

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

ERIC BERNARD PEREIRA MOURA BRASIL
SÉRGIO FELIPE REZENDE DO NASCIMENTO

**Sistema de Auxílio à Escrita e
Aprimoramento de Textos Utilizando
Large Language Model**

Prof. Filipe Braidão do Carmo, D.Sc.
Orientador

Nova Iguaçu, Dezembro de 2023

Sistema de Auxílio à Escrita e Aprimoramento de Textos Utilizando Large Language Model

Eric Bernard Pereira Moura Brasil
Sérgio Felipe Rezende do Nascimento

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

Eric Bernard Pereira Moura Brasil

Sérgio Felipe Rezende do Nascimento

Aprovado por:

Prof. Filipe Braidão do Carmo, D.Sc.

Prof. Leandro Guimarães Marques Alvim, D.Sc.

Prof. Bruno José Dembogurski, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Dezembro de 2023



DOCUMENTOS COMPROBATÓRIOS Nº 28574/2023 - CoordCGCC (12.28.01.00.00.98)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 28/12/2023 16:16)

BRUNO JOSE DEMBOGURSKI
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###249#4

(Assinado digitalmente em 27/12/2023 08:49)

FILIFE BRAIDA DO CARMO
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###295#4

(Assinado digitalmente em 27/12/2023 15:00)

LEANDRO GUIMARAES MARQUES ALVIM
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###008#2

(Assinado digitalmente em 27/12/2023 07:14)

SERGIO FELIPE REZENDE DO NASCIMENTO
DISCENTE
Matrícula: 2019#####6

(Assinado digitalmente em 27/12/2023 04:17)

ERIC BERNARD PEREIRA MOURA BRASIL
DISCENTE
Matrícula: 2019#####7

Visualize o documento original em <https://sipac.ufrrj.br/documentos/> informando seu número: 28574, ano: 2023, tipo: **DOCUMENTOS COMPROBATÓRIOS**, data de emissão: 26/12/2023 e o código de verificação: 33bae042b

Agradecimentos

Eric Bernard Pereira Moura Brasil

Primeiramente, gostaria de agradecer aos meus pais, Helem e Miguel, os meus maiores fãs. Todo o esforço que fizeram para que eu trilhasse meu caminho é, de longe, um dos maiores motivadores que tenho na minha vida. Obrigado por tudo, pelas lições, pelo carinho, mas especialmente por sempre acreditarem em mim, eu amo vocês.

Gostaria de agradecer também às minhas irmãs, Maiza e Evelyn, que sempre me deram muito apoio e estiveram ali por mim quando precisei. Obrigado por dividirem essa vida comigo e fazer dela mais especial.

Outra pessoa a qual preciso agradecer é a minha namorada Carla. Obrigado por estar sempre comigo, mesmo em períodos caóticos da minha vida, e me fazer sentir amado todos os dias. A vida é muito mais fácil e bela com você ao meu lado.

Ao meu parceiro nesse trabalho, Sérgio Rezende, gostaria de agradecer pela amizade durante todo esse curso. Todos os momentos que dividimos nesse período, seja andando quilômetros pintados de Power Rangers ou passando madrugadas fazendo trabalhos, estarão para sempre marcados em minha memória. Obrigado pela parceria e por tornar minha experiência mais especial.

Preciso agradecer especialmente aos meus amigos Caroline Pires, Gabriel Cappa, Jonatas Sennas, Sérgio Taveira e Yuri Menezes, que no momento mais difícil da minha vida me acolheram e ajudaram a me reerguer, sem vocês eu não seria nada.

Agradeço também ao meu amigo Phelipe Zache, que compartilhando a paixão

pelo Fluminense se tornou um dos meus melhores amigos. Obrigado por todos os conselhos e momentos que compartilhamos, te admiro demais.

Aos meus amigos, Rodrigo e Letícia, agradeço pelas inúmeras conversas no discord, seja me aconselhando, ouvindo meus desabafos ou apenas tornando a vida mais fácil me distraíndo e divertindo, principalmente no final dessa jornada.

Agradeço também aos meus amigos que estão comigo desde o ensino médio do CEFET, que mesmo depois de tanto tempo de formados, continuam sendo presentes e especiais na minha vida.

Às amigadas que fiz do grupo “Softawii”, meu muitíssimo obrigado por dividirem essa experiência comigo, compartilhando conhecimento e momentos. Aos meus amigos, Nicolas Magalhães e Yan Figueiredo, meu agradecimento especial, não só pelos tantos trabalhos que fizemos juntos mas também pela amizade e momentos externos à faculdade, eterno grupo 5 MVP.

Agradeço à minha psicóloga, Viviane Pamella, por todas as conversas e conselhos dados nesse ano em que estamos juntos. Você é fundamental no meu entendimento e crescimento pessoal.

Ao meu orientador, Filipe Braida, agradeço por toda a paciência e esforço na nossa orientação, que foi muito além do TCC. Felizmente pude fazer algumas matérias ministradas por você, o que só aumentou minha admiração pela pessoa e profissional que é.

Agradeço a todo o corpo docente do DCC pelos aprendizados e experiências, que foram essenciais nessa minha jornada.

Por fim, gostaria de agradecer e dedicar esse trabalho à minha irmã Maiara, que hoje não está aqui fisicamente mas que carrego comigo todos os dias. Obrigado por sempre acreditar em mim e fazer de tudo para me ajudar, mesmo que de longe. Espero que esteja orgulhosa, você é minha maior saudade.

Sérgio Felipe Rezende do Nascimento

Primeiramente, eu quero agradecer meus pais, Anne Christie e Sérgio Luiz, por terem me trazido até aqui. Todo o amor e carinho de vocês, além de investimento, que me fizeram chegar tão longe. Muito obrigado por serem pais tão bons e especialmente por sempre acreditarem em mim. Espero continuar dando muito orgulho para vocês.

Outra pessoa extremamente importante para mim que eu desejo agradecer é minha irmã Lisandra, que me ensinou muitas coisas ao longo da vida e sempre me ajudou em tudo que precisei. Obrigado por sempre estar ali por mim e me fazer sentir ter minha melhor amiga dentro de casa.

Também gostaria de agradecer a minha namorada Ana Luisa, por me ter me dado todo o suporte que preciso em todos os momentos. Agradeço por sempre me ajudar quando eu precisei, aguentar todos os meus nervosismos. Obrigado por sempre me fazer feliz e por todo amor que compartilhamos.

É impossível deixar de agradecer ao meu parceiro de trabalho Eric Bernard. O meu primeiro amigo na faculdade também é a pessoa que vai fechá-la comigo. Não tenho palavras para descrever o quão importante foi, para mim, te conhecer na faculdade. Obrigado por todas as experiências que tivemos juntos, desde o trote no primeiro dia, até o último dia para entrega desse trabalho.

Àqueles que compõem o grupo "Softawii", meu grupo de amigos excepcionais, meu sincero agradecimento. Obrigado por todos os momentos e ensinamentos compartilhados, vocês foram cruciais para minha jornada nesse curso. Nicolas Magalhães e Yan Figueiredo, em particular, merecem uma menção especial, vocês são pessoas incríveis que eu tive o prazer de conhecer na faculdade. Obrigado por todos os momentos dentro e fora da UFRRJ.

Não posso deixar de agradecer ao meu orientador, Filipe Braidá, que desde a primeira aula teve a minha mais sincera admiração. Obrigado por todos os aprendizados e projetos que foram desenvolvidos juntos, desde as maratonas, AtisLabs e discord. Muito obrigado também por aceitar o convite de orientador para esse trabalho. Agradeço também à professora Juliana Zamith por incentivar minha participação na

iniciação científica, uma experiência crucial para o meu desenvolvimento.

Por último, quero agradecer a todos os professores do DCC, cujo apoio e orientação foram fundamentais para minha jornada até aqui. Obrigado por todas as aulas e aprendizados.

RESUMO

Sistema de Auxílio à Escrita e Aprimoramento de Textos Utilizando Large Language Model

Eric Bernard Pereira Moura Brasil e Sérgio Felipe Rezende do Nascimento

Dezembro/2023

Orientador: Filipe Braida do Carmo, D.Sc.

Com o constante avanço no campo da inteligência artificial e a crescente relevância dos *Large Language Models*, a utilização dessas ferramentas em tarefas de processamento de linguagem natural torna-se de valor inestimável. Este estudo apresenta um sistema web que não apenas permite aos usuários redigir e armazenar documentos de maneira eficiente, mas também desempenha funções cruciais, como tradução, sumarização, formalização, entre outras, de forma ágil. Para isso, integramos o ChatGPT, que executa as tarefas necessárias. Esta abordagem elimina a necessidade de conhecimento prévio sobre tais aplicações, otimizando o processo criativo. Além disso, considerando a evolução significativa dos dispositivos móveis, o sistema destaca-se pela praticidade de uso nesses dispositivos, proporcionando, assim, uma ferramenta acessível e versátil.

ABSTRACT

Sistema de Auxílio à Escrita e Aprimoramento de Textos Utilizando Large Language
Model

Eric Bernard Pereira Moura Brasil and Sérgio Felipe Rezende do Nascimento

Dezembro/2023

Advisor: Filipe Braidão do Carmo, D.Sc.

With the constant advancement in the field of artificial intelligence and the growing relevance of large language models, the use of these tools in natural language processing tasks becomes invaluable. This study presents a web system that not only allows users to draft and store documents efficiently but also performs crucial functions such as translation, summarization, formalization, among others, in an agile manner. To achieve this, we integrated ChatGPT, which executes the necessary tasks. This approach eliminates the need for prior knowledge of such applications, optimizing the creative process. Furthermore, considering the significant evolution of mobile devices, the system stands out for its user-friendly nature on these devices, thus providing an accessible and versatile tool.

Lista de Figuras

Figura 2.1: Representação das arquiteturas da Recurrent Neural Network (RNN) e Long Short-term Memory (LSTM)	6
Figura 2.2: Arquitetura básica de um <i>Transformer</i>	8
Figura 2.3: Gráfico da função de perda em relação ao número de tokens do conjunto de dados (<i>dataset</i>).	9
Figura 2.4: As tendências dos números cumulativos de artigos no arXiv que contêm as palavras-chave <i>language model</i> (desde junho de 2018) e <i>large language model</i> (desde outubro de 2019), respectivamente. . .	11
Figura 2.5: Diagrama de Venn mostrando a divisão dos grupos de desafios dos LLMs.	13
Figura 3.1: Sugestões de correções no texto pelo Grammarly	17
Figura 3.2: Organização dos documentos de um usuário no Grammarly	18
Figura 3.3: Sugestões de correções no texto pela Clarice.ai	19
Figura 3.4: Organização dos documentos de um usuário na Clarice.ai	20
Figura 3.5: Diagrama de integração do sistema com o ChatGPT	22
Figura 3.6: Diagrama de integração do sistema com o ChatGPT Utilizando sistemas de cache e fila	23
Figura 3.7: Modelo Entidade-Relacionamento do sistema	28

Figura 4.1: Rotas declaradas no AdonisJS utilizando <i>authentication</i>	30
Figura 4.2: Estilização de elemento HyperText Markup Language (HTML) utilizando classes do Tailwind	31
Figura 4.3: Comparativo da exibição do menu nas versões <i>mobile</i> e <i>desktop</i> .	32
Figura 4.4: Editor de texto da aplicação utilizando o QuillJS	33
Figura 4.5: Arquitetura utilizada no sistema	34
Figura 4.6: Telas de cadastro e autenticação do sistema.	39
Figura 4.7: Tela de edição das informações do usuário	40
Figura 4.8: Tela de documentos do usuário na versão <i>mobile</i>	41
Figura 4.9: Tela de documentos do usuário na versão <i>desktop</i>	41
Figura 4.10: Tela de edição de documento (editor) na versão <i>mobile</i>	42
Figura 4.11: Menus laterais na tela de edição de documentos	43
Figura 4.12: Tela de edição de documento (editor) na versão <i>desktop</i>	43
Figura 4.13: Tela de edição de documentos com um exemplo da tarefa de tradução.	44
Figura 4.14: Tela de edição de documentos com um exemplo da tarefa de formalização.	45
Figura 4.15: Tela de edição de documentos com um exemplo da tarefa de correção ortográfica.	46

Lista de Códigos

4.1	Corpo da requisição enviada ao ChatGPT	36
4.2	Resposta recebida do ChatGPT	37
4.3	Campo <i>choices</i> extraído da resposta do ChatGPT	37

Lista de Abreviaturas e Siglas

AGI	Artificial General Intelligence
API	Application Programming Interface
CETIC	Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação
CSS	Cascading Style Sheets
EFL	English as a Foreign Language
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IA	Inteligência Artificial
INAF	Indicador Nacional de Alfabetismo Funcional
JS	JavaScript
LLM	Large Language Models
LSTM	Long Short-term Memory
ML	Machine Learning
MVC	Model-View-Controller
ORM	Object-Relational Mapping
PLN	Processamento de Linguagem Natural
RN	Redes Neurais

RNN Recurrent Neural Network

SGBD Sistema de Gerenciamento de Banco de Dados

WYSIWYG What You See Is What You Get

Sumário

Agradecimentos	ii
Resumo	vi
Abstract	vii
Lista de Figuras	viii
Lista de Códigos	x
Lista de Abreviaturas e Siglas	xi
1 Introdução	1
1.1 Objetivo	2
1.2 Principais Contribuições	2
1.3 Organização do Trabalho	3
2 Large Language Models	4
2.1 Introdução	5
2.2 Evolução	9
2.3 Aplicações	11

2.4	Desafios	12
3	Proposta	15
3.1	Motivação	15
3.2	Trabalhos Relacionados	17
3.3	Proposta	20
3.3.1	Arquitetura do Sistema	21
3.3.2	Utilização do LLM	23
3.3.3	Funcionalidades	26
3.3.4	Banco de Dados e Armazenamento	27
4	Implementação	29
4.1	Tecnologias Utilizadas	29
4.1.1	AdonisJS	30
4.1.2	Tailwind CSS	30
4.1.3	AlpineJS	31
4.1.4	QuillJS	32
4.1.5	PostgreSQL	33
4.2	Padrão MVC	34
4.2.1	View	35
4.2.2	Controller	35
4.2.3	Service	35
4.2.4	Model	37
4.3	Interfaces	38

4.3.1	Cadastro e Autenticação do Usuário	38
4.3.2	Edição de informações do Usuário	39
4.3.3	Documentos do Usuário	40
4.3.4	Editor de Documentos	42
5	Conclusão	47
5.1	Considerações finais	47
5.2	Limitações e trabalhos futuros	48
	Referências	50
A	Casos de Uso	52

Capítulo 1

Introdução

A habilidade de escrever é fundamental para a comunicação eficaz, o processo de aprendizado e a expressão de ideias complexas. No entanto, é uma realidade comum que muitas pessoas enfrentem dificuldades ao redigir textos de maneira clara, coerente e correta, especialmente em contextos acadêmicos ou profissionais (Rahimi e Zhang 2019). Tal desafio é ainda mais acentuado quando se considera dados alarmantes, como os fornecidos pelo Indicador Nacional de Alfabetismo Funcional (INAF), que indicam que apenas 12% da população brasileira possui proficiência na escrita¹.

Essa lacuna nas habilidades de escrita pode ter impactos significativos, limitando não apenas a capacidade de expressão individual, mas também afetando negativamente o desempenho em diversas áreas da vida. Em ambientes acadêmicos, a produção de textos de qualidade é crucial para o sucesso dos estudantes, enquanto profissionais de diferentes setores necessitam comunicar ideias complexas de maneira clara e persuasiva. Nesse cenário, a busca por ferramentas que auxiliem no aprimoramento da escrita torna-se essencial.

Nesse contexto, a Inteligência Artificial (IA) se destaca como uma aliada valiosa, oferecendo soluções para diversos desafios relacionados ao Processamento de Linguagem Natural (PLN), incluindo tradução automática (Brants et al. 2007), geração de

¹<https://alfabetismofuncional.org.br/alfabetismo-no-brasil/>

texto (Li et al. 2022) e sumarização de informações (Zhang et al. 2023). A ascensão dessas soluções baseia-se em modelos avançados de IA, conhecidos como Large Language Models (LLM). Esses modelos são treinados em enormes conjuntos de dados textuais coletados da internet, permitindo-lhes compreender e gerar textos em uma variedade de domínios e temas.

Embora essas ferramentas possam ser úteis, elas também podem apresentar dificuldades e limitações, pois converter as demandas dos usuários para a IA pode ser um processo complicado e, às vezes, repetitivo. Além disso, o desconhecimento e o uso impróprio dessas ferramentas podem acarretar em graves problemas, como por exemplo, a chance de gerar respostas tendenciosas ou erradas por parte da ferramenta, devido a falhas no *dataset* usado para o seu treinamento (Kalla et al. 2023).

1.1 Objetivo

Neste trabalho, propomos o desenvolvimento de um sistema web *Mobile First* que integra o ChatGPT, um dos LLMs mais avançados da atualidade, para auxiliar os usuários na escrita e aprimoramento de textos. O sistema permite que os usuários redijam e armazenem documentos de maneira eficiente, e também realizem funções cruciais, como tradução, sumarização, formalização, entre outras, de forma ágil e interativa. O sistema visa otimizar o processo criativo, eliminando a necessidade de conhecimento prévio sobre as aplicações de IA, e proporcionando uma ferramenta acessível e versátil, especialmente para dispositivos móveis.

1.2 Principais Contribuições

- Revisão Bibliográfica e Discussão sobre LLM.

Este estudo propõe uma revisão bibliográfica abrangente e uma análise crítica das LLM. O intuito é consolidar esta pesquisa como uma fonte no âmbito das LLM na língua portuguesa. Ao explorar a origem, evolução, aplicações atuais e desafios enfrentados por essas linguagens, busca-se promover uma discussão

que contribua para a compreensão do tema.

- Desenvolvimento de um Sistema Web para Assistente de Escrita com LLM.

Além da revisão bibliográfica, este trabalho propõe o desenvolvimento de um sistema web para assistência de escrita utilizando as LLM. Esta iniciativa visa ser uma contribuição prática para projetos que buscam integrar LLM em suas aplicações.

1.3 Organização do Trabalho

O trabalho está organizado da seguinte forma:

- No capítulo 2, apresentamos uma revisão bibliográfica sobre os grandes modelos de linguagem, sua origem, evolução, capacidade e desafios.
- No capítulo 3, descrevemos a motivação por trás do proposta, trabalhos relacionados e a proposta da aplicação, tratando da arquitetura, integração com o ChatGPT, modelagem do banco de dados e funcionalidades.
- No capítulo 4, detalhamos a implementação do sistema, as tecnologias utilizadas, o padrão de arquitetura do código e interfaces desenvolvidas.
- No capítulo 5, concluímos o trabalho, apresentando as considerações finais, as limitações e os trabalhos futuros.

Capítulo 2

Large Language Models

O Processamento de Linguagem Natural (PLN) representa um desafio de longa data no campo da IA, dando origem a diversas aplicações de grande utilidade, tais como tradução de idiomas e geração de texto. Devido à natureza sequencial dos dados textuais, a criação de modelos eficazes para abordar esse tipo de problema tem sido historicamente desafiadora. Nesse cenário, emergiram os LLMs, que são modelos de IA projetados para lidar com problemas complexos de PLN, aproveitando vastas quantidades de dados (Zhao et al. 2023).

Ao longo dos últimos anos, diversos modelos notáveis foram desenvolvidos utilizando essa tecnologia, incluindo os *chatbots* ChatGPT¹ da OpenAI, o Bard² da Google e o Bing³ da Microsoft. Este estudo emprega o ChatGPT como ferramenta central para abordar as tarefas relacionadas ao PLN. Nesse contexto, este capítulo visa proporcionar uma contextualização sobre a origem e evolução desses grandes modelos, conforme apresentado na seção 2.1. A seção 2.2 aborda a evolução desses modelos e como cresceram ao longo dos anos. Já a seção 2.3 descreve algumas aplicações onde os LLMs são utilizados. Por fim, a seção 2.4 discute os desafios enfrentados por esses modelos.

¹<<https://chat.openai.com/>>

²<<https://bard.google.com/>>

³<<https://www.bing.com/?/ai>>

2.1 Introdução

O processamento de linguagem natural (PLN) é uma variedade de técnicas computacionais motivadas por teorias para a análise automática e representação da linguagem humana (Cambria e White 2014). Dado o contexto contemporâneo, caracterizado pelo crescimento exponencial da quantidade de dados textuais, impulsionado pela ascensão das redes sociais, a relevância do PLN torna-se ainda mais pronunciada. Diversas aplicações evidenciam sua presença significativa na vida cotidiana, abrangendo tarefas como tradução (Brants et al. 2007), geração de texto (Li et al. 2022) e até mesmo sumarização (Zhang et al. 2023), destacando o alcance abrangente e a influência do PLN na experiência das pessoas.

Ao longo do tempo diversos modelos do campo de inteligência artificial foram propostos com o intuito de realizar as tarefas de PLN. Entre as soluções colocadas como estado da arte, destacam-se as baseadas em redes neurais, notadamente as RNN e as LSTM (Sutskever e Le 2014; Cho et al. 2014; Hochreiter e Schmidhuber 1997). Considerando que o texto é um dado de natureza sequencial, onde a ordem das palavras desempenha um papel crucial, torna-se essencial desenvolver uma estratégia que permita ao modelo capturar essa sequencialidade.

Nesse contexto, as RNN implementam um tipo de *loop*, no qual cada saída gerada pela entrada sequencial é retroalimentada na rede para a próxima entrada (Jain e Medsker 1999). Contudo, uma limitação evidente dessa abordagem emerge em sequências extensas, uma vez que informações relevantes podem ser perdidas ao longo do processo. Essa perda de informação é decorrente do fato de que, a cada iteração, a saída utilizada para realimentar a rede passa por uma série de cálculos que a transformam de maneira incremental (Schmidt 2019). Consequentemente, em sequências longas, o contexto pode gradualmente se deteriorar ao longo das iterações, ocasionando o fenômeno conhecido como “memória curta”, no qual o modelo tende a priorizar mais fortemente as informações próximas umas das outras na sequência.

As LSTM foram concebidas com o propósito de mitigar o desafio da memória curta

inerente às RNN (Schmidt 2019). Essa abordagem introduz um novo componente no modelo, denominada memória de longo prazo, que passa por processamentos menos invasivos, permitindo a retenção de informações relevantes ao longo de sequências mais extensas (Sherstinsky 2020). Apesar da melhoria no desempenho com o problema da memória, em sequências ainda mais extensas, a possibilidade de perda de informações persistia.

Além da perda de memória, é importante destacar que os algoritmos associados a esses modelos possuem uma característica de processamento lento e, devido à falta de paralelização, não exploram plenamente as capacidades das Graphics Processing Unit (GPU) contemporâneas. A figura 2.1 ilustra como ambas as arquiteturas empregam o modelo de *loop*, evidenciando a dependência entre os passos sucessivos, nos quais cada passo requer a conclusão do anterior para sua execução.

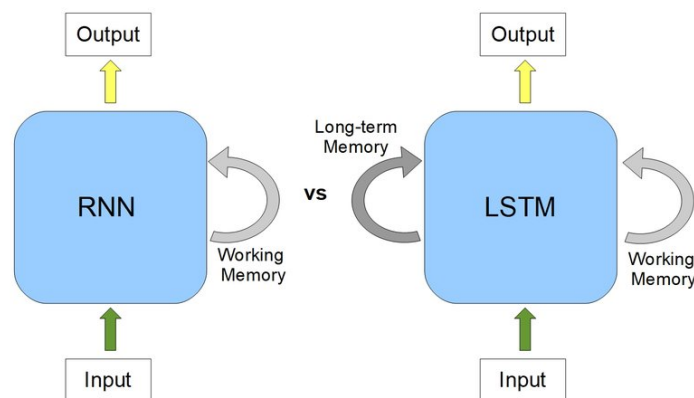


Figura 2.1: Representação das arquiteturas da RNN e LSTM

Fonte: (Yasrab e Pound 2020)

Diante desse contexto, com o intuito de abordar eficazmente o desafio da memória de curto prazo e otimizar a utilização da arquitetura computacional, analistas do Google conceberam um modelo denominado *Transformers* (Vaswani et al. 2017). Este modelo foi desenvolvido com o propósito de mitigar as limitações observadas em abordagens anteriores, permitindo, ao mesmo tempo, a implementação eficiente de paralelização.

O algoritmo *Transformers* emergiu como a base para muitos dos LLMs, oferecendo

a capacidade de processar volumes massivos de dados devido à sua notável velocidade de treinamento. Além disso, conseguiu superar desafios associados à natureza sequencial dos dados, representando uma solução eficaz para a modelagem de tarefas complexas no campo do PLN.

Um *Transformer* é uma arquitetura de Redes Neurais (RN) voltado para o problema de dados sequenciais que revolucionou a área de processamento de linguagem natural (Vaswani et al. 2017). Uma rede neural é um algoritmo de Machine Learning (ML) que têm o objetivo de simular o cérebro humano, e a forma que ocorre o aprendizado. Sendo assim, ocorre um treinamento prévio para desenvolver um conhecimento sobre algum assunto.

Os *transformers* utilizam um modelo de RN chamado *encoder* e *decoder*, onde as informações são recebidas, codificadas e depois decodificadas gerando a saída. A base da ideia desse modelo é a utilização da atenção, que é uma estrutura que permite a análise de diferentes partes de um contexto para gerar a previsão. Além disso, também adiciona um aprendizado mais rápido para os modelos (Vaswani et al. 2017).

Além da divisão por *encoder* e *decoder*, o modelo se divide em várias camadas. A camada de *embedding* transforma os dados para serem entendidos pela rede. Utilizam de métodos para transformar, por exemplo, dados textuais em vetores, mantendo o contexto das informações. A camada de atenção é responsável por dar mais contexto aos dados da camada de *embedding*. Relacionando os dados entre si e dando ênfase aos mais importantes.

A camada de Rede Neural Totalmente Conectada que é responsável pelo aprendizado do modelo. Recebe informações das camadas de atenção e retorna valores processados. É possível analisar a relação entre essas camadas na figura 2.2, percebendo que o *encoder* está representado na parte esquerda e o *decoder* na direita (Vaswani et al. 2017).

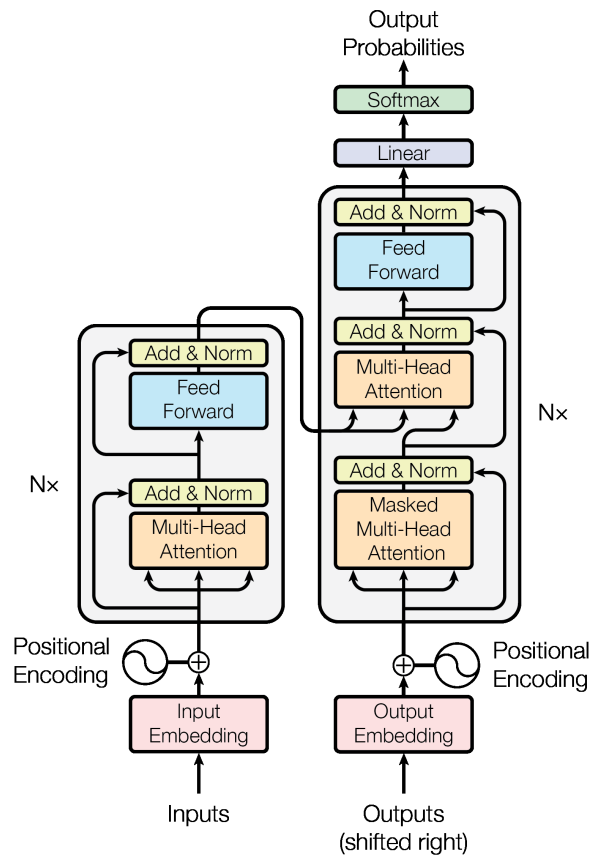


Figura 2.2: Arquitetura básica de um *Transformer*

Fonte: (Vaswani et al. 2017)

Assim, foi desenvolvido o Generative Pre-trained Transformer (GPT), um algoritmo que utiliza a arquitetura dos *Transformers* e aplica conceitos de inteligência artificial generativa. Um dos pilares fundamentais desse modelo é sua natureza pré-treinada (Radford et al. 2019; Brown et al. 2020). Isso implica que o modelo passa por um treinamento inicial em um conjunto diversificado de dados antes de ser utilizado em tarefas específicas. Essa pré-treinamento permite que o modelo adquira uma compreensão abrangente da linguagem e contextos diversos, contribuindo para seu desempenho notável ao lidar com uma variedade de tarefas linguísticas.

2.2 Evolução

Os avanços tecnológicos e a crescente capacidade computacional das GPUs também desempenharam um papel fundamental no desenvolvimento e na evolução dos LLMs. Juntamente a isso, estudos mostraram que a ampliação do modelo, frequentemente, resulta em uma capacidade de modelo aprimorada em tarefas subsequentes (Zhao et al. 2023). Isso pode ser mostrado na figura 2.3, que mostra a função de perda diminuindo em relação ao tamanho do conjunto de dados (*dataset*) e o número de parâmetros.

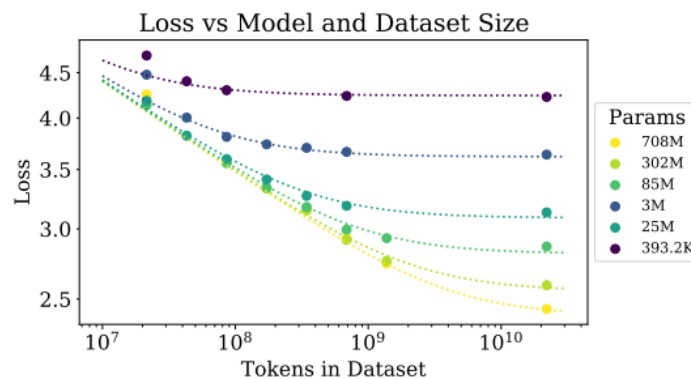


Figura 2.3: Gráfico da função de perda em relação ao número de tokens do conjunto de dados (*dataset*).

Fonte: (Kaplan et al. 2020)

Sendo assim, a medida que a capacidade de processamento aumentou, os pesquisadores foram capazes de projetar modelos mais complexos e sofisticados, impulsionando significativamente o campo da inteligência artificial. O GPT 2.0, por exemplo, com seus 1.5 bilhão de parâmetros, já era considerado um marco significativo em sua capacidade de gerar texto coerente e contextualmente relevante (Radford et al. 2019). No entanto, o salto para o GPT 3.0, com seus impressionantes 175 bilhões de parâmetros, representou uma grande expansão. Essa quantidade massiva de parâmetros permitiu ao modelo capturar nuances mais sutis na linguagem e contextualizar informações de maneira mais eficaz em relação ao seu antecessor, resultando em uma melhoria notável na qualidade de suas respostas e na compreensão de contextos

complexos (Brown et al. 2020).

Com a versão 4.0 utilizando um número ainda maior de parâmetros, a capacidade desses modelos de linguagem para lidar com uma variedade ainda maior de tarefas linguísticas e compreender contextos extremamente complexos ficou ainda maior. Alguns estudos destacam esta versão específica do GPT como a aproximação mais significativa que os modelos de IA alcançaram até agora em direção à Artificial General Intelligence (AGI) (Bubeck et al. 2023). O termo AGI refere-se a sistemas que exibem amplas capacidades de inteligência, englobando raciocínio, planejamento e a habilidade de aprender com a experiência, com essas aptidões equiparando-se ou ultrapassando o desempenho humano (Bubeck et al. 2023). No entanto, vale ressaltar que o aumento na quantidade de parâmetros não é o único fator determinante para o desempenho desses modelos. O refinamento de algoritmos, a diversidade e a qualidade dos dados de treinamento também desempenham papéis cruciais nesse avanço contínuo.

Com o progresso notável nessa área, diversos estudos têm sido conduzidos em várias disciplinas relacionadas aos LLMs. A figura 2.4 ilustra como, ao longo dos últimos anos, houve um expressivo aumento nas publicações científicas dedicadas a pesquisas nesse campo em expansão. Além disso, também mostra como marcos importantes como o ChatGPT também influenciam no número de pesquisas (Zhao et al. 2023).

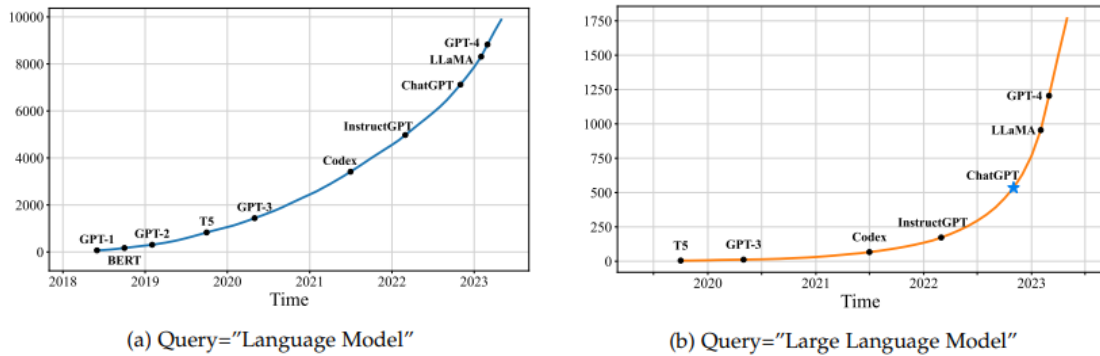


Figura 2.4: As tendências dos números cumulativos de artigos no arXiv que contêm as palavras-chave *language model* (desde junho de 2018) e *large language model* (desde outubro de 2019), respectivamente.

Fonte: (Zhao et al. 2023)

2.3 Aplicações

Devido a esses avanços, esses modelos vêm sendo usados em diversas áreas e para uma ampla variedade de tarefas. Suas capacidades abrangem desde a tradução automática (Brants et al. 2007) e geração de texto (Li et al. 2022) até a sumarização de conteúdos extensos (Zhang et al. 2023). Além disso, esses modelos têm se destacado em aplicações como *chatbots* inteligentes (Thoppilan et al. 2022) e análise de sentimento em textos (Araci 2019).

No âmbito da pesquisa científica, esses modelos de linguagem têm sido empregados para análise de grandes conjuntos de dados textuais, auxiliando na extração de informações relevantes e na formulação de hipóteses em diversas disciplinas (Blanco-González et al. 2023; Sun et al. 2023). Na área de saúde, esses modelos podem contribuir para o processamento de registros médicos, a interpretação de exames e a identificação de padrões em vastas bases de dados clínicos (Garg et al. 2023).

A influência desses modelos também se estende à educação (Kasneji et al. 2023), promovendo avanços notáveis. Ao serem aplicados como assistentes para escrita, esses modelos desempenham um papel essencial ao realizar correções gramaticais e

aprimorar a qualidade textual dos usuários. Sua capacidade de realizar sumarizações eficientes facilita a leitura de textos complexos, promovendo uma compreensão mais rápida e acessível para os estudantes. Além disso, os modelos de linguagem são recursos valiosos para professores, auxiliando na criação de questões para avaliações e contribuindo para o desenvolvimento de materiais educacionais mais refinados e eficazes.

Além disso, em setores como finanças (Araci 2019) e jurídico (Yu e Schilder 2022), esses modelos têm se mostrado valiosos na automação de tarefas rotineiras, análise de tendências de mercado, elaboração de relatórios e até mesmo na redação de documentos legais. A versatilidade desses modelos em lidar com nuances linguísticas e contextos específicos tem promovido avanços significativos em diversas frentes profissionais e acadêmicas, consolidando sua posição como uma ferramenta impactante e multifuncional na inteligência artificial.

2.4 Desafios

Apesar da grande evolução na área dos LLMs, muitos desafios permanecem existindo. No artigo de Kaddour et al. 2023, os autores classificam os desafios em três categorias: design, comportamento e ciência. Na figura 2.5 mostra como esses grupos são colocados em um Diagrama de Venn.

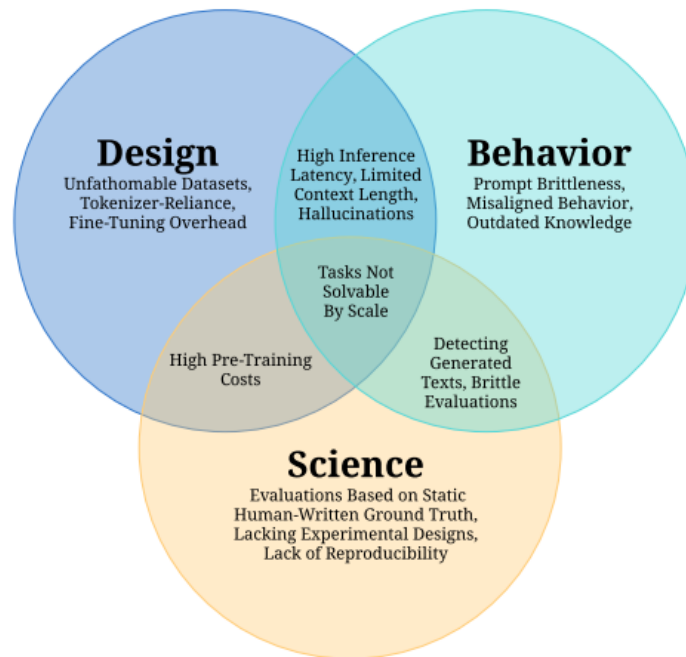


Figura 2.5: Diagrama de Venn mostrando a divisão dos grupos de desafios dos LLMs.

Fonte: (Kaddour et al. 2023)

A categoria de design envolve as decisões sobre os dados, arquitetura e algoritmos usados na construção do modelo (Kaddour et al. 2023). Nessa categoria se encontram desafios como garantir a qualidade e diversidade dos dados que serão usados para fazer o treinamento do modelo. A dependência de um *tokenizador*, que são os algoritmos que farão a transformação das bases em *tokens* a serem utilizados pelo modelo. Também existe o grande custo de treinamento, visto que os modelos são extremamente complexos e grandes, o custo computacional se torna igualmente grandioso. Além disso, também existem os problemas de memória, que também era enfrentado pelos algoritmos anteriores. Ou seja, existe um tamanho limite de contexto que o modelo consegue levar em consideração.

Já a categoria de comportamento abrange as questões sobre como os modelos se comportam quando interagem com os usuários e geram textos (Kaddour et al. 2023). Nesse grupo existem desafios como a dificuldade em usar o *prompt*, recebendo respostas muito diferentes com pequenas mudanças nas entradas. As alucinações, que ocorrem quando o modelo responde textos falsos ou completamente incoerentes com

o contexto dado. Também existe a questão do comportamento tóxico e inesperado, quando o modelo pega exemplos ruins dos dados e replica com o usuário. Por fim, um desafio também comum é manter o modelo atualizado, visto que ocorre um treinamento prévio, muitos modelos continuam com conhecimentos antigos.

Por fim, a categoria de ciência diz respeito aos desafios relacionados à compreensão científica dos LLMs, avaliando a qualidade e impacto (Kaddour et al. 2023). Nesse grupo existem os problemas de como avaliar adequadamente o modelo, visto que dificilmente medidas matemáticas conseguem capturar a capacidade e a diversidade dos modelos. Além disso, existem tarefas que não conseguem ser resolvidas apenas aumentando o número de dados, devido à complexidade de conhecimento necessária, um exemplo disso pode ser a área de matemática.

Capítulo 3

Proposta

Neste capítulo será descrita a proposta da aplicação, contextualizando a motivação para seu desenvolvimento e trabalhos relacionados. Na seção 3.1 será discutido o cenário atual da relação da população brasileira com a *internet*, incluindo o uso de IA e suas problemáticas. Em sequência, na seção 3.2 serão apresentados trabalhos que estão relacionados a este. Por fim, na seção 3.3 é apresentada a proposta, descrevendo sua arquitetura, modelagem do banco e funcionalidades.

3.1 Motivação

O avanço contínuo da tecnologia e a expansão do acesso à internet têm impulsionado significativamente o número de usuários conectados no Brasil. De acordo com dados da pesquisa TIC Domicílios, realizada pelo Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação (CETIC) em 2022, cerca de 80% dos domicílios brasileiros agora têm acesso à internet, representando aproximadamente 60 milhões de residências¹. Na esfera individual, a pesquisa revelou que 81% dos brasileiros, o equivalente a cerca de 149 milhões de pessoas, incorporam o uso da internet em suas atividades diárias. É relevante notar que 62% desse contingente utiliza exclusivamente dispositivos móveis, como telefones, para acessar a rede.

¹<https://cetic.br/media/docs/publicacoes/2/20230825143348/resumo_executivo_tic_domicilios_2022.pdf>

Contudo, de acordo com dados do INAF, constatou-se em 2018 que apenas 12% da população detinha proficiência em língua portuguesa ². No âmbito acadêmico, a professora Valeria de Marco destaca as dificuldades enfrentadas na redação de textos acadêmicos, especialmente nos trabalhos de conclusão de curso ³.

Nesse contexto, é interessante notar que o aumento no emprego de IA para facilitar as atividades cotidianas tornou-se uma notável tendência com os avanços tecnológicos. A introdução dos *Transformers* (Vaswani et al. 2017) impulsionou o desenvolvimento de pesquisas no campo, possibilitando a criação dos LLMs.

Nos últimos anos, aplicações que utilizam LLM, como o ChatGPT⁴ da OpenAI e o Bard⁵ da Google democratizaram o acesso a essas ferramentas, permitindo que mesmo aqueles com menor conhecimento tecnológico as incorporassem em suas rotinas para otimizar e simplificar suas tarefas diárias. No Brasil, de acordo com um estudo realizado pela Ilumeo, 67% da população já fez uso de algum assistente virtual⁶, mercado esse que contempla diversos aplicativos e *chatbots* com recursos de IA. Esse fenômeno reflete não apenas a expansão do uso de IA, mas também a capacidade de tornar essa tecnologia mais acessível e amigável para uma variedade mais ampla de usuários.

No entanto, o uso dessas ferramentas ainda pode ser desafiador e pouco prático, pois traduzir as necessidades dos usuários para a IA pode ser uma tarefa complexa e, em alguns contextos, repetitiva. Além disso, a falta de compreensão e o uso inadequado dessas ferramentas podem resultar em sérias consequências com, por exemplo, a possibilidade de geração de respostas enviesadas ou até mesmo incorretas por parte da ferramenta, devido a inconsistências no *dataset* utilizado no seu treinamento (Kalla et al. 2023). Essas questões destacam a importância não apenas de promover o uso consciente da IA, mas também de buscar soluções que simplifiquem e tornem mais acessível essa tecnologia. Isso é relevante principalmente em

²<<https://alfabetismofuncional.org.br/alfabetismo-no-brasil/>>

³<<https://jornal.usp.br/universidade/disciplina-atende-dificuldades-de-alunos-com-textos-academicos/>>

⁴<<https://chat.openai.com/>>

⁵<<https://bard.google.com/>>

⁶<<https://oglobo.globo.com/patrocinado/dino/noticia/2023/05/ia-maioria-dos-brasileiros-ja-utiliza-assistentes-virtuais.ghtml>>

tarefas fundamentais, como tradução, geração de texto, sumarização de informações e correção ortográfica.

3.2 Trabalhos Relacionados

No contexto de aplicações que auxiliam a escrita, uma das principais e mais consolidada no mercado é o Grammarly⁷, muito popular principalmente nos Estados Unidos. A aplicação verifica em tempo real a gramática, ortografia, o estilo e o tom dos textos, como demonstrado na figura 3.1. Além disso, a aplicação possui diversas outras funcionalidades, como detecção de plágios ou parafraseamento, sendo todas essas tarefas executadas com o auxílio de IA.

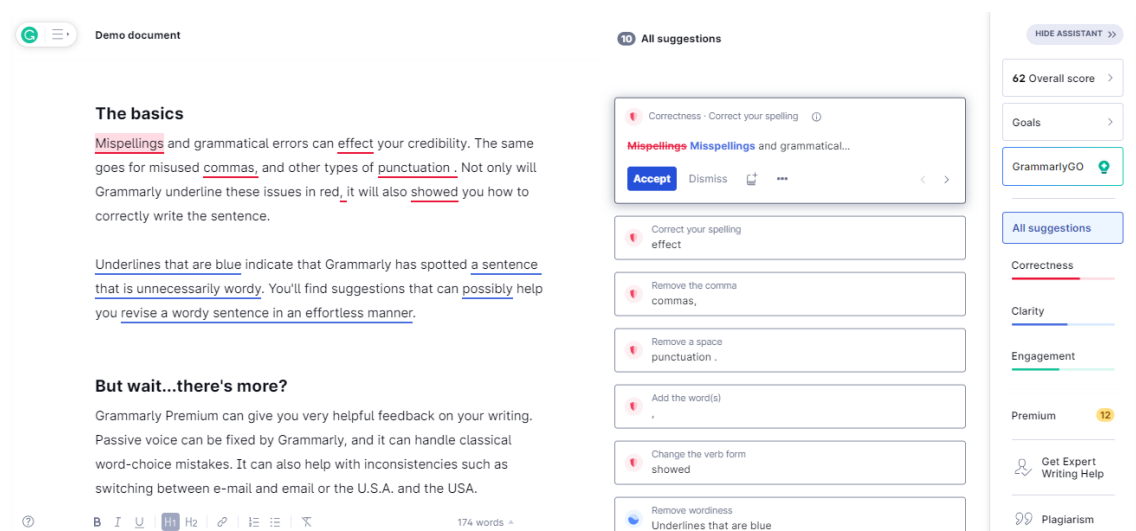


Figura 3.1: Sugestões de correções no texto pelo Grammarly

Na figura 3.2 é possível visualizar a organização de documentos do Grammarly. O sistema permite que o usuário crie pastas e categorias para armazenar seus documentos, de maneira parecida com a organização de arquivos de um computador, o que dá maior praticidade na hora de localizar os textos produzidos.

⁷<https://www.grammarly.com/>

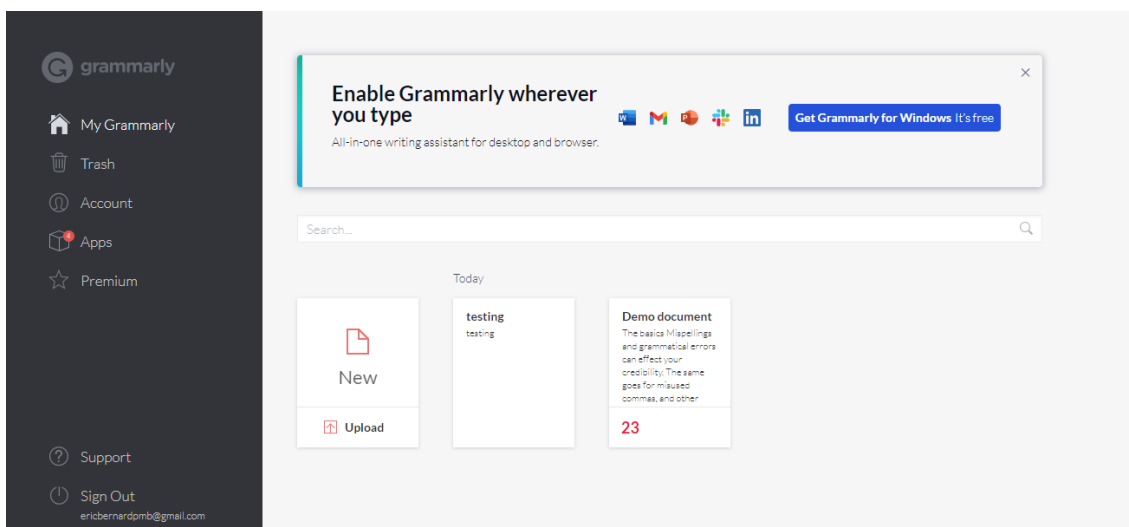


Figura 3.2: Organização dos documentos de um usuário no Grammarly

Diversos estudos têm buscado analisar o impacto do Grammarly no campo acadêmico. A utilização da aplicação por estudantes de inglês como língua estrangeira, do inglês English as a Foreign Language (EFL), no auxílio à escrita de artigos, dissertações ou relatórios, tem se provado de grande valor, principalmente por sua fácil utilização e explicação detalhada do motivo de suas sugestões, auxiliando no aprendizado da linguagem (Armanda et al. 2022). O estudo conclui que o Grammarly é uma ferramenta útil para auxiliar os estudantes, mas não substitui o papel do professor na avaliação e no *feedback* da escrita.

Apesar de ser utilizado por mais de 70 mil equipes⁸, uma das grandes limitações da aplicação são os dispositivos móveis. O Grammarly não possui uma versão *web mobile*, e a aplicação disponível faz parte da integração com o teclado do dispositivo móvel, o que pode não ser muito intuitivo e às vezes complicado de se utilizar. Esses fatores podem explicar por que o Grammarly é mais usado em dispositivos *desktop* (Armanda et al. 2022), onde os usuários podem aproveitar melhor os seus recursos e funcionalidades. Além disso, o Grammarly não possui suporte ao português, o que dificulta a sua utilização por parte de usuários que não possuam fluência em inglês.

⁸<https://www.grammarly.com/business>

Outra aplicação que se destaca no ramo é a Clarice.ai, uma alternativa brasileira de assistente de escrita textual. Assim como o Grammarly, a Clarice.ai incorpora funcionalidades essenciais ao auxílio à escrita, como a verificação em tempo real de gramática, ortografia, estilo e tom, proporcionando sugestões precisas para aprimorar a clareza e coesão dos textos, conforme exemplificado na figura 3.3. Além disso, a aplicação vai além, oferecendo recursos adicionais como detecção de plágio e parafraseamento, ampliando suas capacidades além das correções gramaticais convencionais.

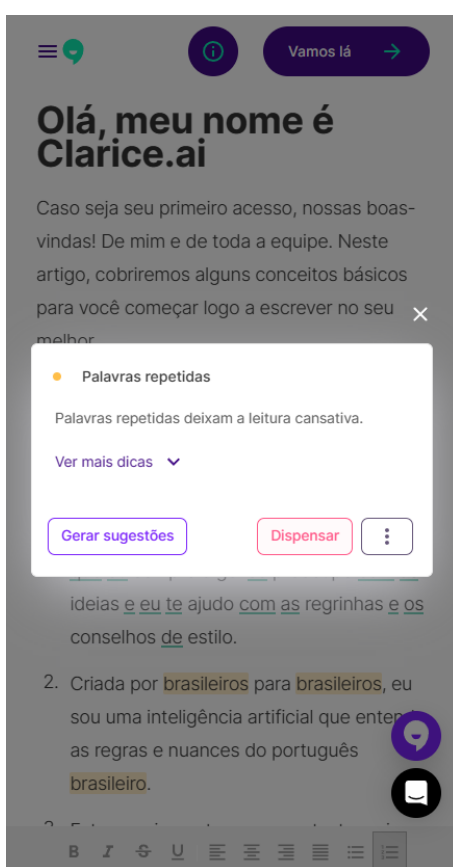


Figura 3.3: Sugestões de correções no texto pela Clarice.ai

A organização de documentos, evidenciada na figura 3.4, destaca-se como um ponto forte da Clarice.ai. Semelhante ao Grammarly, o sistema permite que os usuários organizem seus textos por meio de projetos, facilitando a localização e gestão eficiente de documentos, proporcionando uma experiência de usuário mais fluida.

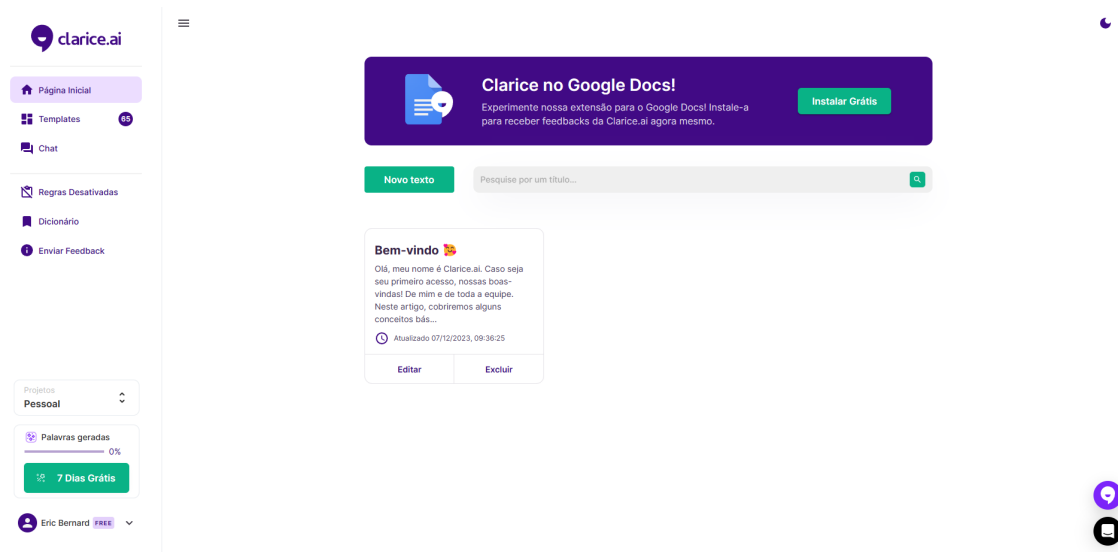


Figura 3.4: Organização dos documentos de um usuário na Clarice.ai

Um diferencial da Clarice.ai é sua abordagem em dispositivos móveis. Ao contrário do Grammarly, a Clarice.ai oferece uma versão web *mobile* acessível diretamente pelo navegador, demonstrado anteriormente na figura 3.3, proporcionando maior praticidade e usabilidade em dispositivos móveis. Essa versatilidade pode ser crucial para usuários que buscam aprimorar suas habilidades de escrita em diferentes contextos e ambientes.

Além disso, a Clarice.ai se destaca por oferecer suporte ao português, atendendo às necessidades de usuários que não possuem fluência em inglês. Em contrapartida, o treinamento do modelo em português limita a sua utilização apenas a usuários fluentes na língua portuguesa, de maneira semelhante à relação do Grammarly com a língua inglesa, citada anteriormente.

3.3 Proposta

Considerando o contexto supracitado, a proposta do trabalho é desenvolver uma aplicação web voltada para dispositivos móveis e com suporte a diversas línguas, tentando suprir a carência das aplicações já existentes. Nessa aplicação, os usuários

podem redigir e armazenar documentos de maneira prática, aproveitando a assistência do ChatGPT para realizar tarefas de PLN de forma simples, eliminando a necessidade de conhecimento prévio sobre esse tipo de aplicação e agilizando o processo criativo.

A subseção 3.3.1 abordará inicialmente a arquitetura do sistema, explorando pontos cruciais para o processo de tomada de decisões. Na subseção 3.3.2, será discutida a motivação por trás da escolha do ChatGPT e detalhado o processo de integração com o sistema. Já a subseção 3.3.3 descreverá as funcionalidades do sistema proposto, apresentando também os casos de uso associados. Por fim, a modelagem do banco de dados da aplicação, incluindo os relacionamentos entre as entidades, será discutida na subseção 3.3.4.

3.3.1 Arquitetura do Sistema

A aplicação segue um padrão monolítico, que reúne todas as funcionalidades em um único sistema. Esse padrão foi escolhido porque a base da aplicação possui funcionalidades limitadas e a maior complexidade está no processamento dos textos. A aplicação é composta por um servidor web e um banco de dados. O servidor web é o único ponto de contato dos clientes, que se comunicam com ele por meio das interfaces gráficas. Ele também gerencia o fluxo de dados dos usuários e documentos, e se conecta diretamente ao banco de dados, onde os dados dos usuários e dos documentos criados na aplicação são armazenados.

Para realizar as tarefas nos documentos, a aplicação utiliza o ChatGPT por meio de uma Application Programming Interface (API). Essa abordagem elimina a necessidade de desenvolver, treinar e manter um modelo próprio, o que reduz custos e complexidade. Os motivos para a escolha do ChatGPT e o processo de integração são descritos posteriormente na seção 3.3.2. A figura 3.5 ilustra a integração entre os componentes da aplicação.

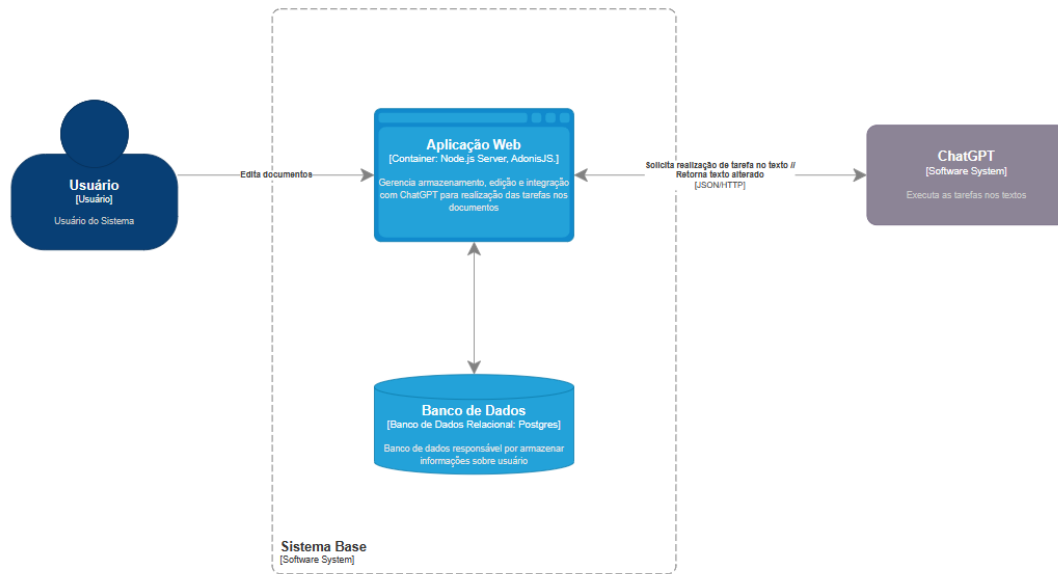


Figura 3.5: Diagrama de integração do sistema com o ChatGPT

Considerando a escalabilidade e melhoria de desempenho da aplicação, algumas técnicas podem ser aplicadas para otimizar a entrega de resultados dos processamentos dos textos, uma vez que a integração é feita via API. Uma delas é o uso de cache para armazenar resultados previamente processados pelo ChatGPT. Dessa forma, quando uma requisição similar é feita, com poucas mudanças em relação ao texto anteriormente processado, a aplicação pode verificar se a resposta já está em cache. Essa abordagem evitaria a necessidade de chamar o modelo novamente em caso de uso repetitivo desnecessário da funcionalidade por parte do usuário. Isso reduz significativamente o tempo de resposta e minimiza a carga no servidor, além de minimizar o custo para o usuário.

Outra técnica é a utilização de uma fila de mensagens para controlar o fluxo de solicitações ao ChatGPT. O processamento dos documentos de maneira fragmentada, por parágrafos, por exemplo, junto de uma estrutura de fila permitiria um controle mais eficiente do envio de solicitações à API. Essa melhoria na arquitetura é ilustrada na figura 3.6.

Ao implementar essas soluções em conjunto, a aplicação ganha em escalabilidade e

desempenho. Textos podem ser quebrados em fragmentos menores, distribuídos para processamento assíncrono por meio da fila, e os resultados podem ser armazenados em cache para otimizar consultas futuras. Dessa forma, a aplicação se torna capaz de lidar com um maior volume de requisições, oferecendo uma experiência mais responsiva aos usuários e garantindo eficiência no uso dos recursos computacionais disponíveis.

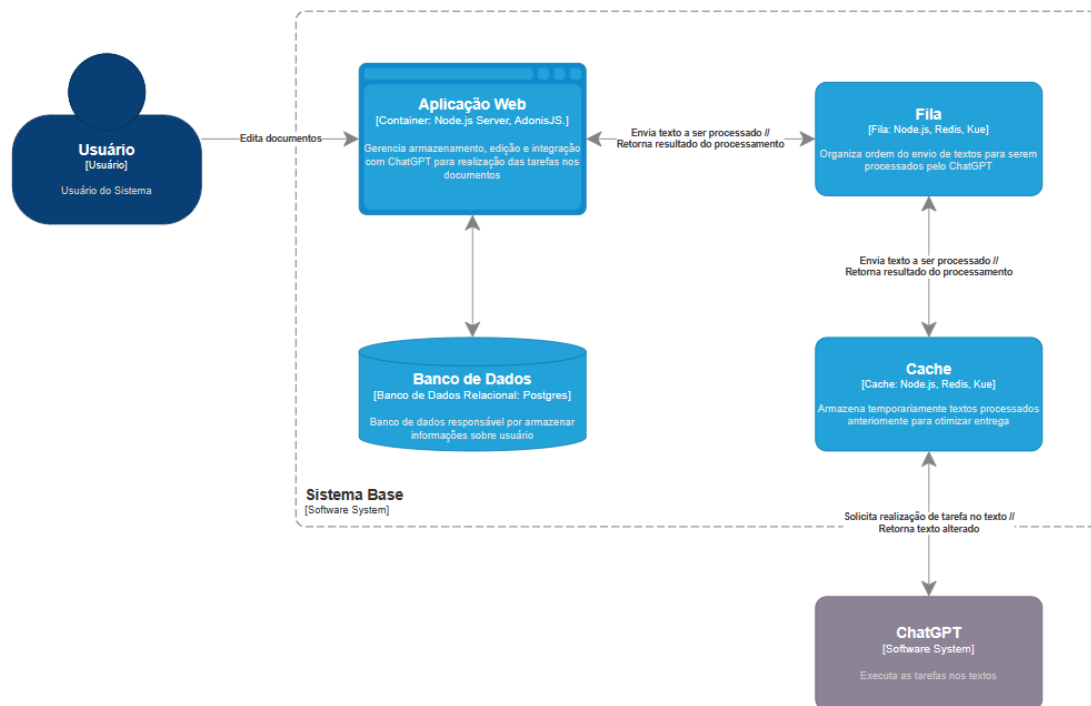


Figura 3.6: Diagrama de integração do sistema com o ChatGPT Utilizando sistemas de cache e fila

3.3.2 Utilização do LLM

O ChatGPT é um dos modelos de LLM mais consolidados no mercado, tendo adesão recorde de 100 milhões de usuários em apenas dois meses após o seu lançamento⁹. Sua alta popularidade impulsionou o uso da ferramenta em diversas áreas, o que conseqüentemente gerou diversos estudos acerca de seu uso. O artigo

⁹<<https://forbes.com.br/forbes-tech/2023/02/chatgpt-tem-recorde-de-crescimento-da-base-de-usuarios/>>

de Wu et al. 2023 explicita o potencial do ChatGPT como ferramenta de correção gramatical em comparativo com o Grammarly. Apesar de possuir um desempenho inferior, o ChatGPT demonstrou um desempenho satisfatório por não ser uma aplicação com esta finalidade especificamente.

A escolha do ChatGPT como ferramenta de PLN para a aplicação foi determinada por alguns fatores, como: o suporte a diversas linguagens, diferente de outras aplicações com esta finalidade, como o próprio Grammarly, e a fácil integração utilizando API. Outro fator crucial é que o ChatGPT é uma ferramenta em constante processo de atualização por parte da equipe da OpenAI. Sua utilização garante que o sistema possa se beneficiar de aprimoramentos futuros, incluindo melhorias no modelo de IA e incorporação de novos dados de treinamento.

A integração com o modelo é feita facilmente utilizando a API fornecida pela OpenAI¹⁰, o que não só agiliza o processo de desenvolvimento mas também garante um menor acoplamento entre o sistema e o ChatGPT, possibilitando a migração para outro modelo de PLN caso necessário. Essa integração opera com base na quantidade de *tokens* enviados e recebidos durante a requisição, seguindo o modelo de custos estabelecido pela OpenAI¹¹. Os custos são vinculados à chave da API, a qual é fornecida pelo próprio usuário no sistema. Esse procedimento é adotado, pois a integração não é gratuita, e seria impraticável para a aplicação arcar com os custos em nome dos usuários.

Para o uso da API, é necessário que três parâmetros sejam enviados nas requisições Hypertext Transfer Protocol (HTTP), sendo estes:

1. Messages: Consiste em uma lista de objetos, cada um contendo um atributo *role* indicando o papel da mensagem, se é do usuário ou do sistema, e um atributo *content* contendo o texto da mensagem. As mensagens são processadas em ordem, e a conversa é construída com base nessas interações.
 - System Role: Mensagens com este papel são usadas para fornecer instruções de alto nível ao modelo. São utilizadas para definir o comportamento

¹⁰ <<https://platform.openai.com/docs/api-reference>>

¹¹ <<https://openai.com/pricing>>

do assistente de linguagem.

- User Role: Indica que a mensagem é do usuário. As mensagens do usuário são essenciais para orientar o modelo sobre a direção da conversa e o que deverá ser realizado.
2. Model: Este parâmetro especifica qual versão do modelo ChatGPT será utilizada para a interação.
 3. API Key: É a chave de identificação do usuário que está realizando a requisição. Através dela o usuário é rastreado e cobrado pelas requisições.

Um elemento crucial na integração com o ChatGPT é a definição do formato do *prompt* enviado durante a requisição, considerando o nosso contexto. É vital para a aplicação que os textos processados mantenham a formatação original realizada pelo usuário. Com essa finalidade, optou-se por empregar o formato HTML, aproveitando suas *tags* para enviar e receber o texto com a formatação correta. Portanto, ao transmitir o texto na requisição HTTP, o campo *System Role* especifica explicitamente que o formato utilizado é HTML e que a resposta deve preservar essa formatação.

Adicionalmente, no *prompt*, é necessário informar a tarefa específica que o ChatGPT deve realizar sobre o texto. Esse detalhe também é incluído na mensagem enviada pelo campo *System Role*, onde uma parte da mensagem é parametrizada com o nome da tarefa a ser executada. A escolha foi feita em prol da utilização do modelo **gpt-4**, por ser a versão mais recente e estável disponível¹².

Outro parâmetro essencial no desenvolvimento do *prompt* é a temperatura de amostragem, que determina o quão aleatórias serão as respostas geradas pelo modelo. A temperatura de amostragem controla a probabilidade de o modelo escolher palavras menos prováveis, mas mais interessantes, em vez de palavras mais prováveis, mas mais comuns. Essa configuração varia de zero a dois, sendo que com o valor zero o modelo tende a ser mais previsível e pragmático, enquanto valores maiores aumentam a versatilidade¹³. No contexto da nossa aplicação, onde a maioria das tarefas se

¹²<<https://platform.openai.com/docs/models>>

¹³<<https://platform.openai.com/docs/guides/text-generation/how-should-i-set-the-temperature-parameter>>

concentra na adequação de padrões textuais, optou-se por manter o valor padrão da API, que é um. Isso faz com que o modelo não seja excessivamente criativo nem excessivamente pragmático.

Aqui está um exemplo de como ficaria o *prompt* completo enviado a API do ChatGPT pelo sistema:

1. Messages:

- User Role: É enviado o HTML referente ao texto formatado.
- System Role: Para orientar o modelo, é utilizada a seguinte mensagem: "Você é uma ferramenta de auxílio na criação e correção de textos e irá receber um texto em formato HTML para (nome da tarefa). Você deverá devolver apenas o HTML do texto novo, mantendo as tags de formatação às partes referentes."

2. Model: É utilizado o modelo **gpt-4**.

3. API Key: É enviada a chave de API do usuário que solicitou a tarefa.

É possível perceber que o parâmetro de temperatura não é definido no corpo da requisição, assim assumindo o valor padrão. Ao enviar uma solicitação à API com esses parâmetros, o ChatGPT processa a conversa e gera uma resposta com base nas mensagens fornecidas. A resposta é tratada no lado do servidor web e utilizada para gerar uma nova versão do documento utilizado na tarefa.

3.3.3 Funcionalidades

Os usuários devem realizar seu cadastro fornecendo as seguintes informações obrigatoriamente: nome, e-mail e senha. Adicionalmente, é solicitado que seja informada a chave de API para a integração com a IA. Contudo, é importante destacar que o preenchimento deste último campo pode ser feito posteriormente, permitindo que o usuário acesse e utilize as funcionalidades básicas do sistema antes de gerar a chave de API.

Após a autenticação, os usuários podem criar e editar documentos de forma ilimitada, aproveitando ferramentas de formatação para estruturar suas ideias. Além disso, eles têm a facilidade de realizar o *download* do documento em formato PDF, mantendo a formatação desejada.

Durante a edição dos documentos, o usuário terá a sua disposição as seguintes tarefas realizadas pela IA:

- Reescrever: Reescreve o texto com outras palavras, mantendo a ideia central.
- Correção Ortográfica: Corrige ortograficamente o texto.
- Formalizar: Altera a escrita do texto para a linguagem formal.
- Sumarizar: Realiza um resumo da ideia extraída do texto.
- Escrita Científica: Altera a escrita do texto para a linguagem científica.
- Tradução: Traduz o texto para a linguagem selecionada.

A cada tarefa realizada será gerada uma nova versão do documento, garantindo assim um melhor controle sobre as mudanças por parte do usuário, permitindo a reversão de alterações que não o agradem.

Tais tarefas só poderão ser utilizadas caso o usuário tenha cadastrado uma chave de API válida, utilizada na comunicação com o ChatGPT. Tanto a chave como o nome e a senha do usuário poderão ser alterados na edição de perfil.

As funcionalidades supracitadas estão descritas, através de casos de uso, no apêndice A .

3.3.4 Banco de Dados e Armazenamento

Visando otimizar o desenvolvimento das funcionalidades, armazenando o mínimo de informação, o banco de dados foi modelado através do uso de apenas três entidades: **User**, **Document** e **Document Version**. A entidade **User** é responsável por representar o usuário no banco de dados, armazenando suas informações básicas

como *name* (Nome), *e-mail* e *password* (Senha), além de armazenar também a *Api Key*, que é utilizada na integração com o ChatGPT. Ela possui relacionamento “um-para-muitos” com a entidade *Document*, uma vez que um usuário pode possuir muitos documentos.

A segunda entidade, **Document**, é a entidade que representa os documentos de um usuário no sistema. Ela é responsável por armazenar o *Title* (título) e o identificador único do usuário que gerou aquele documento, além de possuir uma relação “um-para-muitos” com a entidade **Document Version**, utilizada para o versionamento das edições do documento.

Já a terceira entidade, **Document Version**, mantém o histórico das diferentes mudanças feitas pela IA em um documento, gerando uma nova versão para cada tarefa realizada no texto. Além disso, as versões também armazenam as alterações feitas pelo próprio usuário. Cada registro é identificado por uma *version* (Versão) e armazena o *content* (Conteúdo) correspondente. Ela também armazena a identificador único do documento a qual pertence, fazendo o relacionamento entre as entidades. Na figura 3.7 é possível ver o modelo Entidade-Relacionamento do banco demonstrando todas as relações entre as entidades.

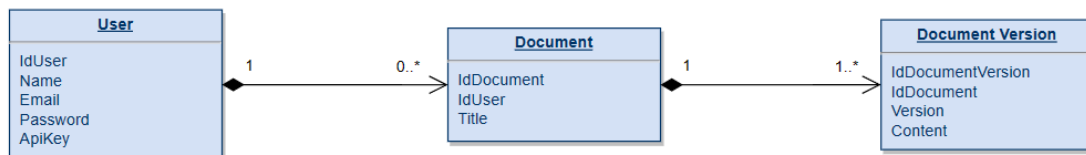


Figura 3.7: Modelo Entidade-Relacionamento do sistema

Capítulo 4

Implementação

Neste capítulo, será detalhada a implementação da solução proposta no capítulo anterior, abordando as tecnologias empregadas, o conceito do padrão de arquitetura Model-View-Controller (MVC) e os resultados obtidos.

4.1 Tecnologias Utilizadas

O servidor web foi desenvolvido utilizando o AdonisJS¹, um *framework* completo para Node.js². Para a construção das interfaces, foram utilizados em conjunto o Tailwind CSS³, um framework Cascading Style Sheets (CSS), o AlpineJS⁴, um microframework JavaScript (JS) e o QuillJS⁵, uma biblioteca JS para auxiliar criação de editores de texto, além do Edge, *template engine* do próprio AdonisJS. O banco de dados é gerenciado pelo PostgreSQL⁶, um Sistema de Gerenciamento de Banco de Dados (SGBD) de código aberto. Tais tecnologias serão descritas nas próximas seções.

¹<<https://adonisjs.com/>>

²<<https://nodejs.org/>>

³<<https://tailwindcss.com/>>

⁴<<https://alpinejs.dev/>>

⁵<<https://quilljs.com/>>

⁶<<https://www.postgresql.org/>>

4.1.1 AdonisJS

O AdonisJS é um *framework* completo focado no desenvolvimento de aplicações web. Ele facilita diversos processos do desenvolvimento, como autenticação e rotas, possuindo *middlewares* que podem ser facilmente configurados, conexão e consultas ao banco utilizando Object-Relational Mapping (ORM) e construção de páginas dinâmicas utilizando *Template Engine*. A figura 4.1 exemplifica a utilização do framework para declaração de rotas, utilizando o *middleware* de autenticação.

```
Route.group(() => {
  Route.post('/correction', 'GptController.textCorrection').as('gpt.textCorrection')
  Route.post('/summarization', 'GptController.textSummarization').as('gpt.summarization')
  Route.post('/translation', 'GptController.textTranslation').as('gpt.translation')
  Route.post('/formalization', 'GptController.textFormalization').as('gpt.formalization')
  Route.post('/rewrite', 'GptController.textRewrite').as('gpt.textRewrite')
  Route.post('/cientific', 'GptController.textCientific').as('gpt.cientific')
})
.prefix('/tasks')
.middleware('auth')
```

Figura 4.1: Rotas declaradas no AdonisJS utilizando *authentication*

Outros critérios que influenciaram na escolha incluem a facilidade na inicialização e configuração de projetos, bem como a qualidade da documentação fornecida, que simplifica o seu uso. Além disso, o AdonisJS utiliza a arquitetura MVC, o que facilita a escalabilidade e manutenção no código, uma vez que lógicas de domínio, negócio e a construção das telas ficam isoladas.

4.1.2 Tailwind CSS

A estilização das páginas foi desenvolvida utilizando o *framework* CSS de código aberto Tailwind. Essa ferramenta oferece uma ampla gama de classes CSS predefinidas destinadas aos elementos HTML, reduzindo a necessidade de criação manual de classes por parte dos desenvolvedores. Essa abordagem proporciona uma experiência mais eficiente e prática na estilização das páginas, destacando-se por sua flexibilidade e agilidade no processo de design. Além disso, essa ferramenta faz com que seja muito prática a criação das telas de maneira responsiva, com suas classes

funcionando para telas de *mobile* e *desktop*. A figura 4.2 demonstra a utilização das classes do Tailwind para construção de um elemento HTML.

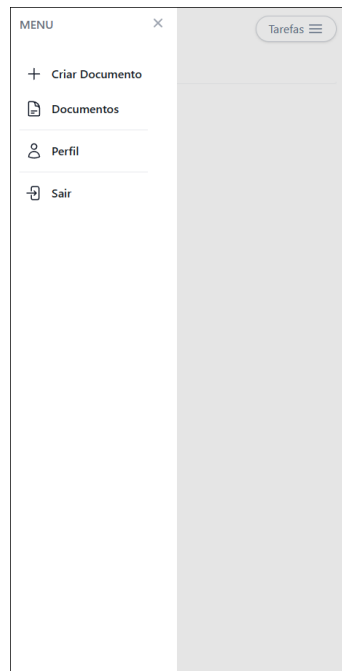
```
<li class="cursor-pointer">
  <a href="{ route('users.show') }"
    class="flex items-center p-2 text-gray-900 rounded-lg dark:text-white hover:bg-gray-100 dark:hover:bg-gray-700 group">
    <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"><path fill="none" stroke="currentColor" ...
    </svg>
    <span class="flex-1 ml-3 whitespace-nowrap">Perfil</span>
  </a>
</li>
```

Figura 4.2: Estilização de elemento HTML utilizando classes do Tailwind

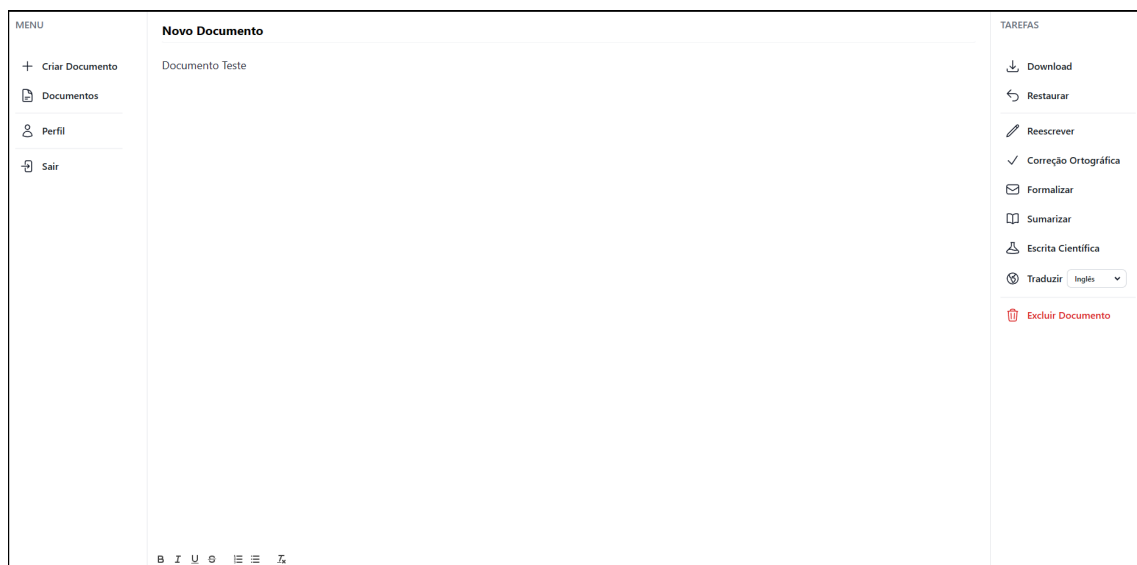
4.1.3 AlpineJS

Na elaboração das interfaces foi utilizado o AlpineJS, um *microframework* JS projetado para conferir comportamento e gerenciamento de estado a componentes HTML. A integração deste *framework* simplificou consideravelmente as operações relacionadas ao controle de visibilidade de menus e à realização de chamadas de funções assíncronas, contribuindo para uma implementação mais eficiente e fluida dessas interações no sistema.

Na tela de edição de um documento, por exemplo, o controle da exibição dos menus laterais foi feito utilizando o AlpineJS. O *microframework* simplificou o processo de desenvolvimento do comportamento adaptável à resolução da tela, que varia na versão *mobile* e *desktop*, como pode ser observado na figura 4.3.



(a) Exibição de menu na versão *mobile* da aplicação



(b) Exibição de menu na versão *desktop* da aplicação

Figura 4.3: Comparativo da exibição do menu nas versões *mobile* e *desktop*

4.1.4 QuillJS

O QuillJS é uma biblioteca JS que auxilia a criação de editores de texto para aplicações web. Ele se caracteriza como um editor What You See Is What You

Get (WYSIWYG), um tipo de editor HTML onde o texto visualizado durante a edição reflete exatamente o seu resultado final, permitindo ao usuário visualizar a organização e formatação do texto em tempo real.

A biblioteca oferece um leque de possibilidades de personalização. Além de alterar a estilização, também é possível adicionar opções de formatação ao editor, como negrito e itálico, e funcionalidades avançadas, como verificação ortográfica (*spell check*), adaptando o editor de acordo com as necessidades específicas do projeto. A versão final do editor integrado à aplicação pode ser visualizada na figura 4.4

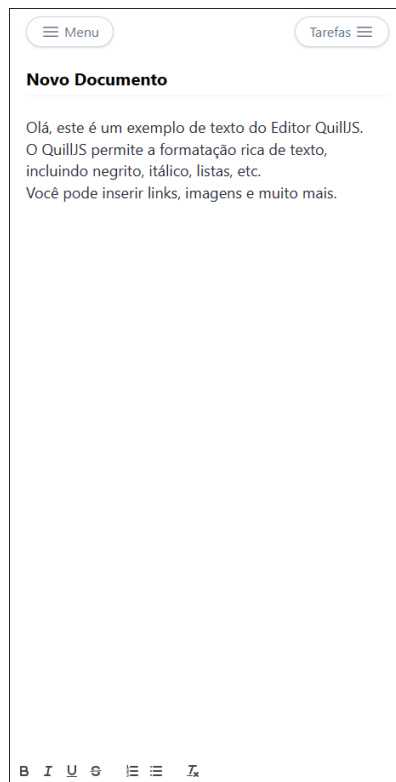


Figura 4.4: Editor de texto da aplicação utilizando o QuillJS

4.1.5 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto, reconhecido por sua confiabilidade e flexibilidade. Com uma ampla gama de recursos, incluindo suporte a SQL avançado e transações ACID, o PostgreSQL é uma escolha popular para desenvolvedores que buscam soluções de banco de dados mais robustas. Além disso, ele é facilmente integrado ao AdonisJS,

fator determinante para a sua escolha. Inicialmente, foi utilizado o SQLite, uma vez que a arquitetura de banco da aplicação é relativamente simples, porém pensando na escalabilidade da aplicação, foi feita a migração para o PostgreSQL.

4.2 Padrão MVC

Como descrito na seção 4.1.1, o adonisJS utiliza a arquitetura MVC para modularizar seu código. Esse padrão foi utilizado com a adição da camada *Service* (Serviço). A incorporação da camada de serviço oferece uma vantagem significativa ao separar as responsabilidades de lógica de negócios e manipulação de dados. Promovendo uma arquitetura mais modular e organizada, também facilita a manutenção e extensão do sistema, permitindo que as operações de serviço sejam reutilizadas em diferentes partes da aplicação. Essa abordagem contribui para um código mais coeso, escalável e de fácil manutenção. A figura 4.5 ilustra o fluxo de informação entre as camadas do sistema:

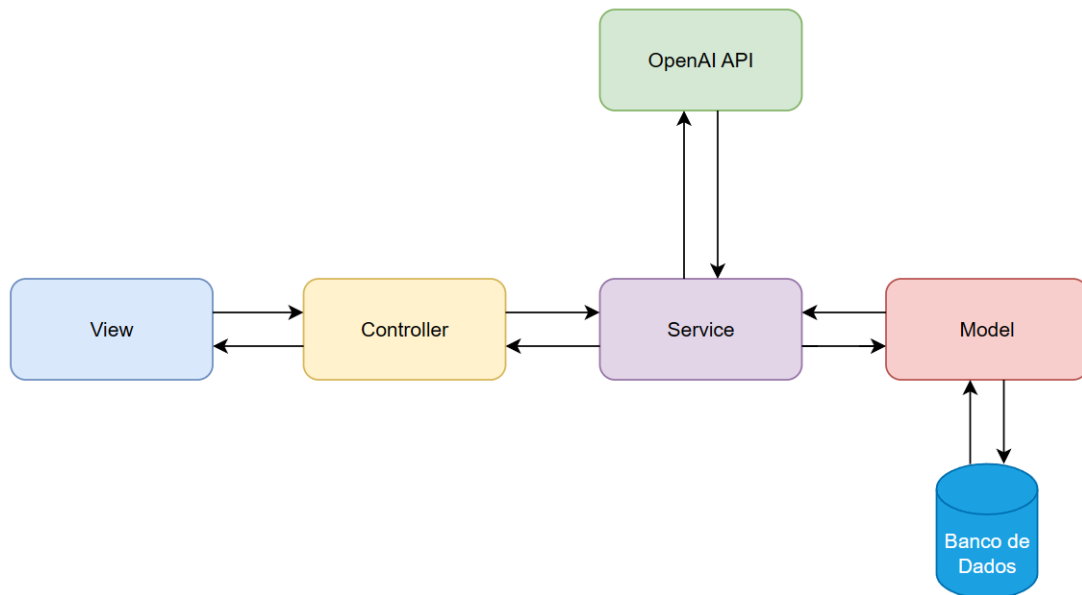


Figura 4.5: Arquitetura utilizada no sistema

4.2.1 View

A camada de *View* (Visualização) é onde são gerados os documentos HTML que serão exibidos no navegador do cliente. Essa camada é responsável por receber os dados processados pela *Controller* e apresentá-los organizadamente ao usuário final, sendo as páginas construídas utilizando o Edge, *template engine* do AdonisJS, em conjunto do Tailwind CSS, QuillJS e AlpineJS.

4.2.2 Controller

A camada de *Controller* (Controlador) é responsável por realizar a comunicação entre a camada de *View* e a camada de *Service*. Em termos gerais, sua função é atuar como uma intermediária entre a apresentação e a persistência dos dados, processando as requisições e gerando respostas. Essa intermediação é essencial para garantir a comunicação eficiente entre a interface do usuário e a lógica de negócios encapsulada na camada de *Service*. Ao receber dados da *View*, o *Controller* encaminha essas informações para a camada de *Service*, onde são realizadas operações de processamento e, eventualmente, interações com o *Model* ou APIs externas.

4.2.3 Service

A camada de *Service* atua como uma interface intermediária entre a *Controller*, a *Model* e, de maneira adicional, serviços externos, como a API utilizada para comunicação com o ChatGPT. Sua principal responsabilidade é encapsular a lógica de negócios da aplicação, promovendo modularidade e reutilização de código. Ao receber requisições do *Controller*, o *Service* executa operações específicas de negócios, incluindo a comunicação com o *Model* para acessar ou manipular dados no banco ou interagir com serviços externos, como a API do ChatGPT.

Quando se trata de interações com o ChatGPT, a camada de *Service* assume a tarefa de realizar as solicitações, gerando o *prompt* personalizado baseado no modelo descrito na seção 3.3.2, enviando utilizando a biblioteca da OpenAI e processando a resposta para retornar ao *Controller*. Assim, a cada solicitação, o *Service* gera

a mensagem *System* referente à tarefa a ser realizada e envia junto da mensagem *User* com o corpo do texto a ser trabalhada, exemplificado no código 4.1. A resposta recebida é extraída do campo *choices*, demonstrado nos códigos 4.2 e 4.3, processada e repassada para o *Controller*.

```
1 {
2   messages: [
3     {
4       role: 'system',
5       content: 'Você é uma ferramenta de auxílio na criação e
6                 correção de textos e irá receber um texto em formato
7                 HTML para reescrever na linguagem formal. Você deverá
8                 devolver apenas o HTML do texto novo, mantendo as
9                 tags de formatação às partes referentes'
10    },
11    {
12      role: 'user',
13      content: 'eu me amarro em jogar bola e assistir tv'
14    }
15  ],
16  model: 'gpt-4'
17 }
```

Código 4.1: Corpo da requisição enviada ao ChatGPT

```
1 {
2   id: 'chatcmpl-8SAwKsg9adjmYqyNdIMPKwthPWCwB',
3   object: 'chat.completion',
4   created: 1701725928,
5   model: 'gpt-4-0613',
6   choices: [ { index: 0, message: [Object], finish_reason: '
7     stop' } ],
8   usage: { prompt_tokens: 116, completion_tokens: 20,
9     total_tokens: 136 },
10  system_fingerprint: null
11 }
```

Código 4.2: Resposta recebida do ChatGPT

```
1 [
2   {
3     index: 0,
4     message: {
5       role: 'assistant',
6       content: '<p>Eu tenho grande interesse em jogar futebol
7         e assistir televisão.</p>'
8     },
9     finish_reason: 'stop'
10  }
11 ]
```

Código 4.3: Campo *choices* extraído da resposta do ChatGPT

4.2.4 Model

A camada de *Model* é responsável pela persistência dos dados da aplicação, realizando operações de leitura e escrita dos dados no banco. Nela é onde ocorre o mapeamento das entidades (User, Document e Document Version) e seus relacionamentos, definindo a estrutura e a integridade dos dados subjacentes. As *Models*

foram construídas utilizando o Lucid, ORM para AdonisJS, responsável por abstrair questões de conexão com o banco de dados, gerenciar versionamento do banco e possibilitar a realização de operações utilizando *Query Builder*.

4.3 Interfaces

Como dito anteriormente, a aplicação possui foco na versão *mobile*, buscando promover seu uso por um maior número de usuários. Além disso, destaca-se a importância da clareza na compreensão da interface. Nesta seção, serão apresentadas as interfaces do sistema, com comparações entre as versões *mobile* e *desktop*, além de oferecer explicações sobre a lógica da criação dessas interfaces. Cada subseção a seguir abordará uma interface específica do sistema.

4.3.1 Cadastro e Autenticação do Usuário

Uma funcionalidade básica, porém essencial para o sistema, é a capacidade de permitir que os usuários se cadastrem e autentiquem, viabilizando a criação de suas contas para iniciar efetivamente a utilização da aplicação. Para estas telas, a abordagem visa simplificar ao máximo o processo, evitando a exigência de uma extensa quantidade de dados.

No cadastro é necessário apenas informar o nome, e-mail, senha e opcionalmente a chave da API para integração com o ChatGPT, como exemplificado na figura 4.6(a). A chave também pode ser inserida posteriormente na edição de perfil do usuário. Por outro lado, para a autenticação do usuário, é preciso informar o e-mail e a senha para iniciar a utilização do sistema, conforme demonstrado na figura 4.6(b). Após a autenticação bem-sucedida, o usuário é redirecionado para a tela de documentos.

The image shows two side-by-side screenshots of the Syntactic web application interface. Both screens feature the Syntactic logo at the top left.

Screen (a) is titled "Cadastre-se grátis" (Sign up for free). It includes a link "Já possui uma conta? Faça login." (Already have an account? Log in). Below this are four input fields: "Nome" (Name), "Email", "Senha" (Password), and "ChatGPT API Key". A blue button labeled "Cadastrar →" (Sign up) is at the bottom.

Screen (b) is titled "Entre com a sua conta" (Log in with your account). It includes a link "Não possui uma conta? Cadastre-se aqui." (Don't have an account? Sign up here). Below this are two input fields: "Email" and "Senha". A blue button labeled "Login →" is at the bottom.

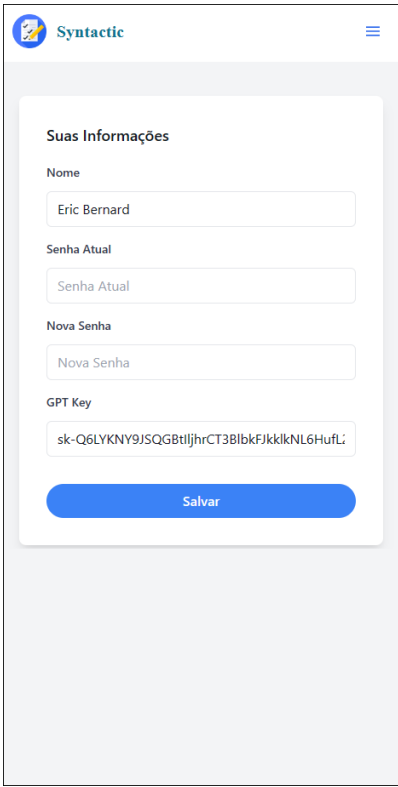
(a) Tela de cadastro de um novo usuário.

(b) Tela de autenticação do usuário.

Figura 4.6: Telas de cadastro e autenticação do sistema.

4.3.2 Edição de informações do Usuário

Devido à necessidade de informar uma chave para a utilização da API do ChatGPT, é importante fornecer a possibilidade do usuário editar essa informação após o cadastro. Além disso, a tela permite que o usuário atualize seu nome e sua senha, caso necessário. Para a atualização do cadastro com as novas informações, é necessário que o usuário insira sua senha atual, adicionando uma camada de segurança no processo. Essa tela pode ser observada na figura 4.7



The image shows a mobile application interface for editing user information. At the top, there is a header with the 'Syntactic' logo on the left and a hamburger menu icon on the right. Below the header, the main content area is titled 'Suas Informações'. This area contains four input fields, each with a label above it: 'Nome' (Name) containing 'Eric Bernard', 'Senha Atual' (Current Password) containing 'Senha Atual', 'Nova Senha' (New Password) containing 'Nova Senha', and 'GPT Key' containing a long alphanumeric string. At the bottom of this form is a blue button labeled 'Salvar' (Save).

Figura 4.7: Tela de edição das informações do usuário

4.3.3 Documentos do Usuário

Na tela de documentos, o usuário pode fazer o gerenciamento de seus documentos, permitindo a criação, exclusão ou abertura de um documento para a edição. O foco na construção da tela reside na facilitação do entendimento das funcionalidades, visando tornar o funcionamento o mais acessível possível. Com isso, para criar um documento é necessário apenas que coloque um nome e aperte um botão, redirecionando o usuário para a tela de edição do documento. Além disso, os botões de excluir e editar são bem sugestivos com relação a sua função. O comparativo entre *mobile* e *desktop* pode ser observado nas figuras 4.8 e 4.9 respectivamente.

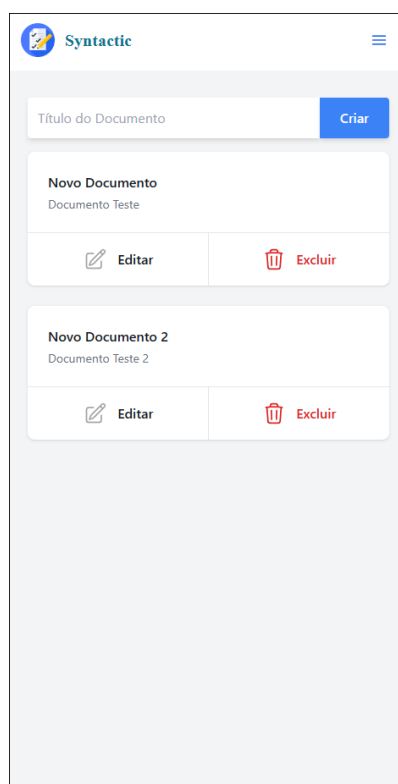


Figura 4.8: Tela de documentos do usuário na versão *mobile*

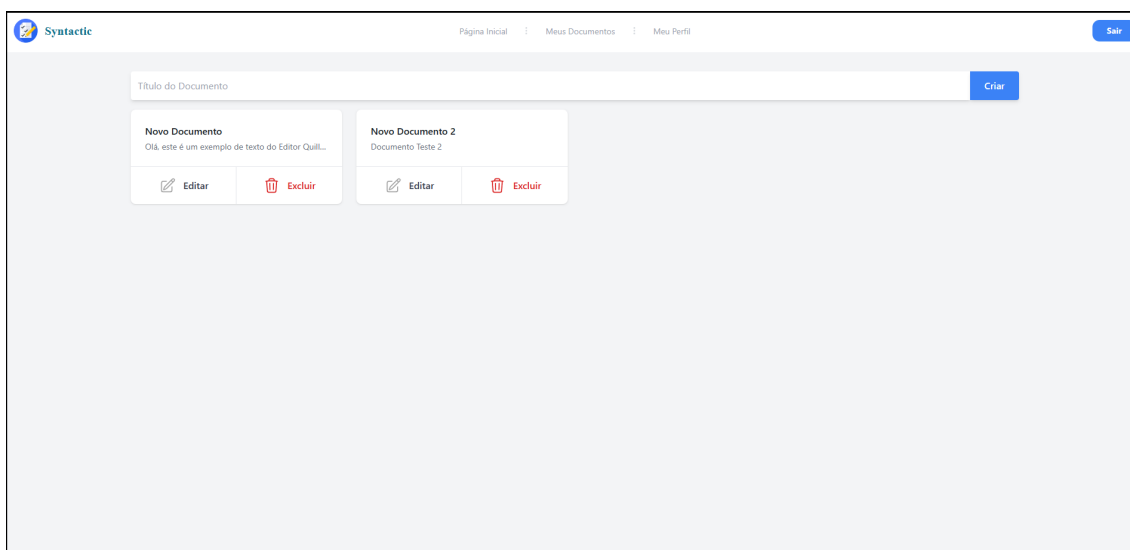


Figura 4.9: Tela de documentos do usuário na versão *desktop*

4.3.4 Editor de Documentos

O editor de documentos é a tela responsável pela principal funcionalidade da aplicação. Nele o usuário pode redigir seus documentos e utilizar da ajuda do ChatGPT para realizar alterações no texto. O propósito primordial desta tela é proporcionar ao usuário uma experiência de edição que se assemelhe ao máximo à sensação de redigir um documento. Para atingir esse objetivo, optou-se por criar uma interface com o mínimo de distrações possível. Para isso, na versão *mobile* os menus laterais ficam escondidos e podem ser exibidos através de botões, comportamento demonstrado nas figuras 4.10 e 4.11. Tal comportamento é diferente na versão *desktop*, onde os menus laterais são sempre exibidos, como pode ser observado na figura 4.12. Além disso, nas figuras 4.13, 4.14 e 4.15 são exemplificadas os usos das tarefas de tradução, formalização e correção ortográfica, respectivamente.

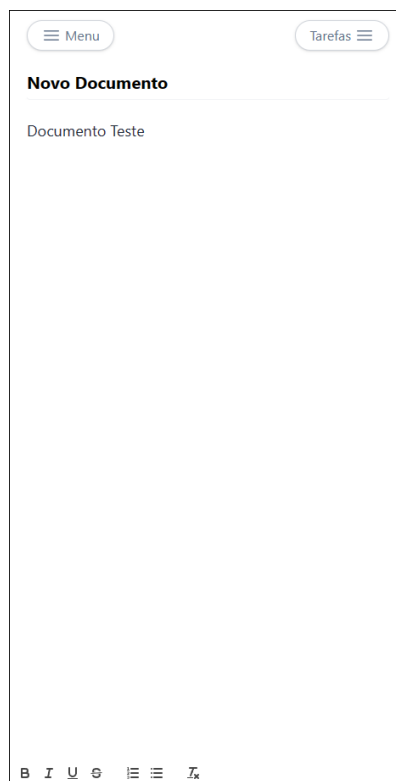
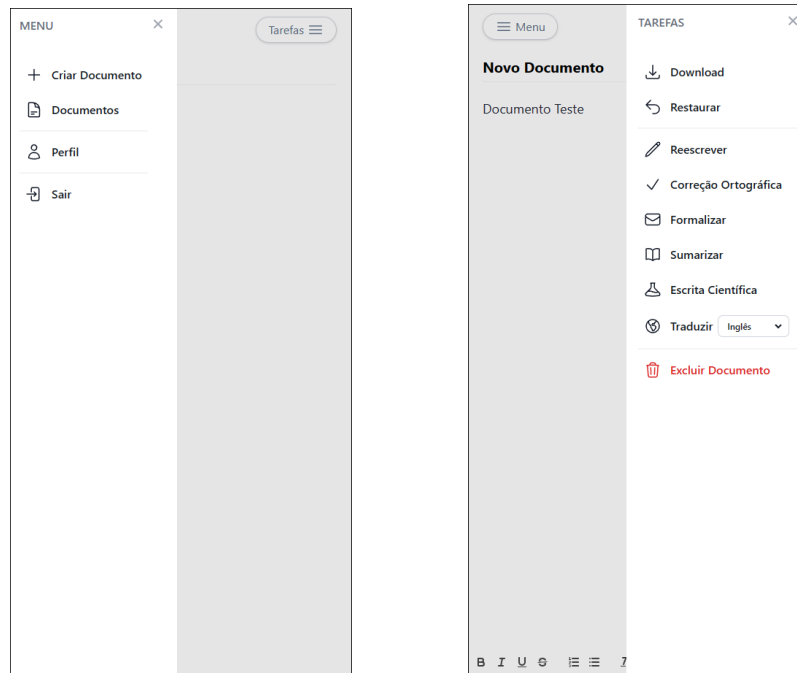


Figura 4.10: Tela de edição de documento (editor) na versão *mobile*



(a) Menu da aplicação na tela de edição de documentos (*mobile*)

(b) Menu de tarefas na tela de edição de documentos (*mobile*)

Figura 4.11: Menus laterais na tela de edição de documentos

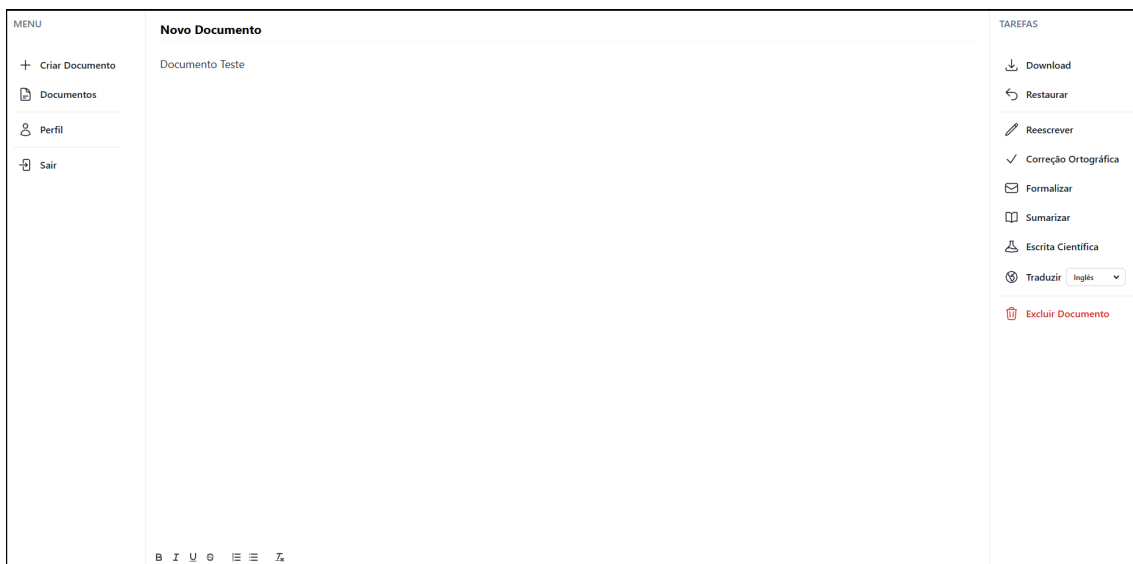
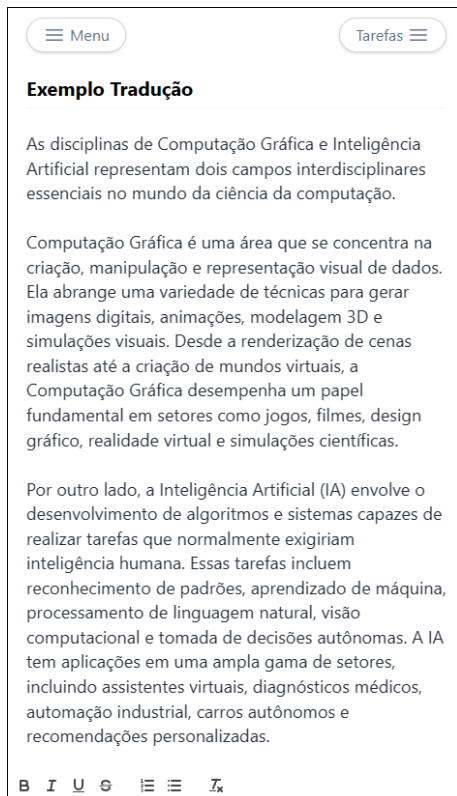
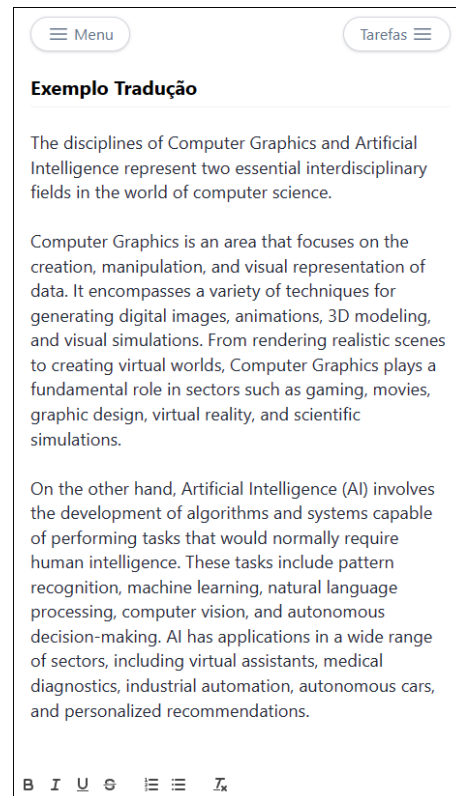


Figura 4.12: Tela de edição de documento (editor) na versão *desktop*

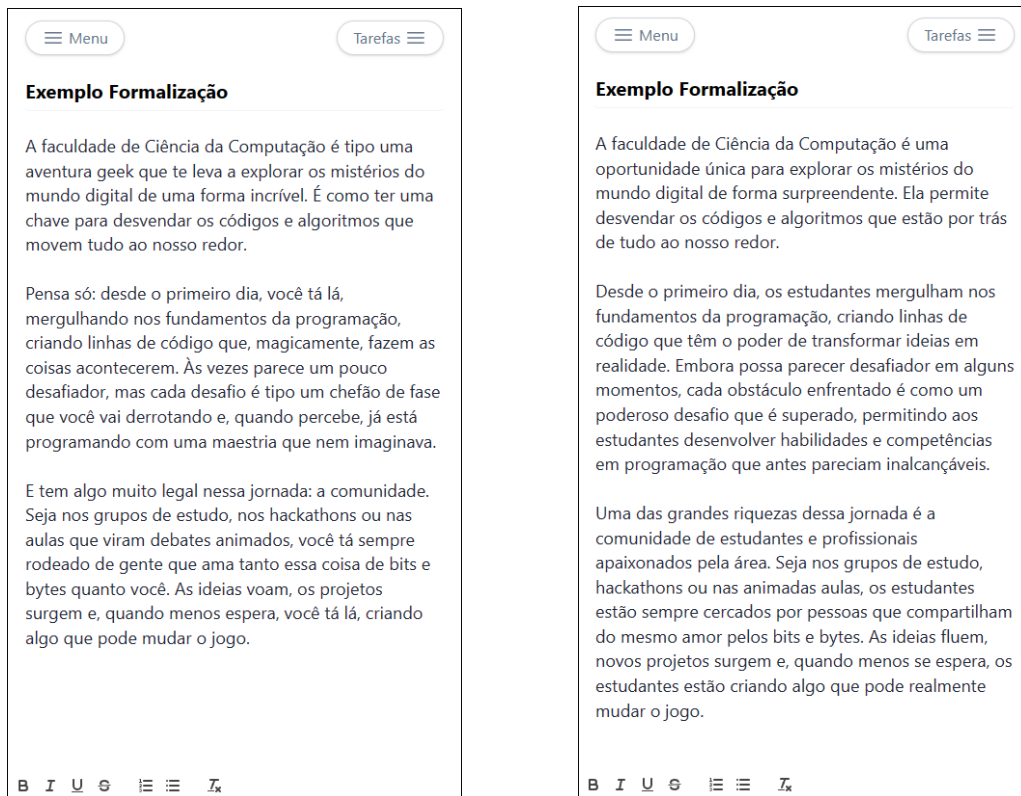


(a) Tela de edição de documento com um texto de exemplo



(b) Tela de edição de documento com o texto exemplo traduzido

Figura 4.13: Tela de edição de documentos com um exemplo da tarefa de tradução.



(a) Tela de edição de documentos com um texto de exemplo

(b) Tela de edição de documentos com o texto exemplo formalizado

Figura 4.14: Tela de edição de documentos com um exemplo da tarefa de formalização.



(a) Tela de edição de documentos com um texto de exemplo

(b) Tela de edição de documentos com o texto exemplo corrigido ortograficamente

Figura 4.15: Tela de edição de documentos com um exemplo da tarefa de correção ortográfica.

Capítulo 5

Conclusão

Este capítulo abordará as considerações finais dos autores do trabalho sobre o desenvolvimento da aplicação. Além disso, serão abordadas limitações encontradas durante o desenvolvimento e possíveis trabalhos futuros que agregariam à proposta.

5.1 Considerações finais

Neste trabalho, foi desenvolvido um sistema web que visa auxiliar os usuários na escrita e aprimoramento de textos, utilizando o ChatGPT, um modelo de LLM. O sistema oferece funcionalidades como tradução, sumarização, formalização, escrita científica, entre outras, de forma ágil e interativa, integrando o ChatGPT por meio de uma API. O sistema também permite aos usuários redigir e armazenar documentos de maneira eficiente, aproveitando ferramentas de formatação e *download*. Além disso, o sistema se destaca pela sua praticidade de uso em dispositivos móveis, proporcionando uma ferramenta acessível e versátil.

O trabalho atingiu os objetivos propostos, apresentando uma solução inovadora e útil para o problema da escrita. A aplicação foi implementada utilizando tecnologias modernas, seguindo um padrão de arquitetura que facilita a escalabilidade e manutenção do código. A integração com o ChatGPT foi realizada de forma simples e eficaz, aproveitando a capacidade do modelo de lidar com diversas tarefas de PLN.

A interface do usuário foi desenvolvida com foco na usabilidade e na experiência do usuário, oferecendo um editor de texto intuitivo e funcional.

5.2 Limitações e trabalhos futuros

Durante o desenvolvimento do trabalho enfrentamos algumas limitações que podem ser superadas em trabalhos futuros. Uma delas foi a quantidade máxima de *tokens* na requisição ao ChatGPT, que restringiu o tamanho dos textos que podiam ser processados pela IA. Devido a isso, não foi possível utilizar o texto extraído no formato do QuillJS, que ajuda a preservar a formatação. Para contornar esse problema, utilizamos o formato HTML para enviar os textos, mas isso ainda impôs uma perda de informação sobre a formatação original. Outra alternativa para contornar o problema seria a fragmentação do texto junto de um sistema de filas, citada na seção 3.3.1.

Outra limitação foi a ausência de uma funcionalidade de busca de documentos pelo usuário, que dificultou a localização dos textos produzidos. Uma técnica que poderia ser empregada para resolver essa questão seria o *Full-Text Search*, que permitiria realizar buscas mais contextuais e abrangentes nos documentos armazenados. O PostgreSQL, conhecido por sua robustez e capacidade de suportar recursos avançados, oferece uma implementação eficiente e acessível do *Full-Text Search*. A facilidade de integração com o PostgreSQL simplificaria significativamente o processo de implementação dessa funcionalidade na aplicação.

Por fim, a implementação de um controle de custos nativo seria uma adição significativa à aplicação. A capacidade de calcular o custo das requisições ao ChatGPT diretamente no servidor, utilizando as informações sobre a quantidade de *tokens* enviados e recebidos, apresenta uma abordagem vantajosa do ponto de vista do usuário. Essa funcionalidade eliminaria a necessidade de criar uma conta e obter uma chave de API em uma plataforma externa para integração, simplificando o processo para os usuários e proporcionando uma experiência mais fluida.

Além da praticidade, a introdução do controle de custos no lado do servidor

abriria possibilidades interessantes, como a implementação de períodos de teste. Isso permitiria aos usuários explorar a aplicação e avaliar sua utilidade sem a preocupação imediata de custos, incentivando a experimentação e adoção inicial. Dessa forma, a incorporação desse controle de custos não apenas simplificaria a integração para os usuários, mas também contribuiria para uma abordagem mais flexível e acessível em termos de custos.

Referências

ARACI, D. *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models*. 2019.

ARMANDA, M. L. et al. “grammarly” as english writing assistant from efl students’ perspective. *English Education: Journal of English Teaching and Research*, v. 7, n. 2, p. 128–137, 2022.

BLANCO-GONZÁLEZ, A. et al. The role of ai in drug discovery: Challenges, opportunities, and strategies. *Pharmaceuticals*, MDPI AG, v. 16, n. 6, p. 891, jun. 2023. ISSN 1424-8247. Disponível em: <<http://dx.doi.org/10.3390/ph16060891>>.

BRANTS, T. et al. Large language models in machine translation. 2007.

BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020.

BUBECK, S. et al. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, v. 9, n. 2, p. 48–57, 2014.

CHO, K. et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014.

GARG, R. et al. Exploring the role of chatgpt in patient care (diagnosis and treatment) and medical research: A systematic review. *Health Promot Perspect*, Tabriz University of Medical Sciences, v. 13, n. 3, p. 183–191, 2023. Published 2023 Sep 11.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Disponível em: <<https://doi.org/10.1162/neco.1997.9.8.1735>>.

JAIN, L. C.; MEDSKER, L. R. *Recurrent Neural Networks: Design and Applications*. 1st. ed. USA: CRC Press, Inc., 1999. ISBN 0849371813.

KADDOUR, J. et al. *Challenges and Applications of Large Language Models*. 2023.

KALLA, D. et al. Study and analysis of chat gpt and its impact on different fields of study. *International Journal of Innovative Science and Research Technology*, v. 8, n. 3, March 2023. Available at SSRN: <<https://ssrn.com/abstract=4402499>>.

- KAPLAN, J. et al. Scaling laws for neural language models. 2020.
- KASNECI, E. et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, v. 103, p. 102274, 2023. ISSN 1041-6080. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1041608023000195>>.
- LI, J. et al. *Pretrained Language Models for Text Generation: A Survey*. 2022.
- RADFORD, A. et al. Language models are unsupervised multitask learners. In: . [s.n.], 2019. Disponível em: <<https://api.semanticscholar.org/CorpusID:160025533>>.
- RAHIMI, M.; ZHANG, L. J. Writing task complexity, students' motivational beliefs, anxiety and their writing production in english as a second language. *Reading and Writing*, v. 32, p. 761–786, 2019. Disponível em: <<https://doi.org/10.1007/s11145-018-9887-9>>.
- SCHMIDT, R. M. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019.
- SHERSTINSKY, A. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, Elsevier BV, v. 404, p. 132306, mar. 2020. ISSN 0167-2789. Disponível em: <<http://dx.doi.org/10.1016/j.physd.2019.132306>>.
- SUN, L. et al. *SciEval: A Multi-Level Large Language Model Evaluation Benchmark for Scientific Research*. 2023.
- SUTSKEVER, O. V. I.; LE, Q. V. *Sequence to Sequence Learning with Neural Networks*. 2014.
- THOPPILAN, R. et al. *LaMDA: Language Models for Dialog Applications*. 2022.
- VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems 30*. [s.n.], 2017. p. 5998–6008. Disponível em: <<http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>>.
- WU, H. et al. *ChatGPT or Grammarly? Evaluating ChatGPT on Grammatical Error Correction Benchmark*. 2023.
- YASRAB, R.; POUND, M. *PhenomNet: Bridging Phenotype-Genotype Gap: A CNN-LSTM Based Automatic Plant Root Anatomization System*. 2020.
- YU, L. Q. F.; SCHILDER, F. *Legal Prompting: Teaching a Language Model to Think Like a Lawyer*. 2022.
- ZHANG, T. et al. *Benchmarking Large Language Models for News Summarization*. 2023.
- ZHAO, W. X. et al. *A Survey of Large Language Models*. 2023.

Apêndice A

Casos de Uso

O sistema conta com apenas um ator: o **Usuário**. As ações realizadas pelo usuário são descritas nos seguintes casos de uso:

UC1 - Caso de Uso: Cadastrar Usuário

- Ator: Usuário
- Descrição: Criação de Usuário
- Fluxo Principal:
 1. O usuário acessa a página de cadastro.
 2. O usuário informa seu nome, email, senha e, opcionalmente, a chave de API para integração com o ChatGPT.
 3. O usuário confirma o cadastro.
 4. O sistema valida os dados.
 5. O sistema exibe o menu de documentos.
- Fluxo de Exceção:
 1. Se o email já estiver presente no banco de dados, o sistema exibe uma mensagem de erro.

UC2 - Caso de Uso: Autenticar Usuário

- Ator: Usuário
- Descrição: Autenticação de Usuário
- Fluxo Principal:
 1. O usuário acessa a página de *login*.
 2. O usuário informa seu email e senha.
 3. O usuário confirma o *login*.
 4. O sistema valida os dados.
 5. O sistema exibe o menu de documentos.
- Fluxo de Exceção:
 1. Se as credenciais estiverem incorretas, o sistema exibe uma mensagem de erro.

UC3 - Caso de Uso: Alterar Usuário

- Ator: Usuário
- Descrição: Alteração dos dados de um usuário
- Pré-requisitos: UC2 - Autenticar Usuário
- Fluxo Principal:
 1. O usuário acessa a página de edição de perfil.
 2. O sistema exibe o nome, email e chave de API do usuário, possibilitando a edição do nome, senha e chave de API.
 3. O usuário informa as novas informações, junto da senha atual.
 4. O sistema valida e altera as informações.
 5. O sistema exibe a página de edição de perfil com os dados atualizados.

- Fluxo de Exceção:

1. Se o usuário não informar a senha atual corretamente, o sistema não atualiza as informações e exibe uma mensagem de erro.

UC4 - Caso de Uso: Criar Documento

- Ator: Usuário

- Descrição: Criação de um novo documento

- Pré-requisitos: UC2 - Autenticar Usuário

- Fluxo Principal:

1. O usuário acessa a página de documentos.
2. O usuário informa o título do documento.
3. O usuário confirma a criação do documento.
4. O sistema cria o novo documento.
5. O sistema exibe o novo documento no menu de documentos.

- Fluxo de Exceção:

1. Se o usuário não informar um título, o documento é criado com o título "Novo Documento"

UC5 - Caso de Uso: Acessar Documento

- Ator: Usuário

- Descrição: Acessar um documento para edição

- Pré-requisitos: UC2 - Autenticar Usuário, UC4 - Criar Documento

- Fluxo Principal:

1. O usuário acessa a página de documentos.

2. O usuário seleciona a opção Editar de um documento.
3. O sistema exibe a tela de edição do documento.

UC6 - Caso de Uso: Excluir Documento

- Ator: Usuário
- Descrição: Exclusão de um documento
- Pré-requisitos: UC2 - Autenticar Usuário, UC4 - Criar Documento
- Fluxo Principal:
 1. O usuário acessa a página de documentos.
 2. O usuário seleciona a opção Excluir de um documento.
 3. O sistema exclui o documento.
 4. O sistema exibe o menu de documentos.

UC7 - Caso de Uso: Alterar Título do Documento

- Ator: Usuário
- Descrição: Alteração do título de um documento
- Pré-requisitos: UC2 - Autenticar Usuário, UC4 - Criar Documento, UC5 - Acessar Documento
- Fluxo Principal:
 1. O usuário altera o título do documento.
 2. O sistema salva o novo título do documento.

UC8 - Caso de Uso: Editar Documento

- Ator: Usuário
- Descrição: Edição do texto de um documento

- Pré-requisitos: UC2 - Autenticar Usuário, UC4 - Criar Documento, UC5 - Acessar Documento
- Fluxo Principal:
 1. O usuário realiza alterações no corpo do documento.
 2. Após 1,5 segundos sem alterações, o sistema salva o documento.

UC9 - Caso de Uso: Realizar Tarefa no Documento

- Ator: Usuário
- Descrição: Utilizar a IA para realizar alguma tarefa no documento
- Pré-requisitos: UC2 - Autenticar Usuário, UC4 - Criar Documento, UC5 - Acessar Documento
- Fluxo Principal:
 1. O usuário abre o menu de tarefas do documento.
 2. O usuário seleciona uma das tarefas realizadas pela IA.
 3. Uma nova versão do documento é criada com o resultado da tarefa.
 4. O sistema exibe a nova versão do documento.
- Fluxo de Exceção:
 1. Se o usuário não possuir uma chave de API válida cadastrada, o sistema exibe uma mensagem solicitando a alteração da chave.

UC10 - Caso de Uso: Restaurar Documento

- Ator: Usuário
- Descrição: Restaurar um documento para a versão anterior à última tarefa realizada por IA.

- Pré-requisitos: UC2 - Autenticar Usuário, UC4 - Criar Documento, UC5 - Acessar Documento, UC9 - Realizar Tarefa no Documento
- Fluxo Principal:
 1. O usuário abre o menu de tarefas do documento
 2. O usuário seleciona a opção Restaurar.
 3. A versão atual do documento é excluída.
 4. O sistema exibe a versão anterior do documento.

UC11 - Caso de Uso: Baixar Documento

- Ator: Usuário
- Descrição: *Download* de um documento
- Pré-requisitos: UC2 - Autenticar Usuário, UC4 - Criar Documento, UC5 - Acessar Documento
- Fluxo Principal:
 1. O usuário abre o menu de tarefas do documento.
 2. O usuário seleciona a opção *Download*.
 3. A versão atual do documento é transferida em formato PDF.