Fundação Universidade Federal do ABC
Pró reitoria de pesquisa
Av. dos Estados, 5001, Santa Terezinha, Santo André/SP, CEP 09210-580
Bloco L, 3ºAndar, Fone (11) 3356-7617
iniciacao@ufabc.edu.br

Projeto de Iniciação Científica submetido para avaliação no Edital: Edital Nº 04/2024 (PIC/PIBIC/PIBIT/PIBIT-AF)

**Título do projeto:** Sistema offline para pesquisa bibliográfica baseado em LLM e RAG.

**Discente:** Maria Eduarda Hamparsomian

**Orientador:** Hugo Puertas de Araujo

**Palavras-chave do projeto:** Redes Neurais Artificiais; RAG; Deep Learning; LLM.

**Área do conhecimento do projeto:** Machine Learning.

# Sumário

# 1 Resumo

A área da tecnologia sempre foi de extrema importância para a humanidade, e nos últimos anos seus avanços foram extremamente significativos, o que nos permite atualmente superar barreiras importantes na área de pesquisa, que são: pesquisar em diversas fontes ao mesmo tempo, leituras muito extensas que limitam o tempo do pesquisador e falta de auxílio na hora de se estudar e entender essas leituras. Vinda desta necessidade, criou-se um sistema que acessa arquivos em PDF em uma pasta local, permitindo que o usuário interaja com essas pastas através de uma interface do tipo chat. O sistema conta com um banco de dados vetorial, técnicas de RAG, Retrieval-Augmented Generation, e o Machine Learning, técnica usada para gerar embeddings dos textos e as respostas para as perguntas do usuário através de LLM rodando localmente. O Retrieval-Augmented Generation (RAG), nada mais é do que o processo de gerar textos através de um LLM, mas não a partir das entradas com as quais o mesmo foi originalmente treinado, mas sim a partir das informações extraídas de uma base de conhecimento confiável. Desta forma, evita-se que ocorram as “ilusões”, ou “alucinações”, que são erros de geração que o modelo comete e que causa desinformação, deixando o texto incoerente com a realidade ou com o contexto. Elas ocorrem principalmente devido à combinação de probabilidades de geração de texto, complexidade da linguagem natural e às limitações dos modelos em “entender” o contexto de maneira humana.

# 2 Introdução e Justificativa

## 2.1 Introdução

Atualmente, com o desenvolvimento da tecnologia, é cada vez maior a parcela da população que possui dispositivos com as mais diversas funções para o cotidiano. Entre as inúmeras ferramentas, programas que se utilizam de inteligência artificial vêm cada vez mais ganhando seu espaço no mercado, especialmente aquelas que usam Large Language Models (LLM's), que são os modelos de IA treinados para interagir com o usuário usando uma linguagem natural na hora de gerarem textos, de forma que fique difícil diferenciar a conversa com o programa de uma outra com um ser humano real, como o famoso chat Generative Pretrained Transformer (GPT) e diversos outros chatbots interativos (LIMA, et al, 2014).

O chat GPT é uma das principais ferramentas utilizadas em aplicações que utilizam conversação natural. Porém, esse sistema não consegue acessar dados ou arquivos para interagir de forma eficiente, pois apesar de nele existir a função de carregar e ler pdf's, sua execução é lenta e não confiável nos modelos gratuitos, caso o usuário queira falar sobre algum texto de forma mais assertiva, ele tem que copiar e colar na interface do site, o que nem sempre é possível, dependendo do tamanho do texto. Além do fato de que, há também a possibilidade deste gerar alguma incoerência textual, uma vez que o foco do GPT não é trabalhar com dados fornecidos pelo usuário. Devido a esta limitação, uma possibilidade para superar esta barreira é a utilização de um programa específico que consiga acessar uma pasta com os arquivos fornecidos pelo usuário e gerar textos e interações a partir deles, utilizando-se: LLM, citado anteriormente, Machine Learning, Retrieval-Augmented Generation (RAG), também já citado anteriormente, e engenharia de prompt. Machine learning é uma técnica que usa, dentro de um espaço pré-definido de possibilidades, dados de entrada para criar as regras que serão usadas para coordenar o programa, orientado a partir de sinais positivos de feedback, permitindo com que o computador aprenda por análise e associação. Já a engenharia de prompt é um método baseado em usar scripts de códigos

(comandos) para guiar um dado programa de forma que ele aja conforme o esperado pelo usuário, sendo necessários conhecimentos básicos de programação e comportamento de máquina para aplicá-lo. O RAG, por ser uma ferramenta de linguagem natural que utiliza um banco de dados vetorial, diferente dos bancos convencionais que apenas buscam palavras exatas, transforma os dados em vetores (embeddings) e, com isso, é capaz de analisar o contexto das informações para encontrar as respostas mais relevantes, oferecendo resultados mais precisos e coerentes com a pergunta do usuário. Desta forma, combinando esses elementos, o programa interage com o usuário de forma natural e coesa, como se as interações fossem feitas diretamente com a pasta de textos (FEIJÓ, et al, 2009; SANTANA, et al, [s.d.]).

## 2.2 Justificativa

Existem certas barreiras que limitam a elaboração de uma pesquisa, dentre elas estão principalmente: a dificuldade de se consultar diversas fontes simultaneamente para conseguir estudar um determinado assunto, e a demasiada quantidade de tempo que se gasta fazendo isso. É neste contexto que veio a proposta do programa: ser uma ferramenta de estudos de textos usando um chatbot interativo, de forma que dê a impressão de que o usuário está interagindo com o próprio texto. Em tal sistema é possível pedir resumos, explicações de trechos específicos, e respostas/perguntas sobre um assunto como forma de testar seus conhecimentos, tornando o estudo mais rápido e eficiente e eliminando assim a possibilidade do sistema passar desinformação para o usuário, o que é uma grande ajuda na hora de se realizar uma pesquisa ou analisar um texto, uma vez que a fonte das respostas são os próprios arquivos fornecidos.

Desta forma, tal programa se mostra útil a todos os estudantes, pesquisadores, docentes, ou quaisquer pessoas que precisam trabalhar com diversos textos ao mesmo tempo, porém não dispõem de muitas horas para tal, ou simplesmente para agilizar e facilitar o trabalho com tais arquivos. Sendo assim, ao invés de desistir da tarefa, tal pessoa usaria o programa como uma alternativa viável para sua rotina, tornando a área de pesquisa um campo mais inclusivo, ao invés de restringi-la apenas àqueles que possuem mais tempo e menos obrigações inadiáveis.

# 3 Objetivo geral

Propõe um sistema que permita ao usuário indicar uma pasta com textos, artigos e demais dados com os quais gostaria de resumir, “conversar”, estudar e interagir com eles. Tal sistema é baseado em IAs generativas que usam LLM e redes neurais artificiais para gerar uma janela de chat que permita ao usuário as interações com os documentos.

## 3.1 Objetivos específicos

Através de metodologias citadas anteriormente, implementa-se um backend eficiente, utilizando ferramentas de programação moderna, com apoio de interfaces *low-code*, para estruturar um sistema capaz de:

- Armazenar documentos em PDF enviados pelo usuário;
- Realizar o particionamento (split) dos textos;
- Transformar esses fragmentos em embeddings vetoriais;
- Indexar os dados para posterior recuperação com base em perguntas do usuário através do RAG;
- Retornar os trechos mais relevantes conforme o conteúdo dos documentos enviados previamente.
- Utilizar engenharia de prompt para assegurar que a IA possui um comportamento esperado e previsível.

E desenvolve-se um frontend responsivo e intuitivo, utilizando React, JavaScript e CSS, que permita ao usuário:

- Fazer upload e acesso aos PDFs de forma simples;
- Interagir com os documentos por meio de uma interface de chat semelhante a aplicativos já familiares;
- Acessar a aplicação via um link web local com sistema de autenticação, que armazene os dados de usuário e seus documentos de forma segura.
- Proporcionar ao autor do projeto uma base sólida de aprendizado prático nas áreas de desenvolvimento web (frontend e backend), integração

com modelos de inteligência artificial e boas práticas de arquitetura de software.

# 4 Metodologia

## 4.1 Revisão da literatura

Durante o projeto, foram consultadas diversas fontes para embasar os principais conceitos que compõem a estrutura da aplicação proposta, para que assim ela pudesse ser desenvolvida. Um dos principais focos foi entender o funcionamento e as vantagens do uso do RAG (Retrieval-Augmented Generation), técnica que permite combinar o poder de geração de texto dos modelos de linguagem com uma base de dados externa confiável. No artigo “Understanding RAG: Part I – Why It’s Needed”, Brownlee (2024) explica como esse método surgiu como resposta às limitações dos modelos que operam apenas com dados do treinamento original, ajudando a evitar as chamadas “alucinações”.

Para que o sistema possa compreender documentos fornecidos pelo usuário, estudou-se também a criação de bancos de dados vetoriais, capazes de armazenar *embeddings* dos textos e associar perguntas a trechos semanticamente próximos. Isso garante maior precisão na geração de respostas. Artigos como os de Amatriain (2024) e Sahoo et al. (2024) foram fundamentais para compreender o papel da engenharia de prompt e sua aplicação na personalização da interação com os dados.

Para fundamentar teoricamente o projeto, foi realizada a leitura do eBook *“Como Utilizar IA Generativa na Pesquisa Acadêmica: Da Revisão de Literatura à Publicação”*, de Harrison S. Santana. Essa obra contribuiu para consolidar o conhecimento sobre o comportamento de modelos de inteligência artificial e técnicas de engenharia de prompt, com o objetivo de garantir que a experiência do usuário com o sistema desenvolvido seja satisfatória e alinhada com os propósitos originalmente definidos no projeto.

As leituras realizadas serviram como base teórica para o desenvolvimento do do projeto, especialmente na estruturação do backend com Langflow e Ollama, que facilitam a integração entre os modelos de linguagem e a base vetorial.

### 4.2 Desenvolvimento do Aplicativo

A fase inicial do desenvolvimento concentrou-se na instalação, configuração e familiarização com as ferramentas principais escolhidas para a construção do sistema: Langflow e Ollama.

O Langflow é uma plataforma *low-code* que permite a criação visual de fluxos de backend utilizando blocos funcionais interconectados. Essa abordagem elimina a necessidade de codificação robusta, ao mesmo tempo em que oferece flexibilidade e controle sobre o comportamento da inteligência artificial, já que possui blocos para engenharia de prompt. No projeto, o Langflow foi utilizado para estruturar a lógica de funcionamento da IA, incluindo o fluxo de entrada de dados, pré-processamento de arquivos PDF, geração de embeddings e recuperação de informações com base nas perguntas feitas pelos usuários, além do teste para verificar o funcionamento adequado da IA.

Paralelamente, foi integrado o Ollama, uma ferramenta para execução local de modelos de linguagem natural (LLMs). Essa escolha se deu por sua facilidade de configuração, por permitir a personalização de comportamentos por meio de engenharia de prompt e por ser uma solução gratuita, dispensando o uso de servidores externos ou APIs comerciais.

Após a validação da estrutura funcional da IA (capaz de processar PDFs fornecidos pelo usuário e responder com base no conteúdo desses documentos) teve início o desenvolvimento do frontend.

Essa etapa foi realizada por meio da clonagem de uma estrutura inicial em React, utilizando o ambiente de desenvolvimento Visual Studio Code (VS Code).

O React é uma biblioteca JavaScript voltada para a construção de interfaces de usuário dinâmicas e reativas. Com base em tutoriais e documentações disponíveis, foram desenvolvidos dois principais painéis de navegação:

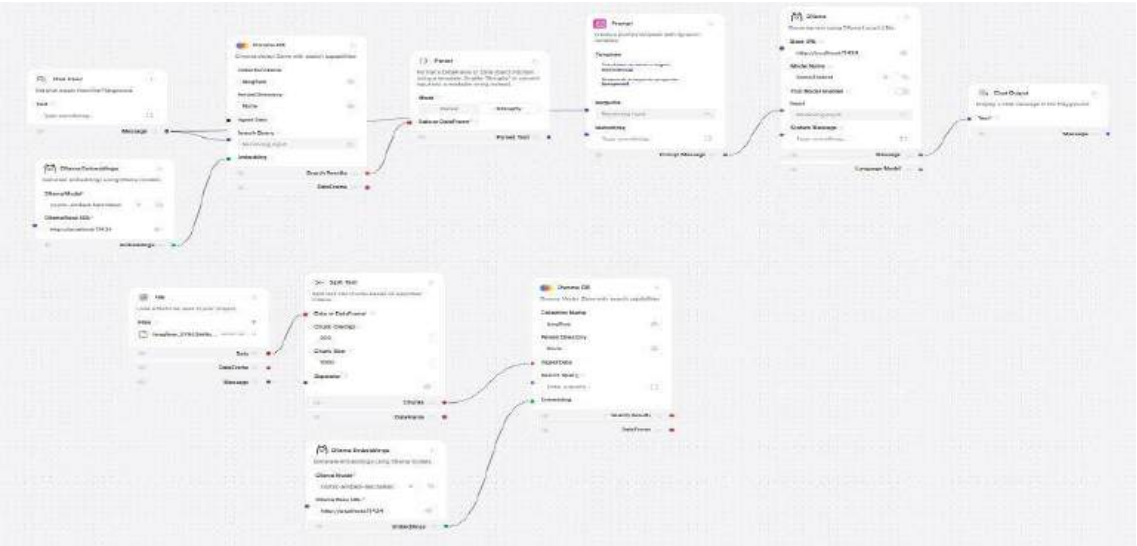
1. Um painel de login/cadastro, responsável por garantir a persistência de dados do usuário;

2. Um painel de interação, com a interface de chat para comunicação com a IA, além de funcionalidades de upload de arquivos PDF.

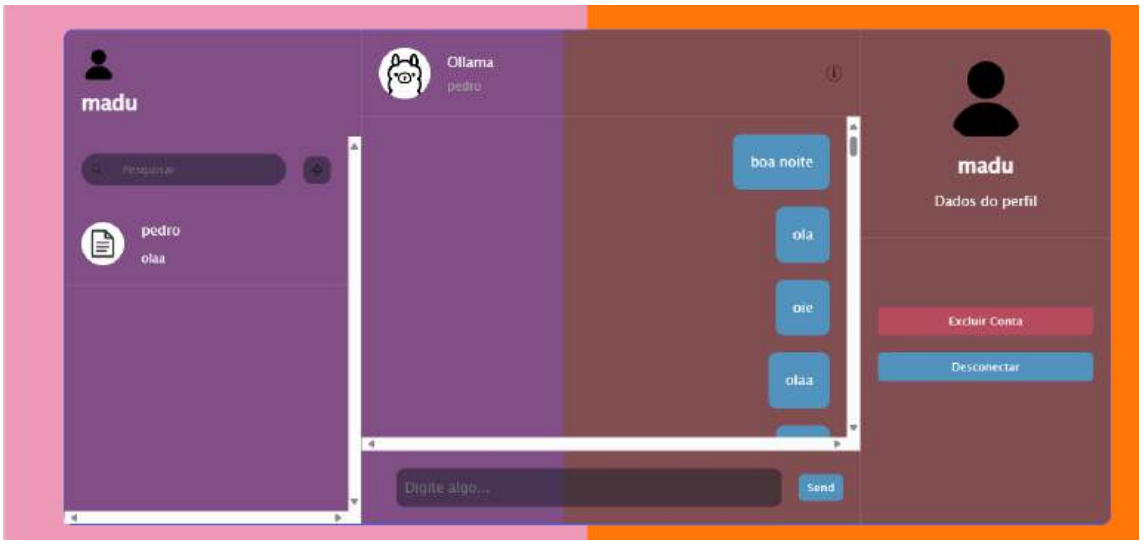
O sistema conta com os recursos essenciais esperados em uma aplicação web moderna, incluindo: login, logout, exclusão de conta, upload de documentos, criação de novos chats, alteração de nome de usuário, entre outros.

Como resultado, foi desenvolvido um aplicativo funcional, com interface intuitiva e fluxo de uso semelhante ao de plataformas de mensagens já conhecidas, permitindo que o usuário interaja de maneira natural com seus próprios textos por meio da inteligência artificial.

**Fluxo no backend do Langflow:**



**Interface no frontend:**


Link: https://primeiro-projeto-nu-cyan.vercel.app/

### 4.3 Interação com o usuário

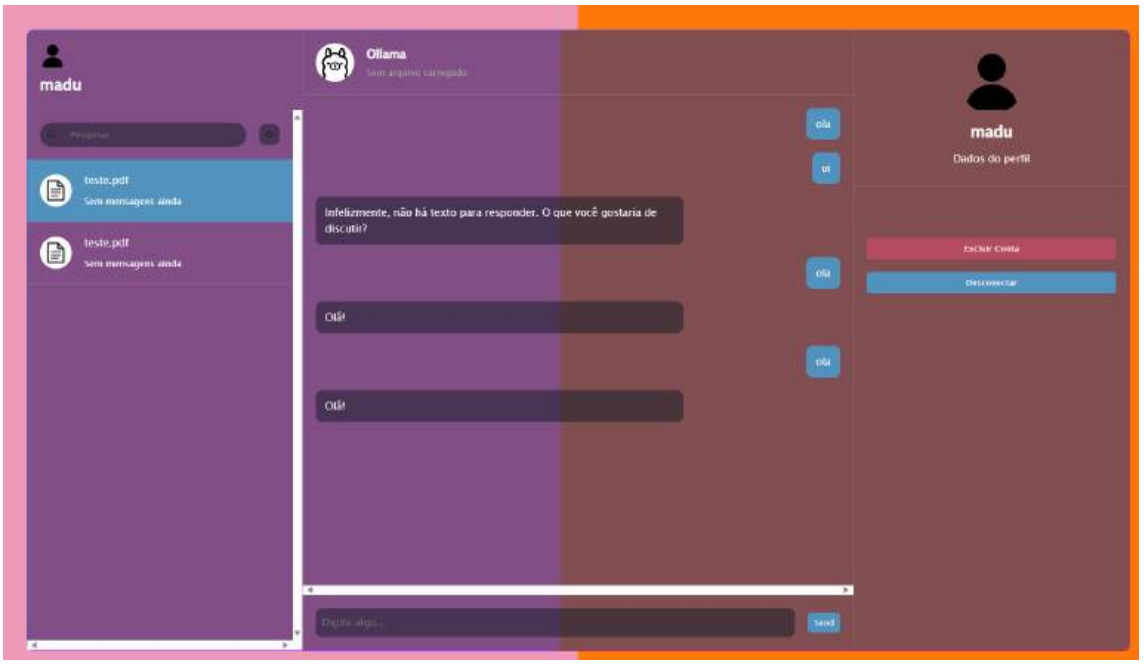
Com o programa já estruturado, nesta etapa ocorrem as interações robô / ser humano, que é dividida em duas etapas: fase de criação do banco de dados e fase de interação com o usuário. A fase de criação do banco de dados é aquela na qual o usuário vai acessar o programa, criar uma conta e fazer o upload de arquivos no formato PDF sobre o assunto em que esteja interessado. Cada assunto tem uma sessão própria, semelhante à interface do chat GPT, ou seja,

suas conversas ficam registradas no banco de dados do programa após este acessar os PDF's, o que também permite testar a capacidade do programa em selecionar os artigos corretos, mesmo com documentos de vários assuntos em sua interface. Uma vez selecionada a pasta, o programa a acessa, abre os arquivos, separa o texto em trechos menores, criando embeddings que ficam armazenadas no banco de dados vetorial.

Na fase de interação com o usuário, o programa recebe algum comando, como por exemplo: resuma estes arquivos em um único texto. Ou recebe perguntas específicas sobre o assunto ao qual pertencem os textos usados para criar a base de dados vetorial. A IA joga a pergunta no espaço vetorial em que foram registradas as Embeddings anteriores e analisa de qual embedding a pergunta ficará mais próxima. A partir disso, ele associa aquela Embedding a uma resposta adequada à pergunta feita e entrega ao usuário uma resposta concisa, e após isso resume essas respostas em um único texto, gerado por um modelo LLM.

Quaisquer outros comandos (que devem ser feitos com base nas pastas) são acatados de forma similar, com a IA sempre se baseando nesse banco de dados.

**Interação com a IA:**



### 4.4 Engenharia de prompt

A Engenharia de Prompt é uma disciplina essencial no desenvolvimento de sistemas interativos baseados em Modelos de Linguagem de Grande Escala (LLMs), como o ChatGPT. No contexto do projeto proposto, que criou um sistema de chat interativo para interação com arquivos PDF locais, a engenharia de prompt desempenha um papel crucial na eficácia e precisão das respostas geradas pelo modelo.

Ao integrar técnicas de engenharia de prompt, como o uso de instruções claras e específicas, o sistema orienta o LLM a fornecer respostas mais relevantes e contextualmente apropriadas, minimizando erros e alucinações.

Por exemplo, ao solicitar um resumo de um artigo técnico, um prompt bem formulado, no estilo: “Resuma o artigo técnico abaixo em até 200 palavras, destacando: (1) o objetivo principal, (2) a metodologia utilizada, (3) os resultados mais relevantes e (4) as conclusões. Use linguagem clara e objetiva, adequada para um público acadêmico.” pode instruir o modelo a identificar e sintetizar os pontos-chave do texto, mantendo a precisão e a coerência. Caso o prompt fosse algo do tipo: “Resuma este artigo.” seria muito genérico, não define escopo nem formato da resposta, o que pode causar alguma incongruência.

Além disso, a engenharia de prompt permite a personalização das interações, adaptando as respostas do modelo às necessidades específicas do usuário. Isso é particularmente útil em ambientes acadêmicos e de pesquisa, onde a compreensão detalhada e precisa do conteúdo é fundamental.

Portanto, a aplicação de práticas eficazes de engenharia de prompt foi fundamental para o sucesso do projeto, garantindo que o sistema ofereça uma experiência de usuário eficiente, precisa e personalizada.

## 5 Materiais e métodos

### 5.1 Materiais Utilizados

Hardware:

- Computador com processador Ryzen 5, GPU RTX 3060 (12GB de VRAM), 16GB de RAM e armazenamento SSD de 250GB.
- Justificativa: A GPU foi essencial para o processamento eficiente de embeddings e execução local do modelo LLM, enquanto o SSD garantiu velocidade no carregamento e manipulação de arquivos PDF.

Software e Ferramentas:

- Langflow: Plataforma low-code para construção do backend, integrando fluxos de pré-processamento de PDFs, geração de embeddings e recuperação de informações via RAG.
- Ollama: Ferramenta para execução local de modelos LLM, selecionada por sua flexibilidade e capacidade de personalização via engenharia de prompt.
- React (JavaScript/TypeScript, CSS): Biblioteca para desenvolvimento do frontend, incluindo interfaces de upload de arquivos e chat interativo.
- Banco de Dados Vetorial: Armazenamento de embeddings gerados a partir dos textos particionados (ex.: FAISS ou Chroma).

Orçamento:

O custo estimado do projeto é de aproximadamente R$3.550,00, incluindo hardware e eventuais licenças de software. Esse valor serve como referência para interessados em replicar o projeto, oferecendo uma base de cálculo para o financiamento necessário. No meu caso específico, esse custo não se aplica, pois utilizei ferramentas e equipamentos disponibilizados gratuitamente pela universidade.

### 5.2 Procedimentos

Pré-processamento de Dados:

- Particionamento (Split): Divisão dos textos em trechos menores (ex.: parágrafos ou seções) para otimizar a geração de embeddings.
- Geração e Armazenamento de Embeddings:

- Transformação dos trechos textuais em vetores (embeddings) usando modelos como llama embeddings, por sua facilidade de uso e por serem alternativas gratuitas para o projeto.
- Indexação no banco de dados vetorial para recuperação semântica via RAG.

Integração do LLM Local:

- Instalação no terminal da máquina do Ollama com modelos como LLama 2 ou Mistral, ajustados no prompt do Langflow para evitar alucinações e garantir respostas contextualizadas.
- Aplicação de engenharia de prompt para orientar o modelo na geração de respostas precisas (ex.: instruções claras para resumos ou perguntas técnicas).

Desenvolvimento do Frontend:

- Criação de uma interface React com funcionalidades de:
- Upload de PDFs.
- Chat interativo (estilo mensageiro) com histórico de conversas.
- Autenticação de usuários (Firebase ou similar).

## 6 Resultados e Discussões

O sistema desenvolvido contou com os seguintes arquivos e componentes durante sua implementação e fase de testes:

No backend:

Foram utilizados arquivos em PDF contendo artigos científicos e materiais didáticos relacionados a machine learning e processamento de linguagem natural, com o objetivo de validar a capacidade do sistema de realizar resumos automáticos e responder a consultas baseadas no conteúdo dos documentos.

No frontend:

A interface foi construída utilizando React, com as seguintes dependências e recursos: Zustand,  para gerenciamento de estado global da aplicação, Firebase, para autenticação de usuários e armazenamento seguro de dados e documentos, e arquivos de imagem para ícones e elementos visuais.

Após a implementação, o sistema demonstrou capacidade minimamente satisfatória de leitura e interpretação de PDFs, geração de resumos e respostas a perguntas, validando a integração entre o frontend e o fluxo de IA. No entanto, observou-se que o prompt de comando ainda requer ajustes para melhorar a precisão e a fluência das respostas.

Durante os testes, foram identificadas as seguintes dificuldades:

- Limitação no tamanho dos textos: Textos muito longos exigem um particionamento mais refinado para evitar perda de contexto durante a geração de embeddings.
- Latência na resposta: Em hardware menos potente, o tempo de processamento para gerar embeddings e consultar o banco vetorial mostrou-se significativo, impactando a experiência do usuário.
- Dependência da qualidade do prompt: Respostas genéricas ou incompletas foram obtidas quando os prompts não eram suficientemente específicos, reforçando a importância da engenharia de prompt.
- Necessidade de manter o Langflow ativo: O funcionamento da IA depende da execução contínua do Langflow, o que pode representar uma limitação operacional em ambientes de produção.

Apesar dos desafios, o sistema mostrou grande potencial para atingir sua proposta inicial. Espera-se que, após os devidos ajustes, ele se torne uma ferramenta completa e eficiente, cumprindo com excelência o objetivo de facilitar a interação com documentos locais por meio de uma interface acessível e intuitiva. Futuros trabalhos poderão explorar otimizações no pré-processamento de texto, a adoção de modelos de LLM mais eficientes e melhorias na interface para suportar grandes volumes de documentos.

## 7 Conclusão e Agradecimentos

O desenvolvimento deste projeto representou um significativo desafio técnico e acadêmico, uma vez que envolveu a aprendizagem de ferramentas e conceitos complexos, como comandos Git, execução de terminal, desenvolvimento frontend em React e integração com modelos de IA a partir de um conhecimento inicial bastante limitado. Apesar das dificuldades, a evolução do trabalho foi notável, resultando em um sistema funcional que atende aos objetivos propostos de forma satisfatória.

O resultado final, embora ainda possa ser aprimorado em diversos aspectos, demonstra a viabilidade da proposta e a capacidade de construção de uma aplicação completa, integrando backend com RAG e frontend responsivo. A experiência adquirida ao longo do processo foi enriquecedora, tanto do ponto de vista técnico quanto pessoal, consolidando conhecimentos em programação, arquitetura de software e inteligência artificial.

Como melhorias futuras, destacam-se:

- Otimização do funcionamento da IA, reduzindo a dependência de ferramentas externas como o Langflow;
- Aprimoramento de performance, especialmente no processamento de textos longos e na redução de latência;
- Expansão das funcionalidades do frontend, incluindo suporte a imagens, emojis e maior interatividade;
- Inclusão de mais formatos de documento além do PDF.

Por fim, gostaria de agradecer ao meu orientador, Prof. Hugo Puertas de Araujo, pela orientação constante, pela oportunidade de crescimento e pelo apoio técnico e motivacional ao longo de todo o projeto. Sem sua mentoria, este trabalho não teria sido possível. O aprendizado obtido superou as expectativas e serviu como base sólida para minha formação na área de tecnologia.

# 8 Referências


CHOLLET, François. *Deep Learning with Python.* Manning. Nova Iorque. Estados Unidos da América. 2018.

DAS GRAÇAS, Maria.; MINAMI, Mário.; WILLEM, Pieter. Bases computacionais da ciência. São Paulo. Brasil. 2013.

FEIJÓ Bruno.; CLUA Esteban.; S. Flávio. Introdução à ciência da computação com jogos: aprendendo a programar com entretenimento. Elsevier. Rio de Janeiro. Brasil. 2009.

LIMA, Isaías.; A. Carlos.; A. Flávia. Inteligência artificial. p. 2-4. 2014.

O que são Large Language Models (LLMs)? Data Science Academy, 19 de jun. de 2023. disponível em: https://blog.dsacademy.com.br/o-que-sao-large-language-models-llms/. Acesso em: 29 de julho de 2024.

SANTANA, Harsson S. *Como Utilizar IA Generativa na Pesquisa Acadêmica: Da Revisão de Literatura à Publicação. Ferramentas, Estratégias e Ética para Potencializar sua Pesquisa com Inteligência Artificial*. 1. ed. [S.l.: s.n.], [2023?]. eBook. Arquivo PDF. Disponível em: computador pessoal. Acesso em: 27 de julho de 2025.

Lama Dev**.** *React Chat App Full Tutorial 2024: Realtime Chat Application Project with Firebase* [vídeo]. YouTube, 10 abr. 2024. Disponível em: https://youtu.be/domt_Sx-wTY. Acesso em: 27 jul. 2025.