	[título](http://)
Imagem	![alt](http:// título)
Código	` código `
Bloco de código	` ` ` linguagem

2.4 LaTeX

O LaTeX é um sistema para escrever documentos. Sua primeira versão amplamente disponível, misteriosamente numerada 2.09, apareceu em 1985. O LaTeX é extremamente popular nas comunidades científica e acadêmica, isso fez com que se tornasse a linguagem franca do mundo científico. Além disso, a linguagem também é amplamente utilizada na indústria (LAMPORT, 1994).

O sistema conta apenas com *design* lógico. Nesse caso, não se trata de um programa WYSIWYG, ou seja, o usuário deve inserir os elementos do texto por meio de comandos lógicos. O resultado final só é visto após processo de compilação. Outrossim, o LaTeX possui abstrações para lidar com bibliografias, citações, formatos de páginas, referência cruzada e tudo mais que não seja relacionado ao conteúdo do documento em si (CORDEIRO; JOAQUIM; CEDRAN, 2013).

2.4.1 Por que LaTeX?

Quando LaTeX foi introduzido em 1985, poucos autores tinham a facilidade para escrever seus próprios documentos. Hoje, a editoração eletrônica é comum. Um software WYSIWYG permite ver exatamente como será o documento enquanto este está sendo produzido. Programas WYSIWYG são muito atraentes, pois facilitam a colocação de texto onde o autor desejar, em qualquer tamanho e estilo (LAMPORT, 1994). Portanto, por que utilizar LaTeX, quando um programa WYSIWYG permite formatar o texto da maneira que o autor deseja?

Os programas WYSIWYG substituem o *design* lógico do LaTeX pelo *design* visual. O *design* visual é bom para documentos curtos e simples, como cartas e memorandos.

Porém não se pode dizer o mesmo para documentos mais complexos, como trabalhos científicos, pois estes documentos exigem conformidade com regras de formatação e diagramação[6], algo que pode ser facilitado com o uso do LaTeX (LAMPORT, 1994). Além do mais, trabalhos científicos costumam ser grandes e contam com uma constante repetição de elementos – que no LaTeX podem ser associados a comandos, facilitando suas inserções no texto (LAMPORT, 1994).

2.5 Expressões Regulares

Expressões regulares são composições de símbolos e caracteres com funções especiais – chamados de metacaracteres – que, agrupados entre si, formam uma sequência, uma expressão. Essa expressão é interpretada como uma regra que indicará sucesso se uma entrada de dados obedecer exatamente a todas as suas condições (JARGAS, 2009). As REGEX[7], como também são conhecidas, são muito utilizadas para buscar trechos em posições específicas dentro de uma cadeia de caracteres, desde que esse trecho corresponda com a regra estabelecida pela expressão (JARGAS, 2009).

2.5.1 Metacaracteres

Os metacaracteres são caracteres especiais com funções específicas, onde cada metacaractere pode estar associado a uma ou mais funções, de acordo com o contexto no qual está inserido. Além disso, eles podem ser combinados uns com os outros, de forma a gerar construções mais complexas (JARGAS, 2009). Na tabela 2.2, é possível ver alguns dos principais metacaracteres e suas respectivas funções.

Tabela 2.2: Exemplos de metacaracteres em expressões regulares. Fonte: (JARGAS, 2009)

Metacaractere	Nome	Função
.	Ponto	Um caractere qualquer
[ ]	Lista	Lista de caracteres permitidos
[^]	Lista negada	Lista de caracteres proibidos
[n-m]	Intervalo	Representa um intervalo de n até m
?	Opcional	Zero ou uma vez
*	Asterisco	Zero, uma ou mais vezes
+	Mais	Um ou mais vezes
{n,m}	Chaves	De n até m vezes
^	Circunflexo	Início de linha
$	Dólar	Fim de linha
\b	Borda	Início ou fim de palavra
\	Escape	Torna literal o caractere a seguir
|	Ou	Ou um ou outro
( )	Grupo	Delimita um grupo

[6] Como por exemplo, a ABNT

[7] Do inglês, *Regular Expressions*

2.6 JSON Web Token

O JSON Web Token (JWT) é um padrão aberto (RFC[8] 7519). Seu objetivo é o de definir um modelo compacto, contendo as informações do usuário, dentro de um cabeçalho HTTP, por meio de um objeto JSON[9], de forma a garantir a segurança na transmissão das informações. O *token* gerado pelo JWT é salvo no cliente e suas informações são verificadas a cada requisição. Um segredo é utilizado para criptografar essas informações, através do algoritmo HMAC (*Hash-based Message Authentication Code*)[10] ou de um par de chaves pública e privada, garantindo assim a sua confiabilidade (MONTANHEIRO; CARVALHO; RODRIGUES, 2017).

2.6.1 Estrutura do token

O *token* gerado é dividido por dois pontos (“.”), totalizando três partes, onde cada uma dessas partes contém um tipo de informação diferente. A primeira parte corresponde ao cabeçalho, onde consta informações a respeito da criptografia utilizada para assinar o *token*. A segunda corresponde ao que é chamado de *payload* do *token*, onde estão informações como o momento em que o *token* foi gerado, número de identificação do usuário a quem o *token* pertence, o nível de acesso desse usuário à aplicação e a validade do *token*. E a terceira e última parte, contém uma assinatura criptografada, com base nas primeiras duas partes e chave secreta (MONTANHEIRO; CARVALHO; RODRIGUES, 2017). Um *token* completo pode ser visto na figura 2.1.

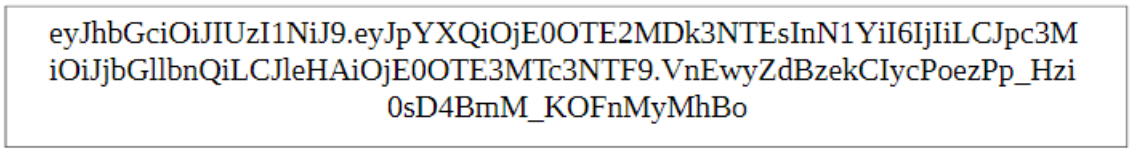

Figura 2.1: Token completo gerado pelo JWT. Fonte: (MONTANHEIRO; CARVALHO; RODRIGUES, 2017)

2.7 Web APIs RESTful

Web APIs RESTful, ou simplesmente APIs REST, é o termo utilizado para representar serviços que utilizam arquitetura REST, onde cada API possui sua própria base de dados não compartilhada. Esses serviços expõem informações em formato texto, normalmente JSON ou XML, e os transmitem via protocolo HTTP, gerando assim uma camada de integração (GALVÃO; SOARES; KAI, 2017).

Essa abordagem é muito utilizada na construção de sistemas distribuídos, visto que reduz o acoplamento das aplicações, de forma que a evolução de uma aplicação não gere grande impacto nas demais, visto que possíveis alterações em um dos sistemas, não incide diretamente na integração entre eles (GALVÃO; SOARES; KAI, 2017).

[8]Documentos técnicos desenvolvidos e mantidos pelo IETF, instituição que especifica os padrões que serão implementados e utilizados em toda a internet.

[9]Acrônimo de JavaScript Object Notation. É um modelo de transmissão de informações no formato texto.

[10]Construção específica para calcular o código de autenticação de mensagem (MAC) envolvendo uma função *hash* criptográfica em combinação com uma chave secreta.

2.8 Visual Studio Code

O Visual Studio Code é um editor de código-fonte leve, mas poderoso, baseado em Electron – que é um *framework* usado para desenvolver aplicativos NodeJS para *desktop*. O sistema foi desenvolvido pela Microsoft para Windows, Linux e macOS, conta com suporte interno para JavaScript, TypeScript e NodeJS e possui um ecossistema de extensões para outras linguagens, como C++, C#, Java, Python, PHP e Go. Além disso, o editor inclui ainda suporte para depuração, controle Git[11] incorporado, realce de sintaxe, complementação inteligente de código, *snippets*[12] e refatoração de código (VSCODE, 2019).

2.8.1 Extensões

As extensões do VS Code permitem adicionar diversos recursos às funcionalidades padrão do sistema, como, por exemplo, adicionar suporte a novas linguagens, depuradores, temas, *snippets* e ferramentas. O sistema possui um *Marketplace*[13] de extensões, onde é possível encontrar variados tipos de extensões e instalá-las com apenas um clique (VSCODE, 2019).

Há ainda a possibilidade dos usuários construírem suas próprias extensões, através dos pacotes NPM[14] (*Node Package Manager*) Yeoman e VS Code Extension Generator. Eles se encarregam de montar um projeto NPM pronto para o desenvolvimento, por meio do comando `yo code`. Algumas configurações iniciais deverão ser preenchidas, como a linguagem utilizada, que poderá ser tanto em TypeScript, quanto JavaScript (figura 2.2).

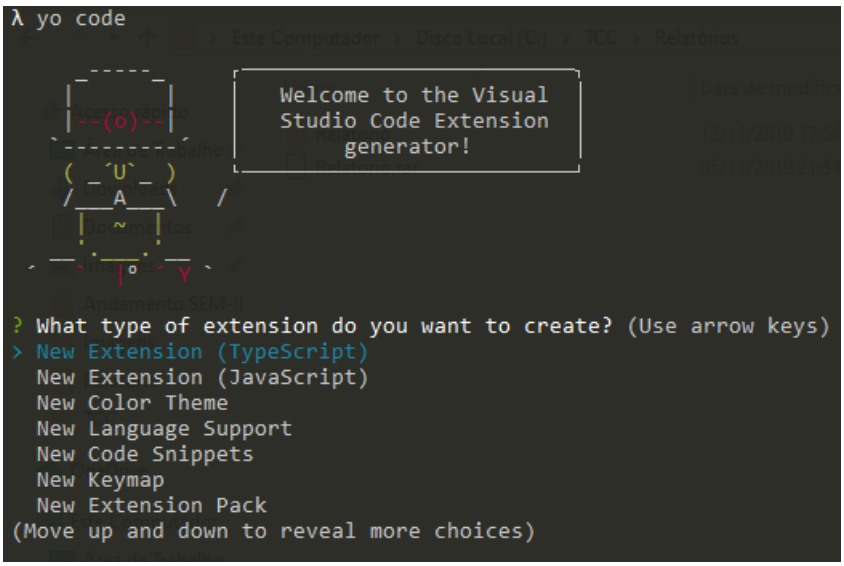

Figura 2.2: VS Code Extension Generator. Fonte: Própria autoria.

2.8.2 Live Share

Live Share é uma extensão da própria Microsoft para o Visual Studio Code que permite editar e depurar de forma colaborativa e em tempo real. Ela permite que os usuários compartilhem sessões de depuração, instancias de terminal e até mesmo chamdas de voz (LIVESHARE, 2019).

[11] Sistema de controle de versões, muito utilizado no desenvolvimento de *softwares*.
[12] Blocos de código reutilizáveis
[13] *Shopping center* virtual.
[14] Gerenciador de pacotes do NodeJS.

2.8.2.1 Login

Ao instalar a extensão, o primeiro passo é realizar o *login* na aplicação, através de um botão presente no sistema. O navegador padrão é inicializado para que o usuário possa inserir as suas credenciais de acesso utilizando uma conta Microsoft. Ao concluir o processo, o Visual Studio Code carregará, automaticamente, uma sessão de colaboração do espaço de trabalho atual (LIVESHARE, 2019).

2.8.2.2 Iniciar uma sessão de colaboração

Para usuários já autenticados, a extensão possui um botão para criar uma nova sessão de colaboração, que pode ser tanto uma sessão de colaboração comum, onde todos os usuários possuem permissão de edição e escrita, quanto uma sessão de colaboração somente leitura, que por sua vez, só permite que os colaborares visualizem o que está sendo editado (LIVESHARE, 2019).

2.8.2.3 Encaminhar link de convite

Outro recurso presente na extensão é a função de encaminhar um *link* de convite a outros usuários. Na URL desse *link* existe um *token* de acesso, assinado com o identificador único do espaço de trabalho a ser compartilhado. Dessa forma é possível que outros usuários possam ingressar na sessão de colaboração criada (LIVESHARE, 2019).

2.8.2.4 Aprovar hóspedes

Por padrão uma sessão de colaboração aceita qualquer usuário que ingressar por meio de *link* de convite, todavia, é possível exigir aprovação explícita de cada usuário que ingressar. Para isso é necessário ativar o recurso nas configurações do sistema (LIVESHARE, 2019).

3 SISTEMAS EXISTENTES

Neste capítulo serão analisados alguns softwares existentes no mercado que possuem algum tipo de relação com o sistema a ser desenvolvido. Será realizada uma avaliação de suas funcionalidades, com base em suas documentações oficiais e experiência própria de uso do autor, a fim de compreender, na visão do autor, suas características, suas vantagens e diagnosticar seus problemas.

3.1 Overleaf

O Overleaf é um editor LaTeX online, gratuito, focado na criação colaborativa, em tempo real, de trabalhos, teses, relatórios técnicos e outros documentos para cientistas. Há 6 anos no mercado, hoje o Overleaf oferece suporte há mais de três milhões de autores em mais de 180 países, com mais de trinta e cinco milhões de documentos criados (OVERLEAF, 2019).

3.1.1 Overleaf: Funcionalidades

Por se tratar de um sistema web, o Overleaf não requer instalação, portanto o único requisito prévio é que o usuário tenha acesso a um navegador web. Edição offline também é possível, pois o sistema oferece integração com GitHub e Dropbox, no entanto, essa função só está disponível para as versões *Collaborator* e *Professional*, ambas pagas (OVERLEAF, 2019).

Além disso, a aplicação conta com um sistema de compilação automática em segundo plano, que pode ser desativado a qualquer momento pelo usuário. Quando desativado, basta o usuário clicar no botão *"Recompile"* para que o código seja compilado, gerando o PDF. O sistema ainda possui visualização do resultado em tempo real, que é mostrada à direita da tela, podendo ser redimensionada de acordo com o gosto do usuário, como pode ser visto na figura 3.1.

Cada documento criado no Overleaf é privado. Existem duas formas de compartilhar o documento com os outros autores: por convite privado e por compartilhamento de link (OVERLEAF, 2019).

- **Convite privado:** O Overleaf permite que usuários enviem convites, por e-mail, a colaborares selecionados e nomeados, para acessarem seus projetos. Na versão gratuita, o número de convites permitidos é limitado a apenas um colaborador, enquanto na versão intermediária esse limite passa para dez. Já para a versão *Profesional* não existe limite no número de convites;

- **Compartilhamento de link:** O sistema permite compartilhar projetos por meio de

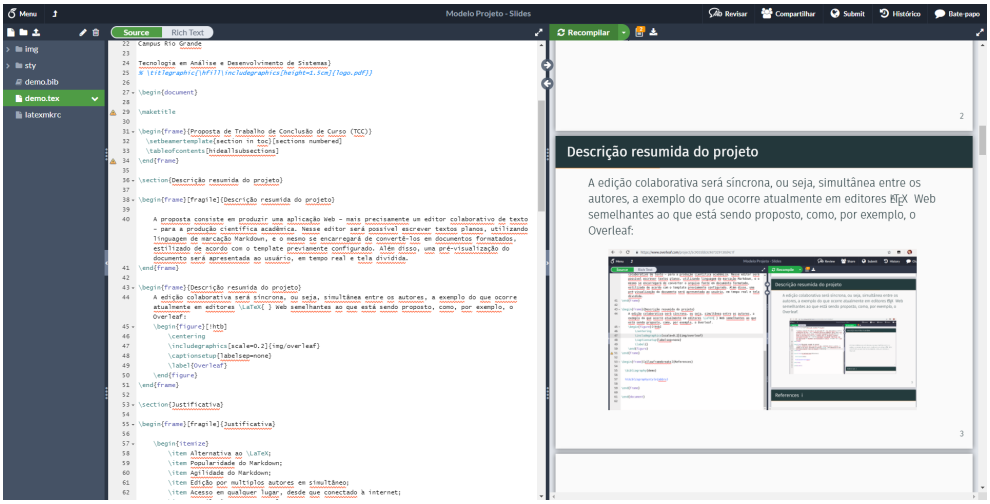

Figura 3.1: Tela de edição de um projeto no Overleaf. Fonte: Própria autoria.

links secretos. Basta ativar o compartilhamento de link e enviar a URL para os co-autores. Ao acessá-la, eles poderão analisar, comentar e editar o documento. Ao desativar o compartilhamento de link é possível tornar o projeto privado novamente.

A ferramenta conta também com comentários em tempo real e bate-papo integrado. Os usuários podem discutir seus trabalhos sem a necessidade de utilizar um outro meio de comunicação. Nos planos pagos, os usuários podem acompanhar todas as alterações feitas em seus projetos e quem as fez, bem como aceitar ou rejeitar alterações individuais (OVERLEAF, 2019).

Usuários sem conhecimento da linguagem de marcação LaTeX também podem editar documentos no sistema. Isso se deve ao fato da aplicação contar com um modo de edição *Rich Text*, o que permite que usuários insiram elementos e estilos no documento através de botões (ver figura 3.2).

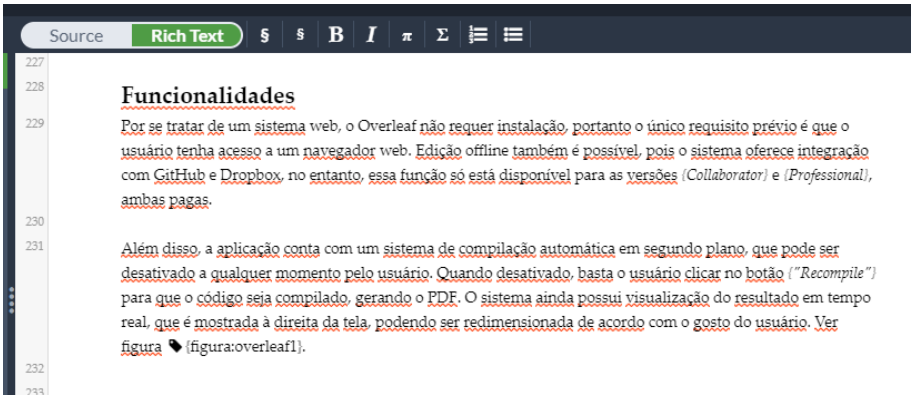

Figura 3.2: Modo de edição *Rich Text* no Overleaf. Fonte: Própria autoria.

Além disso, o Overleaf fornece recursos para que os usuários encontrem erros de sintaxe no documento em tempo real, pois sempre que um código LaTeX é escrito fora de sintaxe, o erro é realçado em vermelho, uma advertência é exibida acima do número da linha, e um *log*[1] de erro de compilação é armazenado, contendo os detalhes do erro

[1]Expressão utilizada para descrever o processo de registro de eventos relevantes num sistema computacional.

(OVERLEAF, 2019).

3.1.2 Overleaf: vantagens e desvantagens

O Overleaf é uma ferramenta colaborativa de escrita e publicação online que torna todo o processo de escrever, editar e publicar documentos científicos muito mais rápido e fácil. A ferramenta fornece a conveniência de um editor LaTeX simples, com colaboração em tempo real e a saída totalmente compilada produzida automaticamente em segundo plano à medida que o usuário digita.

No entanto, as limitações da versão básica podem ser um entrave. A falta de integração com repositórios externos nessa versão dificulta o trabalho offline, ademais o fato de nessa versão permitir apenas o envio de um convite para outro colaborador, pode ser um incômodo em alguns casos.

3.2 Google Docs

O Google Docs é um pacote de aplicações colaborativas da Google. Baseadas em AJAX, essas aplicações funcionam tanto de forma síncrona, quanto de forma assíncrona, permitindo que usuários criem e editem documentos, apresentações, planilhas e formulários, enquanto colaboram em tempo real com outros usuários.

Originalmente batizada de Writely, a ferramenta foi desenvolvida em meados de 2005 pela empresa Writely Team e, meses após, adquirida pela Google, assim a renomeando para como é conhecida até os dias de hoje.

3.2.1 Google Docs: funcionalidades

O Google Docs conta com ferramentas de edição e estilo, o que facilita a formatação e a inserção de elementos no documento (GOOGLE, 2019). Por meio desses recursos, é possível que seus usuários formatem documentos livremente, com alteração de fontes, inserção de *links*, adição de imagens, desenhos, listas, títulos, cor de texto, alinhamentos, entre outros (GOOGLE, 2019). Tudo de forma colaborativa síncrona, sem limite no número de autores simultâneos.

A ferramenta ainda conta com aplicativos *offline* para *tablets*, *smartphones* e computadores, que, apesar de funcionarem fora da rede, quando conectadas as aplicações sincronizam automaticamente as alterações recebidas enquanto não havia conexão, armazenando-as em nuvem (Google Drive) para que assim possam ser acessadas pelos demais autores (GOOGLE, 2019).

Assim como em outros sistemas, o Google Docs gera um link permanente para cada documento criado, que pode ser compartilhado com os demais autores, contudo não basta apenas estar em posse do link para acessar um documento (GOOGLE, 2019). O Google Docs exige que o criador do documento dê permissão aos outros autores, através de suas contas Google (ver figura 3.3). No entanto, também é possível tornar um documento público, o que o deixa acessível por qualquer usuário, seja este registrado ou não (GOOGLE, 2019).

Além disso, o sistema conta com diferenciais como ferramentas de acessibilidade para deficientes físicos e visuais, digitação por reconhecimento de voz, correção ortográfica, salvamento automático e ferramenta de tradução, esta permite que usuários traduzam documentos, ou trechos de texto, para quaisquer idiomas presentes no Google Tradutor (GOOGLE, 2019). O Google Docs possui também compatibilidade com o Microsoft Word, permitindo que usuários abram, editem e salvem arquivos nesse formato, por meio

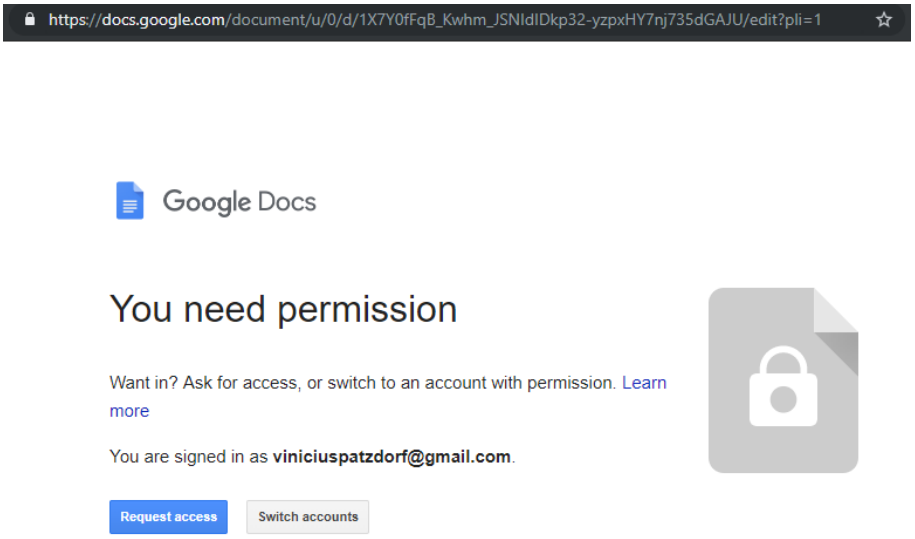

Figura 3.3: Exemplo de tentativa de acesso por usuário sem permissão. Fonte: Própria autoria.

de uma extensão do Google Chrome. A ferramenta ainda permite que usuários convertam arquivos do Word para Google Docs e vice-versa, de modo que os usuários não se preocupem com o formato dos arquivos.

Outro recurso disponível é a instalação de complementos no documento. Funcionalidades não encontrados nativamente no sistema, podem ser inseridas por meio desses complementos, que nada mais são do que softwares desenvolvidos por terceiros, aprovados pela Google. Alguns exemplos de complementos uteis para a produção de documentos técnicos são:

- **EasyBib:** Gerador automático de bibliografia e citações;
- **Lucidchart e Draw.io:** Geradores de gráficos e diagramas;
- **Table of Contents:** Gerador de tabelas;
- **Thesaurus:** Ferramenta de auxílio para fornecimento de sinônimos e antônimos de palavras;
- **MathType:** Complemento para inserção de símbolos, fórmulas e notações matemáticas.

3.2.2 Google Docs: vantagens e desvantagens

O Google Docs possui recursos para que seus usuários possam criar, manipular e gerenciar documentos de forma colaborativa síncrona e assíncrona, multiplataforma, sem limite de autores, totalmente gratuito, contando com funções de acessibilidade e compatibilidade com os principais formatos de documentos. Dessa forma, o Google Docs pode ser considerado uma das principais plataformas colaborativas na atualidade.

Entretanto, quando comparado a outros sistemas utilizados na produção de documentos científicos, o Google Docs ainda fica aquém no que diz respeito a agilidade, pois a ferramenta não abstrai o usuário de cuidados com formatação exigidos por normas técnicas como a ABNT.

3.3 Dropbox Paper

Lançado definitivamente em janeiro de 2017, o Dropbox Paper apresenta uma interface básica e usabilidade bastante intuitiva. A aplicação, além de ser uma plataforma na nuvem para criação de documentos de texto colaborativos, pode ser usada como um gerenciador de trabalho com cronogramas, notas e listas de atividades (DROPBOX, 2019), semelhante ao Trello[2]. Tudo isso com sincronização em tempo real, através do Dropbox, para todos os membros da equipe, estando eles conectados à versão web ou *mobile* (Android e iOS).

3.3.1 Dropbox Paper: funcionalidades

O Paper, assim como os seus concorrentes, conta com edição colaborativa síncrona, no entanto, seus recursos formatativos são um pouco mais enxutos em relação a outros editores colaborativos. Isso se deve ao fato de que o Paper foi projetado para ser um recurso utilizado em empresas, não sendo tão útil quando há a necessidade de se produzir documentos técnicos e científicos.

Além da escrita colaborativa em tempo real, a ferramenta conta com compartilhamento de documentos via convite, ou via menção – um recurso diferenciado. Quando um usuário menciona outro no documento, automaticamente o documento passa a estar compartilhada com esse outro usuário, permitindo-o a visualizar e editar o arquivo (GOOGLE, 2019), como pode ser visto na figura 3.4.

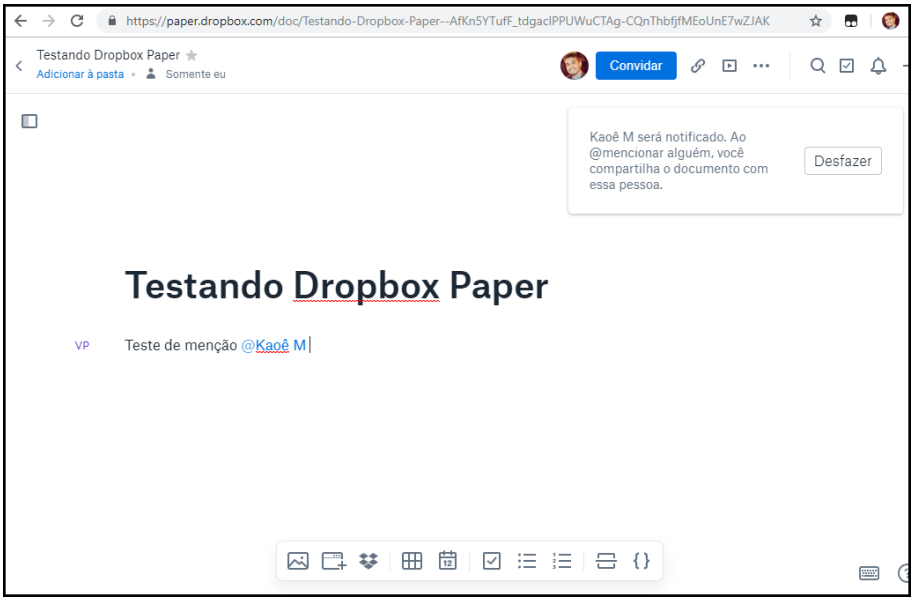

Figura 3.4: Exemplo de compartilhamento de documento via menção no Dropbox Paper. Fonte: Própria autoria.

Outras funcionalidades encontradas no Dropbox paper são: histórico de modificações, o que permite tanto visualizar as últimas alterações no documento e quem as fez, como também restaurar versões anteriores deste; criação de tarefas, recurso de grande utilidade em atividades de planejamento de projetos; adição de prazos; inserção de *feedbacks*[3] visíveis para todos os colaboradores, entre outros (DROPBOX, 2019).

[2]Ferramenta de gerenciamento de projetos
[3]Comentários em palavras ou trechos do texto

3.3.1.1 Markdown e LaTeX

O Dropbox Paper aceita comandos tanto em Markdown, quanto em LaTeX, além de contar com uma infinita lista de atalhos para comandos rápidos de formatação, como inserção de negritos, itálicos, sublinhados, riscados, listas, cabeçalhos, blocos de citação, dentre tantos outros. Os comandos em Markdown são inseridos diretamente no documento e automaticamente reconhecidos e interpretados. Já os comandos LaTeX precisam ser inseridos entre um duplo dólar, para que possam ser interpretados corretamente (ver figura 3.5).

Figura 3.5: Lista de comandos Markdown e LaTeX no Dropbox Paper. Fonte: Própria autoria.

3.3.2 Dropbox Paper: Vantagens e desvantagens

O Dropbox Paper é uma plataforma totalmente em português, com integração instantânea entre a ferramenta web e aplicativo. Ademais, o usuário que possui conta grátis no Dropbox tem direito as mesmas funções do usuário de conta *premium* e os arquivos gerados pelo Paper não ocupam espaço na conta geral do Dropbox. Outro ponto positivo é a capacidade de aceitar comandos tanto em LaTeX quanto em Markdown, fazendo com que usuários ganhem tempo com questões de formatação. Contudo, a aplicação não permite alteração de fonte, exige que todos os colaboradores sejam cadastrados junto ao Dropbox, além de só funcionar no aplicativo após cadastro realizado na versão web.

3.4 Microsoft Word Online

Dentre os programas mais utilizados para escrita científica está o consolidado Word, do pacote office da Microsoft. Lançado oficialmente em 1983 para IBM PC, rodando em MS-DOS[4], o Microsoft Word era descrito como um processador de texto WYSIWYG[5], com a capacidade de adicionar e remover estilos como negrito, itálico e sublinhado, embora não pudesse renderizar fontes. Além disso, o software foi projetado para ser usado com um mouse, o que foi apontado como inovador para a época (BRITANNICA, 2019).

Ao longo dos anos foram sendo lançadas versões para diversos sistemas, como Mac OS, Atari TOS, SCO Unix, Windows e, finalmente, em 2009, recebendo uma versão web-colaborativa – à época batizada de Word Web App – hoje conhecida como Word Online. Ademais, também foram lançadas versões para dispositivos *mobile*, como iOS e Android (OFFICE, 2019).

3.4.1 Microsoft Word Online: funcionalidades

A versão *deskotp* atual do Word, que faz parte do pacote Office 2019, é um processador de texto completo, contendo ferramentas para todo o tipo de formatação e estilização. Dentre os principais recursos disponíveis, conforme suporte oficial (2019), estão:

- Inserção simplificada de gráficos, planilhas e desenhos;
- Variedades de tipos e tamanhos de fontes, incluindo símbolos gráficos;
- Criação de estilos e modelos de documentos com formatações predefinidas;
- Destaques de texto como bordas, sombreamento e destaque de caracteres;
- Revisor ortográfico incorporado;
- Recursos como cabeçalhos, rodapés, texto multi-colunado, gerador de índices analíticos e remissivos, editor de macros, ferramentas para produção de desenhos e logomarcas e editor de fórmulas matemáticas e científicas;
- Autoformatação de textos e documentos;
- Mala-Direta simplificada, com opção para criação de etiquetas, cartas modelos, envelopes e catálogos.

Por outro lado, a versão web atual do Word, vinculada a uma conta Microsoft, possibilita fazer apenas edições e alterações de formatação básicas em documentos, pois inúmeras funções da versão *desktop* não fazem parte da versão web. Recursos como as réguas, as linhas de grade, abrir arquivos protegidos por senha, localizar e substituir, alterar layout de página, o uso de dicionários personalizados, alterar tamanho de células e estilo de tabelas, editar símbolos e equações, editar página de rosto, executar macros[6], gerar sumário e inserir referência não estão disponíveis. Entretanto, o Word online conta com um botão "Abrir no Word", que tem a função de abrir o documento na área de trabalho, ou seja, na versão *desktop* (OFFICE, 2019).

[4] Sistema operacional em disco da Microsoft

[5] What You See Is What You Get - O que você vê é o que você obtém: o usuário tem a imagem real de impressão do documento

[6] Série de comandos e instruções que são agrupados como um único comando para realizar uma tarefa automaticamente.

3.4.1.1 OneDrive

O OneDrive é o serviço de nuvem do Microsoft. Ele está disponível gratuitamente, com capacidade limitada a cinco *giga bytes*, para usuários que possuem uma conta. No entanto, somente ao adquirir uma licença do Office 365 o usuário terá acesso às versões para a área de trabalho dos *softwares* de escritório da Microsft, além de aumentar a capacidade de armazenamento do OneDrive para um *tera byte* ou mais, de acordo com o plano contratado. Ao contratar um desses planos e possuir instalado o Word *desktop* em seu computador, o usuário poderá ter cópias locais de documentos hospedados no OneDrive, e estes serão sincronizados automaticamente, de forma que possa ser editado por múltiplos autores, estejam eles utilizando o Word ou o Word Online (OFFICE, 2019).

3.4.2 Microsoft Word Online: vantagens e desvantagens

Não é possível utilizar somente o Word Online para confecção de documentos científicos, pois, como visto anteriormente, ele possui recursos um tanto limitados. Todavia, é possível usá-lo para tal função em conjunto com sua versão *desktop*, mantendo a versão web apenas para simples edições, ou para o acompanhamento por parte de um autor mais focado em conteúdo escrito, enquanto outros cuidam das funções avançadas através da versão para a área de trabalho.

3.5 HackMD/CodiMD

O CodiMD é um projeto de código aberto que permite que seus usuários criem e gerenciem notas colaborativas em tempo real, utilizando Markdown. Originalmente batizado de HackMD, o projeto possui duas versões: O HackMD EE, renomeada posteriormente para somente HackMD, que é um produto SaaS[7], portanto de código fechado. E o HackMD CE, versão de código aberto renomeada para CodiMD. Ambas as versões são softwares web.

O HackMD possui três planos de serviço, sendo o primeiro gratuito e com funcionalidades limitadas em relação aos outros dois (HACKMD, 2019). No plano gratuito não existe sistema de autenticação, portanto basta um dos autores criar uma nova anotação e a aplicação se encarrega de gerar uma URL[8] contendo um identificador único. O autor compartilha essa URL com outros autores e, dessa forma, todos ingressam no sistema de edição da anotação, podendo realizar qualquer tipo de edição na mesma, como pode ser visto na figura 3.6.

3.5.1 HackMD/CodiMD: funcionalidades

A ferramenta permite três modos de visualização diferentes, onde o usuário poderá optar por cada um deles clicando nos botões presentes no canto esquerdo superior, ao lado do logotipo, como pode ser visto na figura 3.6. Os três modos de visualização são:

- **Modo editar:** Modo em que os autores escrevem em texto plano utilizando linguagem de marcação Markdown;
- **Modo ver:** Modo em que é possível visualizar o resultado final da interpretação do código-fonte;

[7] Forma de comercialização de software em que o cliente paga uma assinatura e recebe em troca toda a estrutura necessária em forma de serviço.

[8] Endereço de um recurso disponível em uma rede

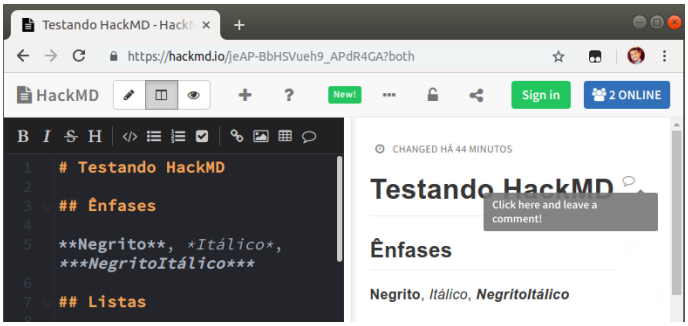

Figura 3.6: Nota sendo editada por dois autores no HackMD. Fonte: Própria autoria.

- **Modo ambos:** Modo em que a tela fica dividida, mostrando em simultâneo ambas as visualizações anteriores.

O sistema conta também com um botão de publicação (canto superior direito). Ao clicar nesse botão a aplicação gera uma nova URL da nota, contendo apenas esta em sua versão formatada, pronta para ser impressa (Ver figura 3.7).

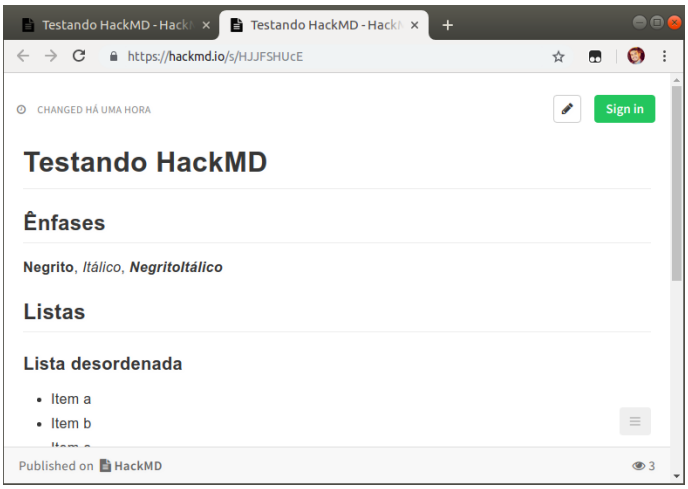

Figura 3.7: Nota publicada no HackMD. Fonte: Própria autoria.

Ao lado do botão de publicação, existe também um outro botão contendo diversas opções, separadas pelas seguintes seções: *options*, *template*, *export*, *import* e *download* (Ver figura 3.8).

- **Seção *options*:** Contém apenas a opção de verificar as revisões da nota (versionamento). Nesta opção é possível visualizar o histórico de alterações da nota, bem como reverter para versões anteriores;

- **Seção *template*:** Contém a opção de guardar a nota atual como um modelo para ser utilizado em futuras notas (exige uma conta registrada) e a opção de inserir um modelo. Esta última, quando selecionada, exibe quatro opções de seleção de modelos próprios da ferramenta, além dos modelos armazenados pelo usuário;

- **Seção *export*:** Contém opções de armazenamento em três repositórios externos (Dropbox, Google Drive e GitHub);

- **Seção *import*:** Contém opções de importação a partir de repositórios externos (os mesmos da seção *export*), além da opção de importar a partir da área de transferência do Sistema Operacional;

- **Seção *download*:** Contém opções de *download* da nota nos formatos Markdown, HTML (contendo o mesmo *layout* exibido na ferramenta), HTML puro e ODT[9].

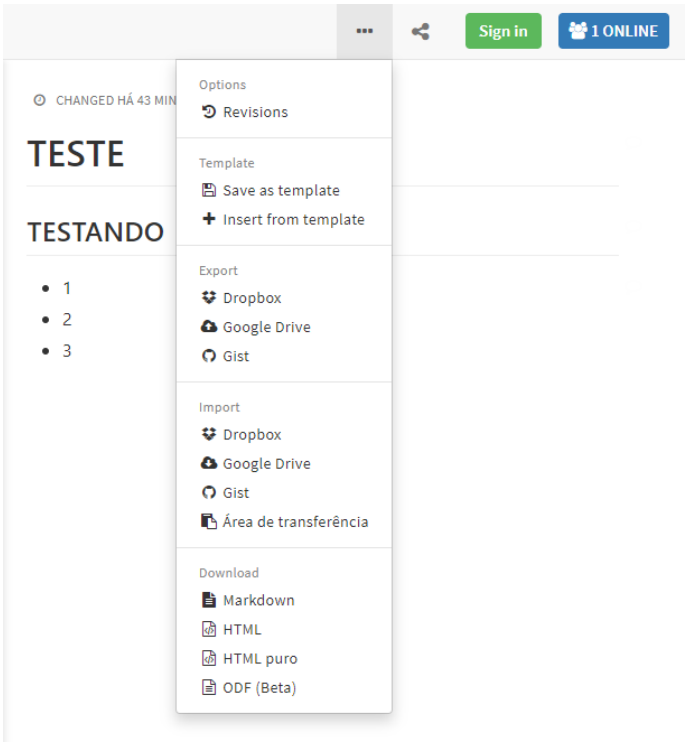

Figura 3.8: Botão de opções no HackMD. Fonte: Própria autoria.

3.5.2 HackMD/CodiMD: vantagens e desvantagens

Esta ferramenta possui inúmeras vantagens em comparação a outros editores Markdown. Uma delas é o fato de ela ser web, o que a torna bastante abrangente e de fácil acesso, visto que para se usar o editor basta ter em mãos um dispositivo contendo navegador web e acesso a internet. Além disso, outro ponto importante é o fato de ela possuir edição colaborativa síncrona, assim, um autor vê em tempo real o que outro está produzindo. As integrações com repositórios externos conhecidos também garantem ao HackMD uma vantagem interessante, já que muitos editores não oferecem tal facilidade. Por fim, o histórico de versionamento da ferramenta se mostra uma poderosa vantagem, pois permite que o usuário restaure versões anteriores de suas notas em poucos cliques.

Por outro lado, apesar de atender muito bem o propósito ao qual foi desenvolvido, ou seja, a criação de anotações de forma simples e objetiva, o HackMD deixa a desejar quando o autor necessita de uma ferramenta com uma maior gama de recursos, como para o caso da produção de documentos técnicos, em que se tem toda uma formatação específica a ser seguida. Outro ponto negativo fica a cargo personalização do layout, que não é configurável.

[9]Documento de Formato Aberto para Aplicações Empresariais

3.6 Comparativo entre os sistemas

Tabela 3.1: Comparação dos recursos presentes em cada sistema existente estudado. Fonte: Própria autoria.

Sistemas	Edição colaborativa*	Interpreta Markdown	Interpreta LaTeX	Produção acadêmica	Plataforma**	Máximo colab.
Overleaf	S/A	Não	Sim	Sim	W	2
Google Docs	S/A	Não	Não	Sim	W/M	Indef.
D. Paper	S	Sim	Sim	Não	W/M	Indef.
MS Word	S/A	Não	Não	Sim	W/D/M	Indef.
HackMD	S	Sim	Não	Não	W	2

** S: Síncrona; A: Assíncrona.*

*** W: Web; D: Desktop; M: Mobile.*

4 SISTEMA PROPOSTO

Este capítulo tem por objetivo detalhar o processo de modelagem do sistema, explorando o fluxograma de funcionamento e sua arquitetura geral bem como o diagrama de casos de uso, diagrama de classes, diagrama de entidades e relacionamentos. Além disso, ao final deste capítulo, são abordadas as tecnologias utilizadas no sistema.

4.1 Arquitetura Geral

O sistema proposto consistirá em duas aplicações: Uma Web API RESTful de transpilação, que se encarregará de receber arquivos Markdown para transpilá-los para arquivos TEX; E uma extensão para o editor de textos Visual Studio Code, da Microsoft, que terá a função de consumir a API de transpilação para gerar código LATEX, compilar o código LATEX gerado e proporcionar um ambiente de edição colaborativa, através de uma outra extensão, chamada *Live Share*. A figura 4.1 demonstra o que seria a arquitetura do sistema idealizado.

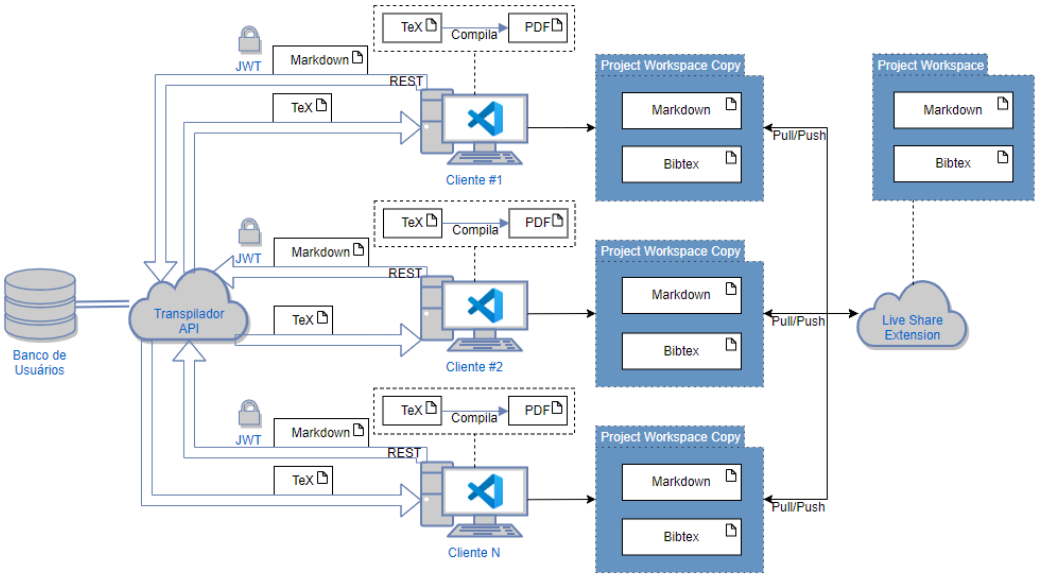

Figura 4.1: Arquitetura geral do sistema. Fonte: Própria autoria.

No desenho, *n* clientes estão editando um mesmo espaço de trabalho colaborativamente, através da extensão *Live Share* – que permite tal recurso. É importante salientar que nessa representação todos os clientes estão utilizando a mesma instância da API de

transpição, no entanto, nada impediria que cada um estivesse executando sua própria instância.

O espaço de trabalho contará, inicialmente, apenas com um arquivo Markdown e um arquivo BibTEX[1], mas após o processo de transpilação e compilação do código LATEX, novos arquivos serão criados no espaço de trabalho, contudo, estes não deverão ser editados. Além disso, a ideia de ter uma banco de usuários conectado ao transpilador é para a segurança da API, que contará com autenticação JWT em seus serviços.

A extensão será batizada de Markdemic, que é a união das palavras Markdown e *Academic*, fazendo alusão a academia – onde são produzidos os documentos científicos – e terá a capacidade de exibir em tela divida o PDF final gerado após o processo de compilação do código LATEX, por meio da criação de um *webview*.

4.2 Fluxograma de Funcionamento

O fluxograma de funcionamento apresenta a sucessão de de acontecimentos do sistema. Nele são representados os procedimentos, cujas etapas são ilustradas na figura 4.2.

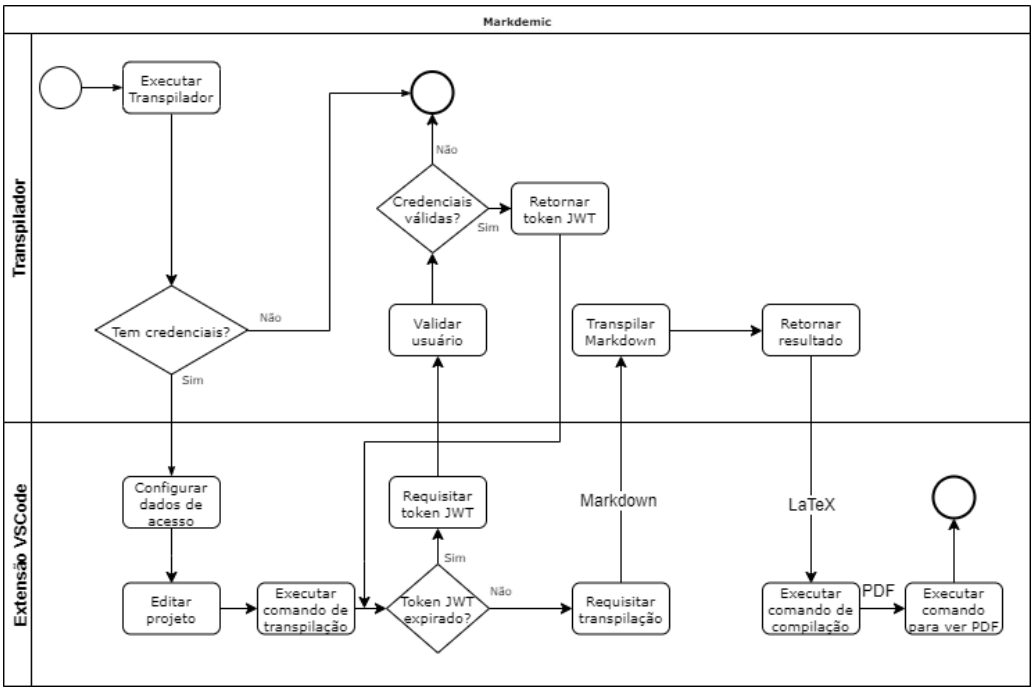

Figura 4.2: Fluxograma de funcionamento do Markdemic. Fonte: Própria autoria.

De acordo com o fluxo mostrado na figura 4.2, primeiramente, o usuário deverá garantir que o transpilador esteja em execução e acessível e se possui as credenciais para acessá-lo, caso não as tenha, o usuário não estará autorizado a utilizar o sistema.

Por outro lado, tendo as credenciais, o usuário irá configurá-las na extensão a ser desenvolvida para o Visual Studio Code e poderá começar a editar seu projeto e a executar o comando de transpilação de código. Quando esse comando for executado, a extensão verificará se existe um *token* JWT em memória e se este não está expirado. Caso não o tenha, ou esteja expirado, uma solicitação de um novo *token* de acesso será realizada para o transpilador. Após receber essa solicitação, o transpilador realizará a validação das

[1] Além dos arquivos Markdown e BibTEX, o espaço de trabalho pode conter imagens e pastas – não estão representadas no desenho por questões de espaço.

credenciais enviadas e, se estas forem válidas, criará um novo *token*, enviando-o como resposta à extensão. Do contrário, o usuário não terá autorização para utilizar o sistema.

Ao executar o comando de transpilação na extensão estando de posse de um *token* válido em memória, a aplicação irá requisitar à API a transpilação do código, enviando o arquivo Markdown no corpo da requisição. Após recebê-lo, o transpilador irá converter o código Markdown em código LaTeX e retornará um novo arquivo como resposta, contendo o código transpilado. Ao receber o retorno, a extensão compilará o código LaTeX, de modo a produzir o PDF final, já formatado e, por fim, o usuário estará apto a executar o comando para visualizá-lo dentro do editor.

4.3 Diagrama de Casos de Uso

O diagrama de casos de uso descreve, do ponto de vista do usuário, o cenário das funcionalidades do sistema. Nele são descritos os atores, as funcionalidades e os seus relacionamentos (ver figura 4.3).

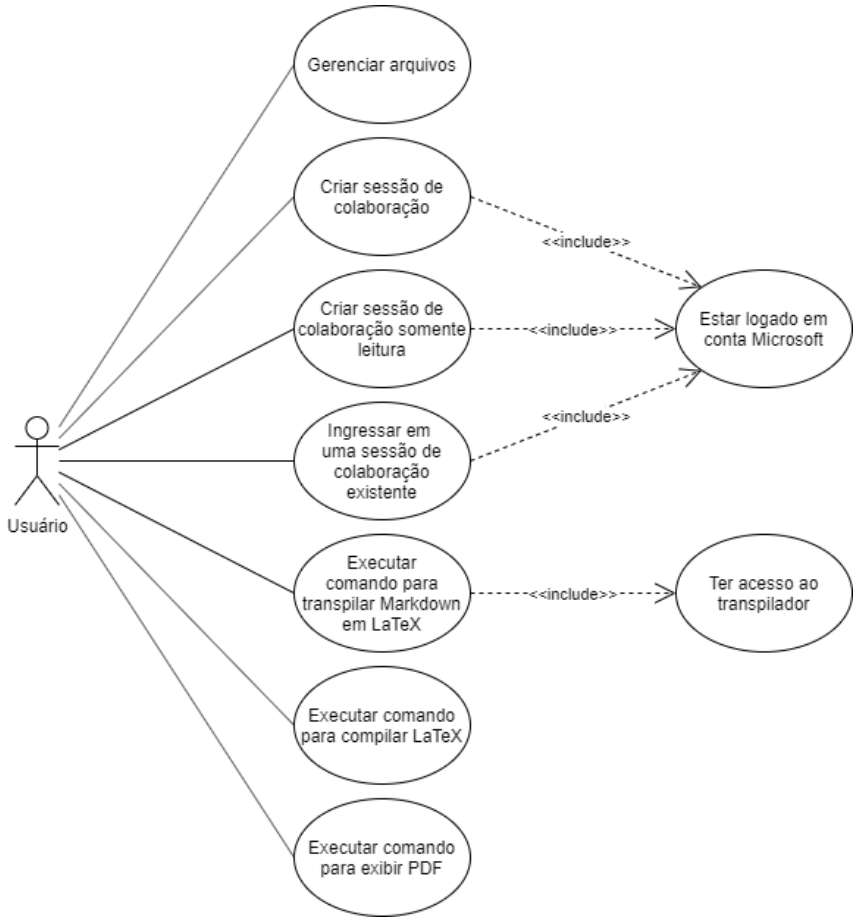

Figura 4.3: Diagrama de casos de uso. Fonte: Própria autoria

- **Usuário:** Ator que representa o usuário do sistema;
- **Gerenciar arquivos:** Caso de uso que representa todo o tipo de gerenciamento dos arquivos que fazem parte do espaço de trabalho;
- **Estar logado em conta Microsoft:** Caso de uso que representa a autenticação de usuários na extensão *Live Share*. Requer uma conta Microsoft;

- **Criar sessão de colaboração:** Caso de uso que representa a operação de criação de uma sessão de colaboração. Requer estar autenticado em uma conta Microsoft. Este tipo de sessão torna os arquivos editáveis por qualquer membro;

- **Criar sessão de colaboração somente leitura:** Caso de uso que representa a operação de criação de uma sessão de colaboração somente leitura. Requer estar autenticado em uma conta Microsoft. Neste tipo de sessão somente o seu criador possui permissão para editar arquivos;

- **Ingressar em uma sessão de colaboração existente:** Ação que representa o ingresso do usuário em uma sessão de colaboração criada por outro usuário. Requer estar autenticado em uma conta Microsoft.

- **Ter acesso ao transpilador:** Caso de uso que representa tanto a autenticação no transpilador, quanto a pré-configuração das credenciais de acesso na extensão Markdemic;

- **Executar comando para transpilar Markdown em LaTeX:** Caso de uso que representa a execução do comando para transpilar código Markdown . Requer acesso ao transpilador;

- **Executar comando para compilar LaTeX:** Ação que representa a execução do comando para compilar código LaTeX gerado pelo transpilador;

- **Executar comando para exibir PDF:** Ação que representa a execução do comando para exibir o PDF gerado pela compilação do código LaTeX.

4.4 Diagrama de Classes

Nesta seção serão apresentados os diagramas de classes que correspondem tanto ao transpilador, quanto à extensão Markdemic, para o Visual Studio Code. O diagrama de classes mostrado na figura 4.4, corresponde à estrutura estática das classes do sistema de controle de usuário, autenticação JWT e a classe principal (*main*) do transpilador.

- **Classe TranspilerApplication:** Classe principal da aplicação *Spring Boot*. Responsável por executar a API;

- **Classe UserController:** Classe responsável por realizar o controle e mapeamento de rotas dos serviços de usuário. Possui um único atributo do tipo `UserRepository`. Seus métodos são:

  - `getAll()`: Método para listar todos os usuários;
  - `getById()`: Método para buscar usuário pelo seu identificador único;
  - `create()`: Método para criar um novo usuário;
  - `update()`: Método para buscar e atualizar um usuário existente com base no seu identificador único;
  - `remove()`: Método para buscar e remover um usuário existente com base no seu identificador único;

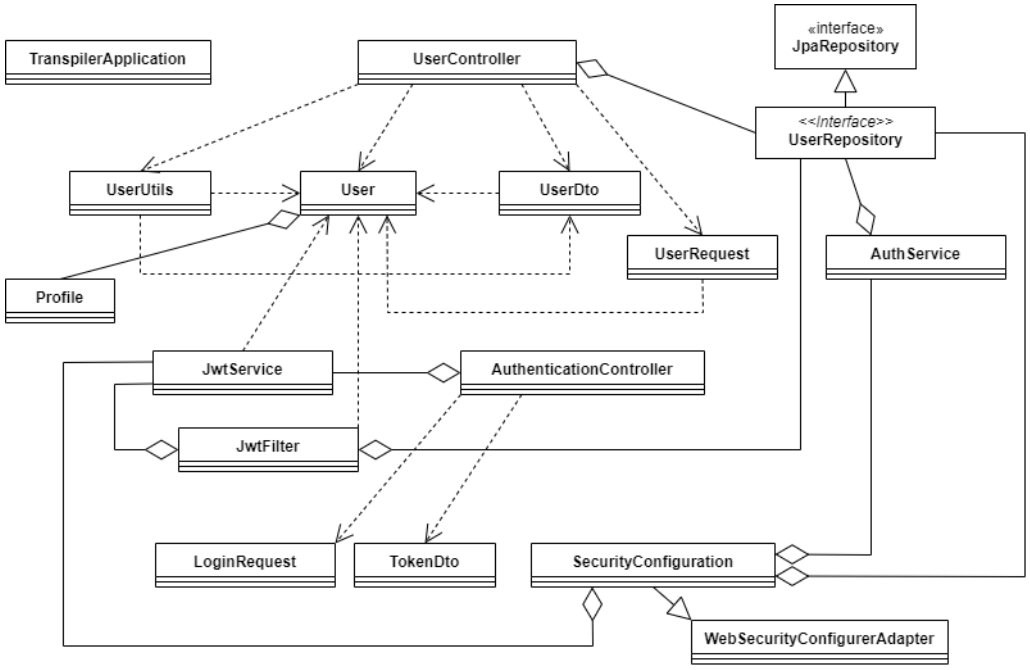

Figura 4.4: Diagrama de classes do controle de usuário, autenticação e classe principal do transpilador. Fonte: Própria autoria

- **Interface UserRepository:** Interface que herda da `JpaRepository`, do pacote Spring, os métodos para persistência de dados da entidade JPA[2] (Java Persistence API) `User`, já mapeada. O seu único método não herdado é o `findByEmail()`, que é utilizado para buscar usuários pelo campo e-mail;
- **Classe User:** Classe modelo responsável por instanciar entidades JPA de usuário. Possui os atributos `id`, `name`, `email`, `password` e `profiles`. Seus métodos são apenas seus dois contrutores e seus *getters* e *setters*;
- **Classe UserDto:** Classe modelo de transferência de dados, responsável por instanciar objetos de usuário. Possui os atributos `id`, `name` e `email`. Seus métodos são apenas seu contrutor e seus *getters*;
- **Classe UserRequest:** Classe modelo de formulário, responsável por instanciar objetos de usuário que serão enviados no corpo de um requisição de criação ou atualização. Possui os atributos `name`, `email` e `password`. Seus métodos são:
  - `getters e setters`;
  - `convert()`: Método para criar um `User` a partir dos atributos da própria instancia de `UserRequest`;
  - `update()`: Método que recebe um `User` e atualiza suas propriedades com base nos atributos da própria instancia de `UserRequest`, dentro de uma transação;
- **Classe Profile:** Classe modelo responsável por instanciar entidades JPA de perfil de usuário. Possui os atributos `id` e `name`. Seus métodos são apenas seus *getters* e

[2] API padrão da linguagem Java que descreve uma interface comum para *frameworks* de persistência de dados.

setters. Esta classe é obrigatória para se trabalhar com controle de acesso no Spring Boot. Com ela é possível definir perfis com diferentes níveis de acesso, entretanto na presente aplicação, não foram definidos diferentes perfis;

- **Classe UserUtils:** Classe criada com o intuito de fornecer métodos utilitários estáticos relacionados a usuário. Possui até o presente momento um único método: `convert()`, que tem a função de converter uma lista de `User` em uma lista de `UserDto`;

- **Classe AuthenticationController:** Classe responsável por realizar o controle e mapeamento de rota do serviço de criação do *token* JWT de acesso. Possui atributos do tipo `AuthenticationManager` e `JwtService`. Conta somente com o método `auth()`, que recebe as credenciais do usuário, as valida e gera o *token* de acesso.

- **Classe JwtService:** Classe de serviço, responsável por fornecer métodos para criação e validação do *token* JWT de acesso. Possui os atributo `expiration` e `secret` (tempo de expiração e segredo do *token*). Seus métodos são:

  - `generateToken()`: Método para gerar o *token* de acesso;
  - `isValidToken()`: Método para validar o *token* recebido;
  - `getUserId()`: Método para extrair o identificador único do usuário a partir do *token* recebido;
  - `base64Encode()`: Método privado para codificar uma *string* em base64;
  - `getKey()`: Método privado para pegar a chave a partir dos *bytes* do segredo em base64 decodificado. Essa chave é utilizada na assinatura do *token* de acesso;

- **Classe AuthService:** Classe de serviço, necessária para "ensinar" ao Spring como buscar um usuário através do seu nome de usuário, que no caso da presente aplicação é o campo `email`, a fim de realizar o controle de acesso. Possui um único atributo do tipo `UserRepository` e um único método, `loadUserByUsername()`, utilizado para a função recentemente descrita;

- **Classe JwtFilter:** Classe necessária para validar *token* recebido e autenticar usuário na aplicação. Possui atributos do tipo `JwtService` e `UserRepository`. Seus métodos são:

  - `JwtFilter()`: Construtor;
  - `doFilterInternal()`: Método para validar o *token* recebido e autenticar usuário;
  - `getTokenFromRequest()`: Método privado para extrair o *token* da requisição;
  - `auth()`: Método privado para autenticar usuário com base no *token*;

- **Classe TokenDto:** Classe modelo de transferência de dados, responsável por instanciar objetos de *token*. Possui os atributos `token` e `type`. Seus métodos são apenas seu contrutor e seus *getters*;

- **Classe LoginRequest:** Classe modelo de formulário, responsável por instanciar objetos de *login* que serão enviados no corpo de um requisição de criação de um novo *token* de acesso. Possui os atributos `email` e `password`. Seus métodos são seus *getters* e *setters* e o método `convert()`, que é o responsável por criar uma instância de `UsernamePasswordAuthenticationToken` com base nos dados de *login*.

Já o diagrama de classes mostrado na figura 4.5 apresenta as classes restantes do transpilador, que correspondem ao controle de arquivos e a conversão de elementos Markdown em elementos LaTeX, com base em expressões regulares.

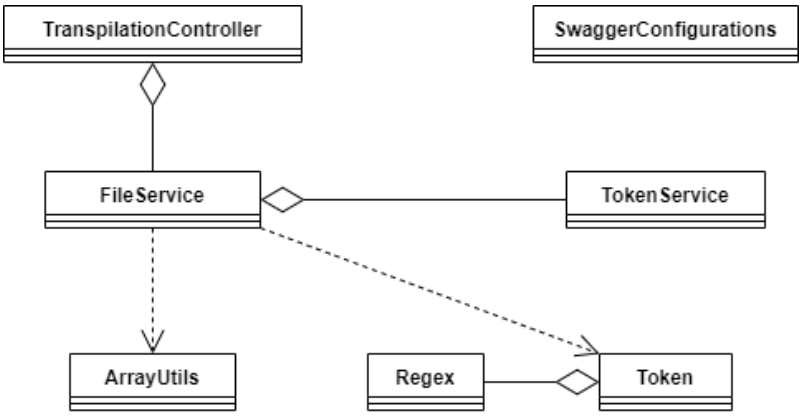

Figura 4.5: Diagrama de classes do controle de arquivos e conversão de elementos Markdown em elementos LaTeX do transpilador. Fonte: Própria autoria

- **Classe TranspilationController:** Classe responsável por realizar o controle e mapeamento de rota do serviço que cuida da transpilação de código Markdown em código LaTeX, a partir do arquivo recebido na requisição. Possui um único atributo, do tipo `FileService`, e também um único método, nomeado `uploadFile()`;

- **Classe SwaggerConfigurations:** Classe de configurações do *SpringFox* (biblioteca para gerar documentação *Swagger* de forma automatizada);

- **Classe FileService:** A classe importante da aplicação. É a classe que contém quase toda a lógica de transpilação. Possui os seguintes atributos:

  - `tokenService`, do tipo `TokenService`: Neste caso, a palavra "*token*" não deve ser confundida com o *token* JWT, pois se refere a classes de símbolos. O termo é usualmente utilizado dessa forma em análise léxica de compiladores. A classe *TokenService* será descrita posteriormente;
  - `content`, do tipo `String`: Utilizado para armazenar o conteúdo total do arquivo Markdown, recebido na requisição, em formato `String`;
  - `lines`, do tipo `List<String>`: Utilizado para armazenar o conteúdo de cada linha do mesmo arquivo, em formato de lista de `String`;
  - `currentLine`, do tipo `String`: Utilizado para armazenar a linha atual durante iterações;

- `currentIndex`, do tipo `int`: Mesmo que o anterior, mas neste caso para armazenar o índice. Este atributo é inicializado com valor igual a zero;
- `needOpenBegin`, do tipo `boolean`: Atributo booleano utilizado para auxiliar a aplicação da necessidade de abertura da tag `\begin{x}`, do LaTeX, onde "x" é um valor dinâmico. Este atributo é inicializado com valor igual a `true`;

e conta com os seguintes métodos:

- `transpileToLatex()`: Método público que recebe o arquivo Markdown, chama outros métodos privados para realizarem a transpilação e, por fim, cria um novo arquivo do tipo TeX para retorná-lo;
- `initializeReplacement()`: Método privado que inicializa as substituições de texto (Markdown por LaTeX) no atributo `content`;
- `stripComments()`: Método privado para remover comentários. Em Markdown os comentários são como os do HTML. Exemplo: `<!-- texto -->`;
- `stripMultipleEmptyLines()`: Método privado para remover sequências de duas ou mais linhas vazias;
- `replaceLineByLine()`: Método privado que analisa e substitui linha a linha o conteúdo a ser transpilado;
- `findAndReplaceTokens()`: Método privado que chama outros métodos para buscarem e substituírem certas classes de *tokens* Markdown por conteúdo LaTeX;
- `findAndReplaceEpigraph()`: Método privado para buscar e substituir o epígrafo;
- `findAndReplaceHeaders()`: Método privado para buscar e substituir cabeçalhos, como títulos de capítulos, seções e etc;
- `findAndReplaceEmphasis()`: Método privado para buscar e substituir ênfases textuais, como negritos e itálicos;
- `findAndReplaceLists()`: Método privado para buscar e substituir listas;
- `findAndReplaceReferences()`: Método privado para buscar e substituir referências e citações;
- `findAndReplaceBegins()`: Método privado para buscar e substituir "*begins*" genéricos como, por exemplo, o resumo, que no LaTeX é inserido por meio da tag `\begin{abstract}`;
- `findAndReplaceTables()`: Método privado para buscar e substituir tabelas;
- `findAndReplaceFigures()`: Método privado para buscar e substituir figuras;
- `convertTableAlign()`: Método privado para converter o alinhamento das colunas de tabelas, que no Markdown é definido por *tokens* do tipo ":—", ":—:" e "—:", enquanto no LaTeX são formados letras minúsculas, como "l", "c" e "r";

  - `convertTableRows()`: Método privado converter linhas dentro de tabelas. Este método possui três diferentes assinaturas (conceito conhecido como sobrecarga);

- **Classe TokenService:** Classe responsável por ler arquivos estáticos em formato JSON[3] que contêm os *tokens* da linguagem Markdown e suas *tags* correspondentes, de abertura e fechamento, em LaTeX. Todos os seus métodos e atributos estão relacionados à leitura desses arquivos;

- **Classe ArrayUtils:** Classe criada com o intuito de fornecer métodos utilitários estáticos relacionados à manipulação de listas e vetores. Possui até o presente momento um único método: `combineMatrix()`, que tem a função de converter um vetor bidimensional (matriz) de tamanho [M,N] em um vetor unidimensional de tamanho [MxN];

- **Classe Token:** Classe modelo responsável por mapear o conteúdo dos arquivos estáticos JSON que contêm os *tokens*. Possui os atributos `name`, `args`, `regex`, e `priority`. Além de seus *getters* e *setters*, possui o método `matcher()`, que recebe uma `String`, cria um `Pattern` com base na REGEX da instância atual, testa a `String` recebida e retorna um objeto do tipo `Matcher`;

- **Classe Regex:** Classe modelo responsável por mapear o conteúdo da REGEX contida em uma classe `Token`.

Por outro lado, o diagrama apresentado na figura 4.6, detalha a estrutura de recursos e funções da aplicação Markdemic, que é uma extensão para o editor de textos Visual Studio Code.

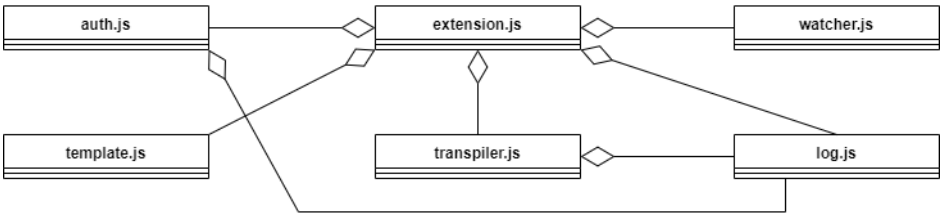

Figura 4.6: Diagrama estrutural da extensão Markdemic, para o Visual Studio Code. Fonte: Própria autoria.

- **Arquivo extension.js:** Arquivo principal. É neste arquivo que são criados e registrados os comandos da extensão;

- **Arquivo auth.js** Arquivo onde está localizada a função que consome o serviço de criação do *token* JWT no transpilador e cuida da autenticação;

- **Arquivo template.js** Arquivo onde estão localizadas as funções que aplicam as configurações iniciais realizadas pelo usuário, como título do documento e nome do autor, nos *placeholders* do *template* e cuidam da lógica de criação de arquivos;

[3] Acrônimo de *JavaScript Object Notation*

- **Arquivo transpiler.js** Arquivo onde está localizada a função que consome o serviço de transpilação de código Markdown em código LaTeX do transpilador;
- **Arquivo log.js** Arquivo para gerar *logs* dos tipos *info*, *warning* e *error*;
- **Arquivo watcher.js** Arquivo que contém a lógica do observador. Este possui a função de analisar quando arquivos de extensão Markdown (.md) ou BibTeX (.bib) sofrem alterações, a fim de, nesses casos, retranspilar ou recompilar o código.

4.5 Diagrama de Entidades e Relacionamentos

O diagrama de entidades e relacionamentos detalha a estrutura do banco de dados utilizado no transpilador, conforme ilustrado na figura 4.7. Como tudo relacionado à transpilação é obtido em tempo de execução, nada fica armazenado no banco, todavia, o que diz respeito a usuários, sim.

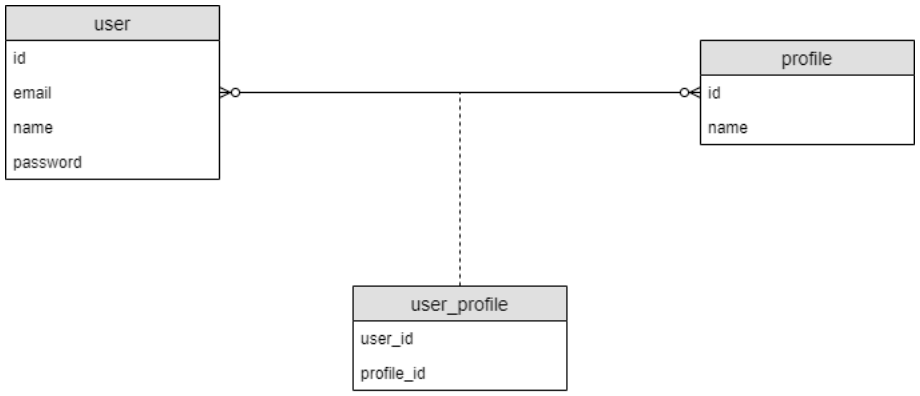

Figura 4.7: Diagrama de entidades e relacionamentos. Fonte: Própria autoria.

- **Entidade user:** Entidade que representa o usuário, seus atributos e seus relacionamentos. Possui relação de zero ou muitos para zero ou muitos com a entidade que representa o perfil.
- **Entidade profile:** Entidade que representa o perfil, seus atributos e relacionamento. Possui a relação com a entidade `user`, de cardinalidade zero ou muitos para zero ou muitos, conforme já mencionado;
- **Entidade user_profile:** Entidade associativa entre as entidades `user` e `profile`, devido a relação entre elas ser de muitos para muitos.

4.6 Ferramentas e Tecnologias Utilizadas

Neste capítulo serão mostradas as ferramentas, as tecnologias, os *frameworks* e as bibliotecas utilizadas no desenvolvimento do protótipo.

4.6.1 Spring Boot

Como já explicado, o transpilador se trata de uma Web API RESTful. Seu desenvolvimento foi realizado utilizando o Spring Boot, que é um *framework* para a criação de Web APIs em linguagem Java, Além disso, o Spring Boot facilita a importação de bibliotecas

externas, chamadas de dependências. Essas dependências devem ser declaradas de em um arquivo `pom.xml`, quando for um projeto Maven, ou `build.gradle`, quando se tratar de um projeto Gradle. No caso do transpilador, este foi criado como um projeto Gradle, fazendo uso dos seguintes módulos:

- **Spring Boot Starter Data JPA**: Módulo do próprio Spring Boot, utilizado para mapeamento e persistência de bancos de dados;
- **Spring Boot Starter Security**: Módulo do próprio Spring Boot, utilizado para auxiliar na implementação da segurança da API;
- **Spring Boot Starter Web**: Módulo do próprio Spring Boot para o desenvolvimento de aplicações Web;
- **Lombok**: Biblioteca focada em produtividade, que faz uso de anotações para evitar a criação de construtores, *getters* e *setters*;
- **JWT API, JWT Impl e JWT Jackson**: Três módulos que se complementam, para auxiliar na implementação da autorização e autenticação JWT;
- **Springfox Swagger**: Biblioteca para a criação automatizada de documentação Swagger;
- **Driver JDBC**[4]: Biblioteca utilizada para conectar ao banco de dados da aplicação.

4.6.2 Expresssões Regulares e objetos JSON

Neste trabalho, o transpilador faz uso de expressões regulares para encontrar elementos conversíveis de linguagem Markdown para linguagem LaTeX. Essas expressões são armazenadas em objetos JSON presentes na aplicação.

4.6.3 Visual Studio Code

O desenvolvimento de extensões para o editor de textos Visual Studio Code permite que quase todas as partes do editor possam ser editadas e aprimoradas, além de também permitir que sejam adicionados novos recursos e suporte a novas linguagens. No caso da extensão criada para o Markdemic, esta foi desenvolvida em linguagem NodeJS (JavaScript), com o auxilio dos seguintes módulos e dependências:

- **Path**: Módulo que fornece utilitários para trabalhar com caminhos de arquivos e diretórios;
- **File System (fs)**: Módulo que fornece uma API para interagir com o sistema de arquivos de uma maneira modelada em torno das funções padrão do POSIX[5];
- **HTTP**: Módulo para trabalhar com protocolo HTTP no NodeJS;
- **JWT-Decode**: Pequena biblioteca NPM para decodificar o *tokens* JWT codificados em base64;

[4] O nome deste módulo depende do SGBD utilizado. Exemplo: Para o MySQL, o nome do módulo é *MySQL Connector Java*.

[5] Família de normas para a manutenção de compatibilidade entre sistemas operacionais.

- **Child Process**: Módulo que fornece a capacidade de gerar processos filhos para executar comandos dentro de um *shell*[6];
- **Chokidar**: Biblioteca NPM para o monitoramento de alterações em sistemas de arquivos;
- **Form Data**: Biblioteca NPM para enviar formulários de *updoad* de arquivos (mime type `multipart/form-data`);
- **LaTeX Workshop**: Extensão para trabalhar com LaTeX no VSCode. Adicionada como dependência do Markdemic para usufruir do recurso de visualização de PDF dentro do editor;
- **Live Share**: Extensão incluída como dependência por fornecer recursos de edição colaborativa em tempo real.

[6]Interface de usuário para acessar os serviços de um sistema operacional.

5 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos, bem como os detalhes da implementação do protótipo da aplicação, os casos de testes e suas restrições e limitações.

5.1 Protótipo

Como pôde ser visto no capítulo anterior, o protótipo foi dividido em dois sistemas: Uma API REST independente, capaz de transpilar código-fonte Markdown em código LaTeX e uma extensão para o editor de textos Visual Studio Code, da Microsoft, responsável por consumir a API REST (transpilador), compilar o código LaTeX gerado pela API para produzir o PDF formatado, exibi-lo em tela dividida e fornecer ferramentas para a edição colaborativa síncrona. O motivo do desacoplamento foi a separação de papeis, permitindo que os usuários possam usufruir do transpilador sem que os obrigue a utilizar o Visual Studio Code.

Além disso, é importante salientar que a extensão está apta a produzir, inicialmente, apenas trabalhos de conclusão de curso, e que possuam modelo semelhante ao deste trabalho. Outros tipos de documentos científicos não serão possíveis de serem produzidos ao fazer uso do protótipo da extensão, contudo, o transpilador, caso utilizado individualmente, sim.

5.1.1 Transpilador

Como dito anteriormente, o transpilador se trata de uma API REST. Seu desenvolvimento foi realizado em linguagem Java, por meio do *framework* SpringBoot. Como pré-requisito, é necessário que o usuário tenha um banco de dados relacional instalado na máquina onde rodará a API para o armazenamento de dados de usuários. Os dados do banco, como a classe do driver do SGBD[1], a URL e as credenciais de acesso, devem ser configuradas previamente no arquivo `application.yml` (figura 5.1).

```yaml
# datasource
spring:
    datasource:
        driverClassName: com.mysql.cj.jdbc.Driver
        url: jdbc:mysql://localhost:3306/markdemic?useSSL=false&useTimezone=true&serverTimezone=UTC
        username: transpiler
        password: Dsu2hw¨%sba56h(#sa
        initialization-mode: never
```

Figura 5.1: Configurações do banco de dados no transpilador. Fonte: Própria autoria.

[1] Sistemas de Gestão de Base de Dados

5.1.1.1 *Serviço para gerar token de acesso*

A segurança dessa API é feita via autenticação JWT e, para isso, foi exposto na API um serviço para gerar o *token* de acesso, com duração de vinte e quatro horas. O corpo da requisição desse serviço deve conter os dados de *login* do usuário, que são e-mail e senha. Portanto, o serviço verificará se as credenciais são válidas e, caso sejam, gerará o *token* (ver figura 5.2), utilizando o algorítimo HS256[2] (HMAC com SHA-256) para assiná-lo.

Esse serviço é o único não autenticado, ou seja, onde não há uma verificação de autorização. Todos os outros deverão enviar no cabeçalho da requisição HTTP o mapa `Authorization: Bearer TOKEN_GERADO`.

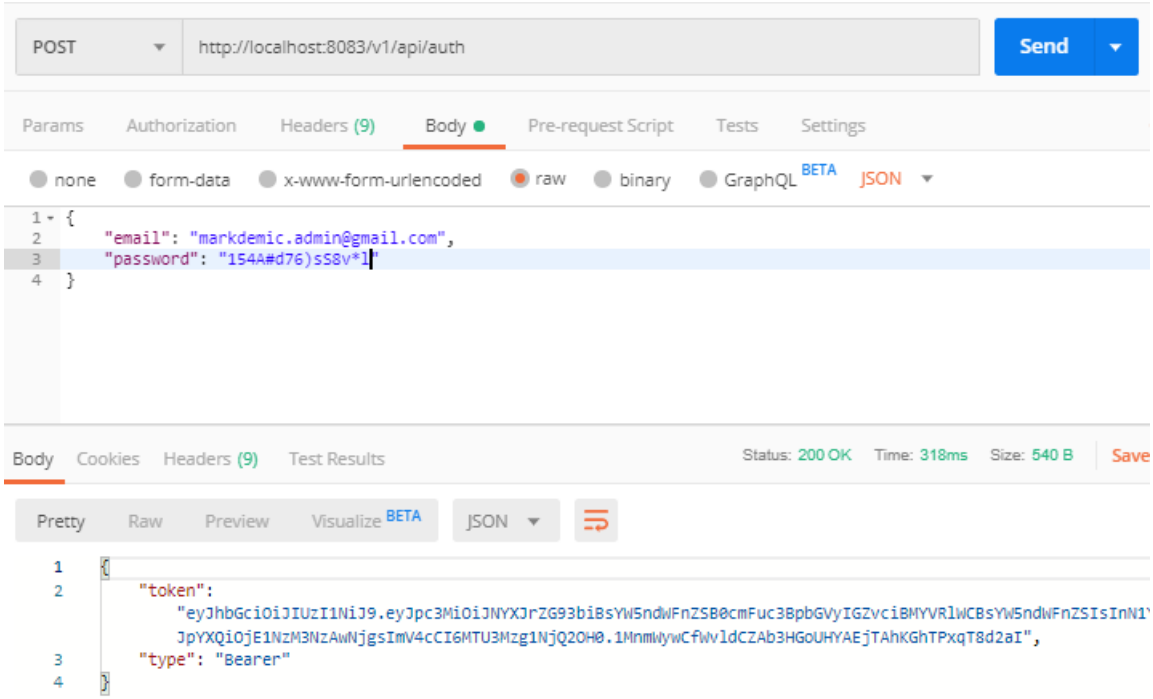

Figura 5.2: Requisição e resposta ao serviço para gerar token de acesso utilizando o sistema Postman. Fonte: Própria autoria

5.1.1.2 *Serviços de usuário*

A API conta com um CRUD[3] de usuário, contendo cinco serviços: Buscar todos os usuários, buscar usuário por ID, criar novo usuário, atualizar usuário e deletar usuário. Além disso, é utilizado o conceito de modelos DTO[4] (*Data transfer object*) nas respostas das requisições, de modo a não devolver informações sensíveis, como senha, por exemplo.

5.1.1.3 *Serviço de transpilação*

O serviço de transpilação é o mais importante da API. Ele é o responsável por receber o arquivo Markdown como entrada e devolver o arquivo LaTeX como saída. Para tal, o

[2]Utilizado para criar assinaturas envolvendo uma combinação de uma função de *hash* e uma chave (secreta).

[3]Acrônimo do inglês *Create*, *Read*, *Update* and *Delete*

[4]Padrão de projeto de software usado para transferir dados entre subsistemas de um software.

transpilador conta com uma base de arquivos estáticos, em formato JSON, contendo objetos que possuem expressões regulares – estas servem para capturar os elementos dentro do texto Markdown, como cabeçalhos, negritos, itálicos, listas, tabelas e imagens – e o código LaTeX de abertura e fechamento, correspondente aos respectivos elementos (ver figura 5.3).

```
{
    "name": "figure",
    "regex": {
        "markdown": "(?:!\\[([^\\s]+)\\]\\((?!http)([^\\s]+\\.(?
        :jpe?g|gif|png|svg|pdf))(?:\\s?\"(.*)\")?\\))(?:\\{([0-9.]+)
        \\})?",
        "latexOpen": "\\\\begin{figure}[h]
        \n\t\\\\centering\n\t\\\\includegraphics[scale=?]{?}
        \n\t\\\\caption{?}\n\t\\\\label{?}",
        "latexClose": "\n\\\\end{figure}"
    }
}
```

Figura 5.3: Exemplo de arquivo estático contendo a REGEX do elemento Markdown figura e seus correspondentes LaTeX de abertura e fechamento. Fonte: Própria autoria.

Por conseguinte, toda vez que a aplicação recebe uma requisição para esse serviço, o sistema lê o arquivo enviado, busca por elementos Markdown a serem substituídos por elementos LaTeX dentro do texto, com o auxílio das expressões regulares presentes nos arquivos estáticos, os substitui e, por fim, retorna como resposta um novo arquivo transpilado, no formato LaTeX, como pode ser visto na figura 5.4.

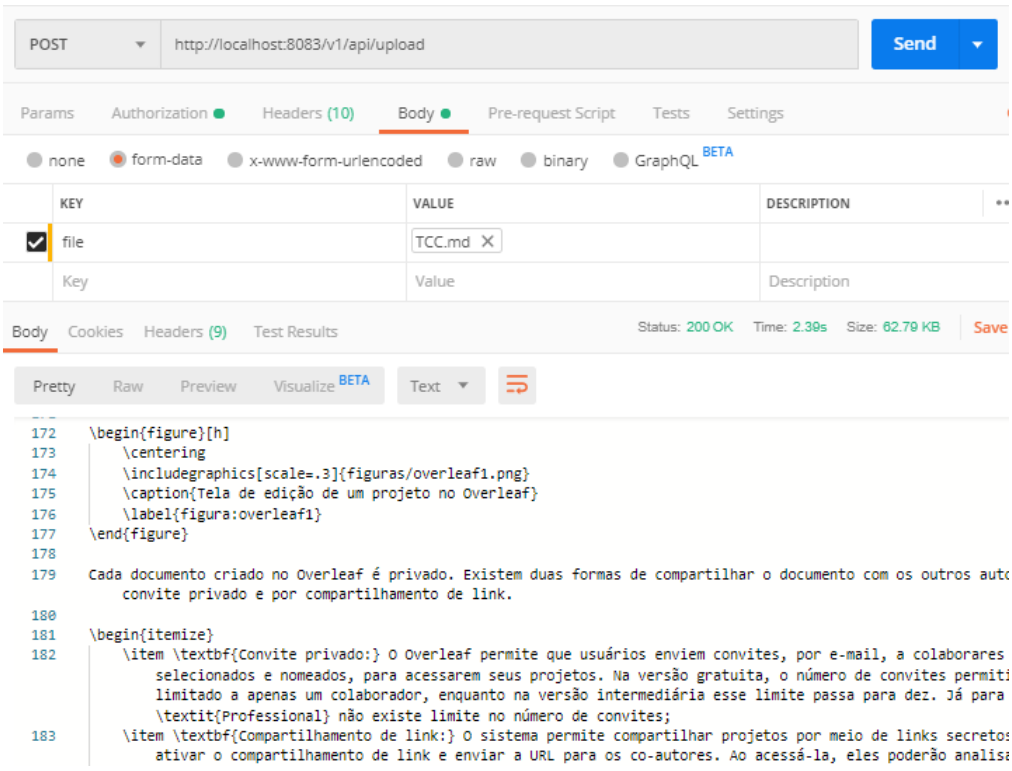

Figura 5.4: Requisição e resposta ao serviço de transpilação de Markdown para LaTeX utilizando o sistema Postman. Fonte: Própria autoria.

5.1.1.4 *Documentação*

Além dos serviços mencionados, foi implementada na API uma documentação *Swagger*[5] automatizada, com o auxilio do conjunto de bibliotecas *SpringFox*, que funciona examinando a aplicação em tempo de execução, para inferir a semântica da API com base nas configurações do *Spring*, na estrutura de classes e nas anotações Java em tempo de compilação.

A documentação é acessível via *browser*, desde que a API esteja em execução, através do seguinte endereço: `http://api-domain/v1/api/swagger-ui.html`. Na documentação é possível visualizar os dados de requisição e resposta, *endpoint*[6] e método HTTP de cada um dos serviços expostos, como pode ser observado na figura 5.5. Além disso, também é possível testar as chamadas dos serviços.

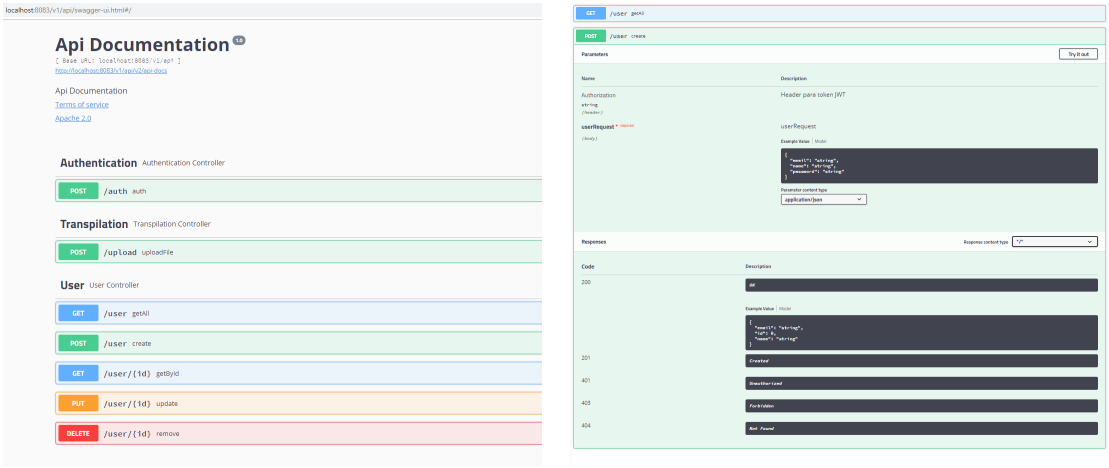

Figura 5.5: Serviços do transpilador (esquerda) e detalhes do serviço de criação de usuário (direita) na documentação *Swagger*. Fonte: Própria autoria.

5.1.2 Extensão para Visual Studio Code (Markdemic)

A extensão para o VSCode foi implementada em linguagem JavaScript. Como dito no início deste capítulo, suas funções são: Consumir a API do transpilador a cada vez que o arquivo Markdown for salvo ou o comando `transpile` for chamado manualmente no editor de texto, compilar o código LaTeX gerado para produzir o PDF formatado, exibi-lo em tela dividida e fornecer ferramentas para a edição colaborativa síncrona. Além disso, o Markdemic – nome dado à extensão – possui dependência de outras duas extensões, no entanto, não é necessários instalá-las de antemão, pois o Markdemic se encarregará de instalá-las, caso ainda não tenham sido instaladas, durante a sua própria instalação. Os pré-requisitos são:

1. Instalação do Visual Studio Code na versão 1.39.0 ou superior;
2. Distribuição TeX, incluindo PdfLaTeX e BibTeX. Exemplo: TeXLive;
3. Inclusão da distribuição TeX no PATH das variáveis de ambiente do sistema operacional;

[5]Framework *open source* que auxilia desenvolvedores no desenho, especificação e documentação de APIs

[6]URL onde o serviço pode ser acessado por uma aplicação cliente.

4. API do transpilador em execução e acessível.

5.1.2.1 Configurações iniciais

Logo após instalar o Markdemic, é necessário que o usuário faça algumas configurações iniciais, antes de começar a usá-lo. Estas incluem configurações do transpilador e das páginas iniciais do documento. As do transpilador são: *Host*, porta, *login* e senha; Do documento: Título, nome e sobrenome do autor, instituição, data, cidade, estado, nome e sobrenome do co-autor (caso este exista), nome, sobrenome e título do orientador, nome, sobrenome, título e instituição dos membros da banca, palavras-chave, entre outras (ver figura 5.6). Essas configurações podem ser realizadas no menu `File/Preferences/Settings`, ou, alternativamente, utilizando o atalho `Ctrl+,` (*Control*, vírgula).

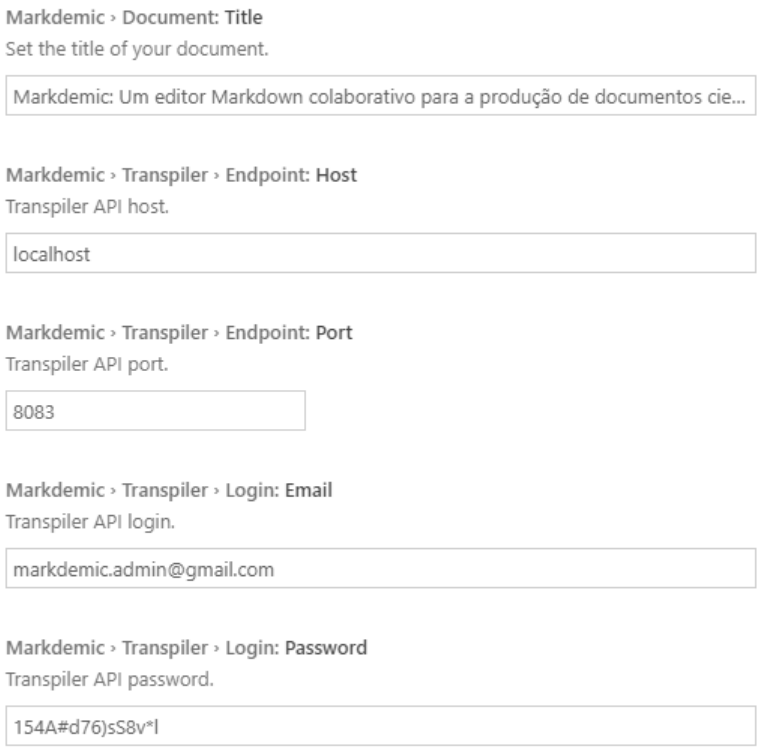

Figura 5.6: Exemplo de algumas configurações iniciais do Markdemic. Fonte: Própria autoria.

5.1.2.2 Template LaTeX

O Markdemic tem, por padrão, o *template* LaTeX do curso de Tecnologia em Análise em Desenvolvimento de Sistemas, da instituição de ensino IFRS - Campus Rio Grande. É possível alterá-lo manualmente, caso seja necessário, através da edição dos arquivos da extensão, no diretório onde ficam as extensões instaladas do VSCode, contudo, existem algumas regras que deverão serem seguidas.

O *template* LaTeX padrão do Markdemic conta com cinco arquivos: `abnt.bst`, `compila.bash`, `ifrs.cls`, `ifrs.sty` e `TCC.tpl`. Este último é o arquivo principal. Ele possui *placeholders*[7] (ver figura 5.7) que, durante o processo de transpilação

[7]Espaços reservados

de Markdown para LaTeX, receberão o conteúdo das configurações iniciais, como título do documento, autor e demais configurações e o conteúdo proveniente do transpilador (que seria o núcleo documento, ou seja, o conteúdo do elemento `\begin{document}`. Portanto, a alteração diretamente nesses arquivos é única forma, até momento, de se modificar o *template* da ferramenta.

```
\documentclass[oneside]{ifrs}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{graphicx}
\usepackage{times}
\usepackage{listings}
\usepackage{multirow}
\usepackage{scalefnt}
\usepackage{amsmath}
\usepackage{rotating}
\usepackage{subfigure}
\usepackage{verbatim}
\usepackage{dirtytalk}

\bibliographystyle{abnt}

\hyphenation{en-si-na-men-tos a-gra-de-ci-men-to}
\hyphenation{tec-no-lo-gi-a}

\author{@{author.lastname}}{@{author.firstname}}

\title{@{title}}

\advisor[@{advisor.title}. ]{@{advisor.lastname}}{@{advisor.firstname}}

\location{@{location.city}}{@{location.state}}

\revisor[@{revisor1.title}. ]{@{revisor1.lastname}}{@{revisor1.firstname}}{@{revisor1.institution}}
\revisor[@{revisor2.title}. ]{@{revisor2.lastname}}{@{revisor2.firstname}}{@{revisor2.institution}}

\date{@{date.day}}{@{date.month}}{@{date.year}}

\keyword{@{keyword1}}
\keyword{@{keyword2}}
\keyword{@{keyword3}}

\begin{document}

\maketitle

@{content}

\bibliography{bibliografia}

\end{document}
```

Figura 5.7: Arquivo TCC.tpl do template LaTeX do Markdemic. Fonte: Própria autoria.

5.1.2.3 Espaço de trabalho

O espaço de trabalho deverá conter, inicialmente, apenas dois arquivos: `TCC.md` e `bibliografia.bib`. Estes serão os arquivos que sofrerão alterações por parte dos usuários. Todavia, após processo de compilação, serão gerados arquivos auxiliares e o arquivo `TCC.pdf`, que é saída final do documento.

No arquivo `TCC.md` estará presente o núcleo do documento, com todos os capítulos, seções, subseções e assim por diante. A linguagem Markdown utilizada sofreu algumas pequenas modificações, para se adequar às necessidades da produção científica, como a adição de títulos em tabelas e *labels* em tabelas e figuras. Essas adequações da linguagem serão apresentadas mais adiante. Já no arquivo `bibliografia.bib`, deverá estar presente a bibliografia no formato BibTeX padrão, pois foi decidido não utilizar Markdown para a bibliografia, visto que, além de ser um modelo mundialmente conhecido, difundido e consolidado, diversas ferramentas na Web já fornecem código BibTeX de livros e artigos de forma pronta.

5.1.2.4 Dependências

Conforme fora colocado anteriormente, o Markdemic depende de outras duas extensões do VSCode, ambas instaladas automaticamente durante a instalação do Markdemic. São elas: *LaTeX-Workshop* e *Live Share*, esta desenvolvida pela própria Microsoft.

A extensão *LaTeX-Workshop* tem a função de exibir o resultado em uma *webview*[8] no editor, como pode ser visto na figura 5.8. Ela foi utilizada no projeto emergencialmente, pois não era parte do escopo. Sua necessidade se deu devido ao pouco tempo hábil para o desenvolvimento da *webview* que seria responsável pela exibição da saída final do documento (`TCC.pdf`), em tela divida. Por outro lado, a compilação do código LaTeX, apesar de ser um recurso presente na extensão *LaTeX-Workshop*, foi implementada no próprio Markdemic, com o auxílio do módulo do NodeJS `child_process`, que permite executar comandos de terminal dentro da aplicação.

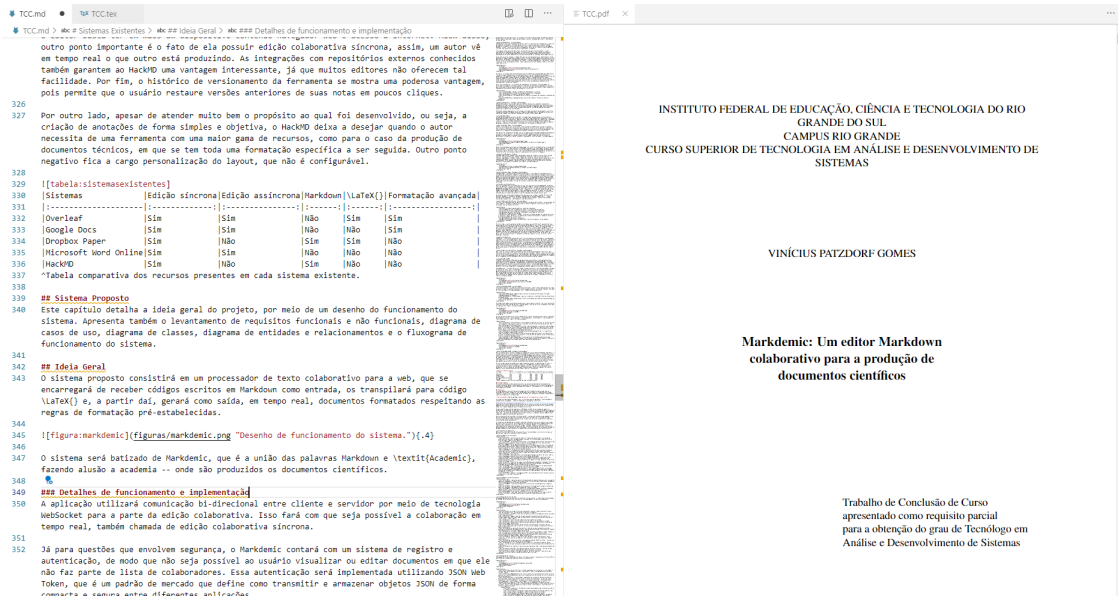

Figura 5.8: Exibição do PDF final em uma *Webview* gerada pela *LaTeX-Workshop*. Fonte: Própria autoria.

Já a extensão *Live Share* tem a função de compartilhar espaços de trabalho com outros usuários, para serem editados colaborativamente e em tempo real. Ao ser instalada, essa extensão adiciona um ícone na barra esquerda do editor. Este da acesso a três opções de sessão, que são:

- **Join collaboration session**: Possibilita entrar em uma sessão de colaboração já existente;
- **Start collaboration session**: Possibilita criar uma nova sessão de colaboração para o espaço de trabalho atual;
- **Start read-only collaboration session**: Possibilita criar uma nova sessão de colaboração, onde os participantes não terão permissão de edição.

Existe ainda a opção para convidar participantes, onde um e-mail é enviado, convidando o possível participante a fazer parte da sessão de colaboração.

[8] Conteúdo web exibido dentro de um sistema não-web.

5.1.2.5 Comandos

O desenvolvimento de extensões para Visual Studio Code fornecem recursos para que sejam registrados comandos. Esses comandos lançam eventos que executam ações específicas. Para o Markdemic foram implementados três comandos:

- **Generate LaTeX code** (`transpile`): Comando responsável por consumir a API do transpilador, executar o comando interno `compile` e atualizar a *webview* de exibição do documento PDF, caso esta esteja aberta. Tudo ocorre de forma síncrona, pois cada ação depende da finalização da ação anterior. Atalho: `Ctrl+Alt+B`;
- **Compile LaTeX code** (`compile`): Comando interno responsável por compilar o BibTeX e o LaTeX gerado, para produzir o documento PDF. Por se tratar de um comando intenro, não foi associado um atalho ao mesmo;
- **Open PDF viewer** (`pdfviewer`): Comando responsável por gerar a *webview* de exibição do documento PDF. Atalho: `Ctrl+Alt+V`.

5.1.2.6 Observador

O comando `transpile` pode ser executado manualmente a qualquer momento, todavia, isso não se faz necessário, pois foi implementado no Markdemic, com o auxilio do módulo NPM *Chokidar*, uma funcionalidade capaz de identificar quando existe alterações nos arquivos `TCC.md` e `bibliografia.bib`. Deste modo, sempre que os usuários alterarem um desses arquivos e salvarem as alterações, o observador executará internamente o comando `transpile`, gerando um novo código LaTeX, compilando-o e atualizando a *webview* de exibição do documento.

5.1.3 Markdown utilizado no Markdemic

Conforme introduzido anteriormente, houve pequenas mudanças e acréscimos de novos elementos na linguagem Markdown, com o intuito de fazê-la suprir as necessidades das normas técnicas em documento científicos. A tabela 5.1 apresenta todos os elementos e a que cada um corresponde.

```
![label]
|cabeçalho|cabeçalho|cabeçalho|cabeçalho|cabeçalho|cabeçalho|
|:--------|:-------:|:-------:|:-------:|:-------:|:-------:|
|  texto  |  texto  |  texto  |  texto  |  texto  |  texto  |
|  texto  |  texto  |  texto  |  texto  |  texto  |  texto  |
^título
```

Figura 5.9: Como inserir uma tabela no Markdown utilizado no Markdemic. Fonte: Própria autoria.

Como pôde ser visto na tabela 5.1, muitos elementos LaTeX não tiveram um correspondente Markdown implementado, como, por exemplo, a inserção de equações matemáticas. Em vista disso, o transpilador está apto a receber código misto, ou seja, um texto em que exista tanto código Markdown, quanto código LaTeX, sem que isso interfira no processo de transpilação. Portanto, o usuário poderá utilizar LaTeX sempre que não houver um correspondente Markdown.

Tabela 5.1: Códigos Markdown válidos para o transpilador. Fonte: Própria autoria.

Elemento	Código Markdown
Itálico	`*texto*` ou `_texto_`
Negrito	`**texto**` ou `__texto__`
Fonte monoespaçada	``` ``texto`` ```
Epígrafe	`>> [autor] frase`
Resumo	`[abstract]{texto}`
Lista de siglas e abreviaturas	`- [SIGLA] significado`
Lista não-ordenada	`- texto`
Lista enumerada	`1. texto`
Capítulo	`# nome`
Seção	`## nome`
Subseção	`### nome`
Subsubseção	`#### nome`
Citação direta	`> texto`
Citação indireta	`:[autor]`
Citação indireta (ano)	`:[autor][y]`
Citação indireta (autor)	`:[autor][a]`
Referência	`%[label]`
Nota de rodapé	`texto^{nota}`
Figura	`![label](url "título"){escala}`
Tabela	Ver figura 5.9

5.2 Casos de testes

Para validar as funcionalidades do protótipo, foram montados cenários de testes utilizando um trabalho de conclusão de curso pronto, transcrito para Markdown, com alguns trechos em LaTeX, visando também validar a capacidade do transpilador de trabalhar com código misto. Ademais, foram validados cenários envolvendo questões de acesso e exibição do PDF.

5.2.1 Acesso não autorizado

Como a API do transpilador só permite acesso aos seus serviços expostos mediante autenticação via JWT, primeiramente foram realizados testes com credenciais de usuários nulas, inexistentes, ou contendo senha inválida. Conforme esperado, em nenhuma das três tentantivas foi possível obter o *token* de acesso, interrompendo a execução do comando `transpile` do Markdemic (ver figura 5.10).

```
INFO [21/11/2019 14:20:08] - Congratulations, your extension "markdemic" is now active!
INFO [21/11/2019 14:20:10] - Generating access token...
WARNING [21/11/2019 14:20:10] - Could not get access token
```

Figura 5.10: *Log* de alerta do Markdemic por motivo de não obtenção do *token* de acesso. Fonte: Própria autoria.

5.2.2 Transpilação

Após foi testada a transpilação de elementos do Markdown para o LaTeX, desde ênfases, como negrito e itálico, passando por epígrafe, resumo, listas, cabeçalhos, citações e referências, figuras e tabelas.

5.2.2.1 Textos com ênfases

O teste de textos com ênfases se mostrou satisfatório. A transpilação de tais elementos ocorreu conforme esperado, como pode ser visto nas figuras 5.11 e 5.12, onde são mostrados, respectivamente, o negrito e o itálico, este marcado em verde para facilitar a sua visualização.

5.2.2.2 Epígrafe

Assim como com as ênfases, o teste com o elemento epígrafe transcorreu com sucesso, conforme mostrado na figura 5.13.

5.2.2.3 Resumo

Na figura 5.14 pode ser visto o resultado do teste do elemento resumo. O mesmo foi satisfatório.

5.2.2.4 Listas

Foram testados os três tipos de listas implementados no transpilador: Não-ordenada, enumerada (ambas na figura 5.15) e de siglas e abreviaturas (figura 5.16). A transpilação dos três tipos ocorreu como esperado.

5.2.2.5 Cabeçalhos

São ditos cabeçalhos os capítulos, as seções, as subseções e as subsubseções. Não foram encontrados erros nos testes envolvendo cabeçalhos, conforme demonstrados na figura 5.17.

5.2.2.6 Citações, referências e figuras

Os testes envolvendo citações, referências e figuras englobaram citações do tipo direta e indireta e a referência a uma figura inserida no texto, como pode ser visto na figura 5.18.

5.2.2.7 Tabelas

O teste envolvendo tabelas se mostrou satisfatório, inclusive no que diz respeito ao alinhamento das colunas, que no Markdown é definido pelo caractere ":" (dois pontos) nas células da segunda linha da tabela (linha divisória). Quando colocados à esquerda dos traços, ou, opcionalmente, não colocados, o alinhamento da coluna deverá estar à esquerda, enquanto quando colocados à direita, deste mesmo mesmo lado. Já para alinhamento centralizado, o caractere ":" deverá ser colocado de ambos os lados dos traços (ver figura 5.19).

5.2.3 Edição colaborativa

Conforme descrito na seção 5.1, o recurso de edição colaborativa no Markdemic é fornecido por meio de outra extensão, a *Live Share*. Os testes realizados nessa ferramenta foram satisfatórios, apresentando bom rendimento, inclusive enquanto os usuários

escrevem com cursores sobrepostos. Neste caso, a extensão trata de "empurrar" um dos usuários para a linha imediatamente inferior. Na figura 5.20 é possível ver um segundo usuário editando o mesmo arquivo `TCC.md` em tempo real.

Entretanto, durante os testes um ponto negativo chamou a atenção. O comando "desfazer" (Ctrl+Z) do Visual Studio Code, desfaz a última entrada no arquivo, independente de qual usuário esta tenha partido, assim, acredita-se que não seja bom praticá-lo enquanto usuários estiverem em uma sessão compartilhada, de modo a evitar desfazer entradas de outros colaboradores.

5.3 Restrições e limitações

Nesta seção serão mostradas as restrições e as limitações do protótipo desenvolvido, como as questões antes mencionadas, relacionadas aos elementos não implementados no transpilador e outras questões envolvendo a extensão criada para o Visual Studio Code.

5.3.1 Transpilador

A maior parte das restrições estão presentes no transpilador, pois, por motivo de tempo, não foi possível implementar todas as conversões de Markdown para LaTeX inicialmente planejadas, deixando muitos elementos importantes do LaTeX sem um correspondente em Markdown. O que se teve, foi o que no mercado é conhecido como MVP[9] (*Minimum Viable Product*). Segue abaixo lista de alguns desses elementos:

- `\begin{description}`: Cria lista de descrições;
- `\begin{equation}`: Para adicionar equação;
- `\begin{quotation}`: Como um bloco de citação, mas com indentação de parágrafo;
- `\begin{verse}`: Bloco de citação para verso;
- `\begin{center}`: Alinha ao centro;
- `\begin{flushleft}`: Alinha à esquerda;
- `\begin{flushright}`: Alinha à direita;
- `\begin{minipage}`: Cria uma mini-página dentro de um corpo;
- `\begin{array}{especificação}`: Cria um vetor;
- `\pageref{marcador}`: Exibe a página de uma marcador (*label*);
- `\underline{texto}`: Sublinha um texto;
- `\linespread{x}`: Substitui espaços pelo multiplicador x;
- `\hspace{tamanho}`: Adiciona espaços horizontais (Exemplo: tamanho = 20pt);
- `\vspace{tamanho}`: Adiciona espaços verticais (Exemplo: tamanho = 20pt);

[9] Do inglês, produto mínimo viável

- `\multicolumn{n}{especificação}{texto}`: Célula com n colunas mescladas em uma tabela;

Além dessa lista de elementos, existem outras limitações, como não ser possível alterar o leiaute de tabelas; Utilizar outra unidade, que não a unidade "escala", para o tamanho de imagens; Aninhamento de listas; entre outras.

5.3.2 Extensão para Visual Studio Code

Em geral, as limitações envolvendo a extensão Markdemic são, em maioria, relacionadas à falta de elementos visuais que tragam ao usuário a ideia de que algo está sendo processado quando, por exemplo, o código LaTeX está sendo compilado, ou quando o sistema está aguardando a resposta do transpilador. Isso pode trazer ao usuário o sentimento de que o comando executado não funcionou conforme o esperado. Além disso, não foram implementados elementos visuais que indiquem ao usuário a ocorrência de erros no sistema, assim como no transpilador, também por questão de falta de tempo.

A dependência *LaTeX Workshop* é uma extensão robusta, com diversos recursos disponíveis para usuários LaTeX, no entanto, um único recurso é realmente utilizado pelo Markdemic: A exibição do PDF dentro do editor (*webview*). Isso faz com que o Markdemic carregue consigo uma grande quantidade de código não utilizado, o que somente ocupará espaço no disco do computador do usuário.

Outrossim, a edição colaborativa do Markdemic está restrita a funcionar apenas através da extensão *Live Share*, que também é registrada como uma dependência, pois não haveria tempo hábil para desenvolver tal recurso.

5.4 Comparativo entre os sistemas estudados e o Markdemic

Tabela 5.2: Comparação dos recursos presentes em cada sistema existente estudado e o Markdemic. Fonte: Própria autoria.

Sistemas	Edição colaborativa*	Interpreta		Produção acadêmica	Plataforma**	Máximo colab.
		Markdown	LaTeX			
Overleaf	S/A	Não	Sim	Sim	W	2
Google Docs	S/A	Não	Não	Sim	W/M	Indef.
D. Paper	S	Sim	Sim	Não	W/M	Indef.
MS Word	S/A	Não	Não	Sim	W/D/M	Indef.
HackMD	S	Sim	Não	Não	W	2
Markdemic	S/A	Sim	Sim	Sim	D	Indef.

** S: Síncrona; A: Assíncrona.*

*** W: Web; D: Desktop; M: Mobile.*

Figura 5.11: Texto em negrito em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

Figura 5.12: Texto em itálico em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

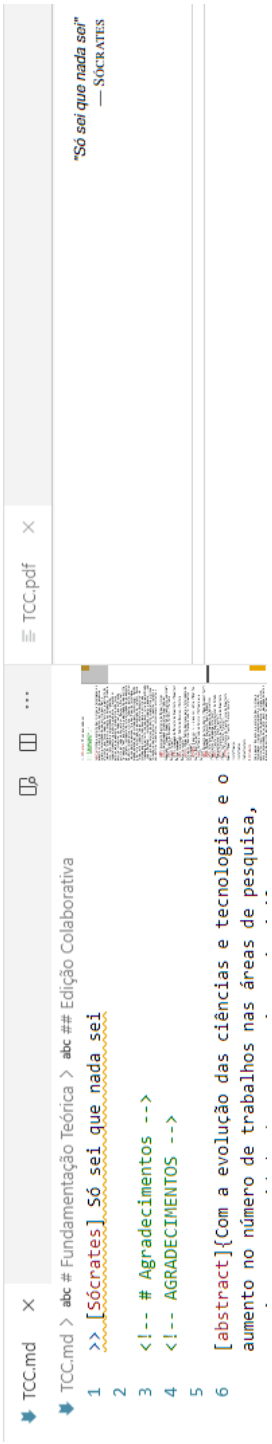

Figura 5.13: Epígrafe em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

Figura 5.14: Resumo em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

Figura 5.15: Lista enumerda e lista não-ordenada em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

Figura 5.16: Lista de siglas e abreviaturas em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

Figura 5.17: Cabeçalhos em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

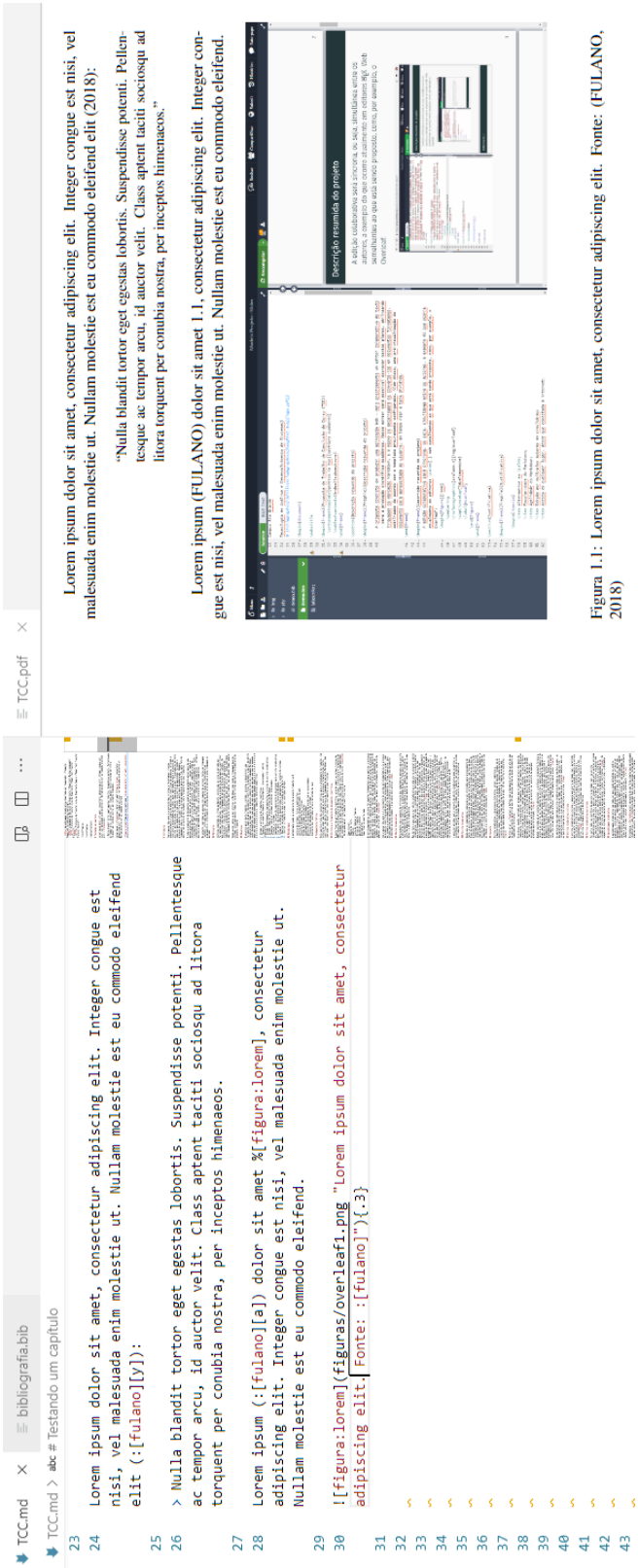

Figura 5.18: Citações, referências e a inserção de uma figura em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

Figura 5.19: Inserção de tabelas em Markdown e seu resultado final no Markdemic. Fonte: Própria autoria.

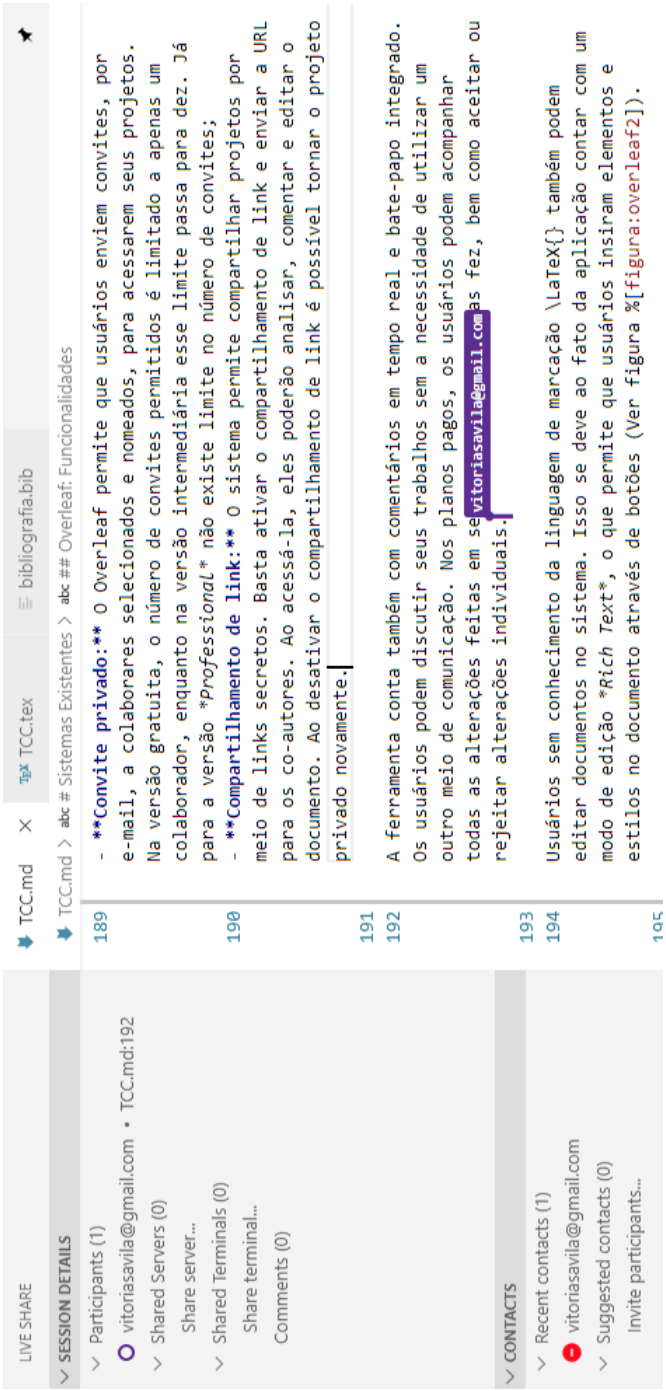

Figura 5.20: Edição colaborativa em tempo real através da extensão *Live Share*. Fonte: Própria autoria.

6 CONCLUSÃO

No presente trabalho foi apresentada a proposta de uma ferramenta focada na produção colaborativa de documentos científicos, a partir da conversão em tempo real de arquivos Markdown.

Para isso, buscou-se compreender os conceitos por trás da escrita colaborativa de documentos, principalmente no que diz respeito a escrita colaborativa síncrona. Também foram realizados estudos acerca da elaboração de documentos científicos, das linguagens Markdown e LaTeX, da construção de Web APIs RESTful, do uso de expressões regulares com intuito de encontrar cadeias de caracteres específicas e do desenvolvimento de extensões para o editor de textos Visual Studio Code, da Microsoft. Além disso, pesquisou-se sobre sistemas e implementações existentes, de modo a levantar suas características, pontos fracos e pontos fortes. Além do mais, realizou-se a análise do projeto, descrevendo a arquitetura, diagramas, fluxos e as ferramentas a utilizadas. Por fim, foram demonstrados os resultados obtidos, os casos de testes e as restrições e limitações presentes no protótipo desenvolvido.

Conclui-se que este trabalho cumpriu com os objetivos estabelecidos, haja visto que os resultados obtidos foram satisfatórios no que tange a proposta de análise e desenvolvimento de sistemas.

6.1 Trabalhos futuros

Como comentado em capítulos anteriores, devido à falta de tempo hábil, algumas funcionalidades não foram desenvolvidas, deixando-as então para implementações futuras. Dentre os principais trabalhos futuros a serem realizados estão:

- Hospedar a API do transpilador em um domínio de acesso público, eliminando a necessidade de execução manual da API por parte dos usuários;
- Fornecer, no mesmo domínio, web site contendo formulário de *login* para cadastro dos usuários, visto que o registro é uma necessidade para se obter o *token* JWT de acesso;
- Ampliar a capacidade de transpilação da API, alimentando-a com novas expressões regulares, de modo que ela possa transpilar, de Markdown para LaTeX, outros elementos que atualmente não são convertidos;
- Implementar funcionalidade para criar *webview* de exibição do PDF final na extensão desenvolvida para o Visual Studio Code (Markdemic), dispensando a necessidade de dependência da extensão LaTeX Workshop, atualmente utilizada apenas para essa função;

- Tornar o template do documento parte do espaço de trabalho, de modo que este possa ser editado colaborativamente;
- Permitir a personalização de expressões regulares, bem como a inclusão de novas expressões por parte dos usuários;
- Permitir a inclusão de *placeholders* dinâmicos nas configurações iniciais da extensão Markdemic, de forma que os usuários possam criar documentos científicos de outros tipos, que não só trabalhos de conclusão semelhantes a este trabalho.

REFERÊNCIAS

ALAM, K. **Learn Markdown**: the complete guide on markdown formatting. [S.l.]: INDEPENDENTLY PUBLISHED, 2018.

ARLOING, T.; DUBOIS, Y.; DUCASSE, S.; CASSOU, D. Pillar: a versatile and extensible lightweight markup language. , [S.l.], p.25, 2016.

BRITANNICA, E. **Microsoft Word History**. Acessado em: 21/08/2019, https://www.britannica.com/technology/Microsoft-Word.

CORDEIRO, E. d. C. A.; JOAQUIM, C. H.; CEDRAN, D. H. Tutorial de uso do LaTex para escrita científica. , [S.l.], 2013.

DILLENBOURG, P. **What do you mean by collaborative learning?** Acessado em: 29/04/2019, https://telearn.archives-ouvertes.fr/hal-00190240/document.

DROPBOX. **Dropbox Paper**: site oficial. Acessado em: 16/06/2019, https://www.dropbox.com/paper/guide.

EDE, L. S.; EDE, L.; LUNSFORD, A. A. **Singular texts/plural authors**: perspectives on collaborative writing. [S.l.]: SIU Press, 1990.

GALVÃO, J.; SOARES, F.; KAI, P. Modelo de API REST para integração de sistemas biometricos agnosticos. , [S.l.], 2017.

GITTER, S. **Markdown basics**. Acessado em: 18/11/2019, https://gitter.zendesk.com/hc/en-us/articles/200176682-Markdown-basics.

GOOGLE. **Google Documentos**: site oficial. Acessado em: 08/06/2019, https://www.google.com/docs/about.

HACKMD. **HackMD**: site oficial. Acessado em: 12/05/2019, https://hackmd.io/.

HERRERO, A. **Instant Markdown**: learn to efficiently manage your content and use different services with markdown. [S.l.]: Packt Publishing, 2013.

JARGAS, A. M. **Expressões regulares**: uma abordagem divertida. [S.l.]: Novatec Editora, 2009.

LAMPORT, L. **LATEX**: a document preparation system: user's guide and reference manual. [S.l.]: Addison-wesley, 1994.

LIVESHARE. **Live Share**: site oficial. Acessado em: 19/12/2019, https://docs.microsoft.com/en-us/visualstudio/liveshare/quickstart/share.

LOWRY, P. B.; CURTIS, A.; LOWRY, M. R. Building a taxonomy and nomenclature of collaborative writing to improve interdisciplinary research and practice. **The Journal of Business Communication (1973)**, [S.l.], v.41, n.1, p.66–99, 2004.

MARTINS, F.; LOPES, F.; QUELHAS, N.; ROCHA, R.; AZEVEDO, R.; TEIXEIRA, T. **Edição Colaborativa de Documentos**: que uso fazem os docentes da feup de ferramentas de edição colaborativa? Acessado em: 28/04/2019, https://paginas.fe.up.pt/ ee02254/Relatorio_EQ58B.pdf.

MONTANHEIRO, L. S.; CARVALHO, A. M. M.; RODRIGUES, J. A. Utilização de JSON Web Token na Autenticação de Usuários em APIs REST. **XIII Encontro Anual de Computação. Anais do XIII Encontro Anual de Computação EnAComp**, [S.l.], p.186–193, 2017.

OFFICE, S. **Microsoft Office**: suporte. Acessado em: 20/06/2019, https://support.office.com/pt-br/article/diferen%C3%A7as-entre-o-uso-de-um-documento-no-navegador-e-no-word-3e863ce3-e82c-4211-8f97-5b33c36c55f8.

OVERLEAF. **Overleaf**: site oficial. Acessado em: 08/06/2019, https://www.overleaf.com/about.

SERMANN, K. **Edição colaborativa em tempo real**. Acessado em: 16/06/2019, https://medium.com/tend%C3%AAncias-digitais/edi%C3%A7%C3%A3o-colaborativa-6ac2a89ce55d.

SEVERINO, A. **Metodologia do trabalho científico**. [S.l.]: Cortez Editora, 2017.

TORRES, P. L.; IRALA, E. A. F. Aprendizagem colaborativa: teoria e prática. **Complexidade: redes e conexões na produção do conhecimento. Curitiba: Senar**, [S.l.], p.61–93, 2014.

VSCODE. **Visual Studio Code**: site oficial. Acessado em: 18/11/2019, https://code.visualstudio.com/docs.

WIERSEMA, N. How Does Collaborative Learning Actually Work in a Classroom and How Do Students React to It? A Brief Reflection. , [S.l.], 2002.