André Franco Ranieri
Cesar Hideki Imai
David Varão Lima Bentes Pessoa
João Victor Dallapé Madeira

Aplicação de LLM com RAG para consulta de processos ambientais

São Paulo

2024

André Franco Ranieri
Cesar Hideki Imai
David Varão Lima Bentes Pessoa
João Victor Dallapé Madeira

Aplicação de LLM com RAG para consulta de processos ambientais

Documentação Inicial do Trabalho de Conclusão de Curso SamsAI para a disciplina de Laboratório de Engenharia de Software

Universidade Presbiteriana Mackenzie
Faculdade de Computação e Informática

Orientador: Lucas Cerqueira Figueiredo

São Paulo
2024

André Franco Ranieri

Cesar Hideki Imai
David Varão Lima Bentes Pessoa
João Victor Dallapé Madeira
Aplicação de LLM com RAG para consulta de processos ambientais/ André Franco Ranieri
Cesar Hideki Imai
David Varão Lima Bentes Pessoa
João Victor Dallapé Madeira. – São Paulo, 2024-
37p. : il. (algumas color.) ; 30 cm.

Orientador: Lucas Cerqueira Figueiredo

Tipo do Trabalho (TCC) – Universidade Presbiteriana Mackenzie
Faculdade de Computação e Informática, 2024.

1. Palavra-chave1: Large Language Model 2. Palavra-chave2: Retrieval Augmented Generation 2. Palavra-chave3: Direito ambiental I. Orientador: Lucas Cerqueira Figueiredo II. Universidade Presbiteriana Mackenzie. III. Faculdade de Computação e Informática. IV. Aplicação de LLM com RAG para consulta de processos ambientais

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

UC	Caso de Uso
LLM	Large Language Model
RAG	Retrieval Augmented Generation
IA	Inteligência Artificial

Sumário

1 Introdução

Em nosso país, o sistema judiciário desempenha um papel crucial para a sociedade: administrar a justiça. Sua função é garantir os direitos individuais, coletivos e sociais e resolver conflitos entre cidadãos, entidades e Estado (Tribunal de Justiça de São Paulo, 2024). Porém, há dois grandes desafios a superar quando se trata da eficiência processual e da velocidade com que os casos são solucionados.

O Conselho Nacional de Justiça (CNJ) publicou recentemente o Relatório Justiça em Números 2024. O documento afirma que existem quase 84 milhões de processos tramitando no Brasil, que só aumentam a cada ano que passa (Conselho Nacional de Justiça, 2024). Sendo assim, o poder Judiciário acaba sendo sobrecarregado pelo enorme volume de processos.

Além disso, devido a enorme burocracia presente no poder Judiciário, os processos se tornam mais lentos, exigindo análises e pesquisas minuciosas por parte dos juízes e advogados.

Portanto, a ideia deste projeto é desenvolver uma aplicação Large Language Model (LLM) com Retrieval Augmented Generation (RAG) capaz de unificar a origem de documentos jurídicos e gerar respostas baseadas nas informações dos documentos, que tenham proximidade com a informação desejada por meio de inteligência artificial. Dessa forma, seria possível tornar o procedimento de consulta e pesquisa de documentos mais ágil e menos exaustivo para os usuários "desafogando", assim, o poder Judiciário. Contudo, é importante destacar que este trabalho se limita à área do direito ambiental.

2 Definição da Demanda

2.1 Descrição do projeto e definição dos usuários

A fim de cumprir o objetivo deste projeto, será desenvolvida uma aplicação web, semelhante ao ChatGPT, que seja capaz de entregar ao usuário respostas contendo as informações de documentos judiciais necessários para o andamento de um processo (como legislações, acórdãos e súmulas) no âmbito do direito ambiental. Para diminuir as possíveis alucinações da Inteligência Artificial (IA), será aplicada a técnica RAG, em que se utilizará um banco de dados contendo estes documentos judiciais para que a IA o consulte. Sendo assim, os potenciais usuários de nossa aplicação são: estudantes de direito, juízes, desembargadores, procuradores e advogados.

2.2 Etapas para a construção da aplicação e critérios

1. Adquirir um conjunto de documentos jurídicos para o teste;
2. a) Deve ter documentos com temas ambientais;
   b) Deve conter acórdãos;
   c) Deve conter súmulas;
3. Escolher um LLM treinado (Sabiá-3, ChatGPT);
4. a) Deve ser priorizado o menor custo;
   b) Deve ter capacidade mínima de 4000 tokens por prompt;
5. Estudar um dos frameworks (LlamaIndex ou Langchain) para aplicação RAG;
6. Fazer a integração do LLM através do framework (por chave de API ou baixado localmente);
7. a) Deve demorar no máximo 20 segundos para gerar a resposta;
8. Fazer a integração da fonte externa (conjunto de documentos) através do framework;
9. Integração com um banco de dados fornecido pela AWS;
10. Testar o sistema;
11. Construção da interface web da aplicação;

12. Integração com a interface web;

13. Testar o sistema com a interface.

3 REQUISITOS DO PRODUTO

Neste capítulo, são apresentados os requisitos que foram levantados. Separamos os requisitos em duas tabelas, na Tabela 1, os funcionais e, na Tabela 2, os não funcionais. Em cada uma delas, cada linha corresponde a um requisito, enquanto a coluna "ID"indica seu código de identificação, "Requisito"contém uma breve explicação do mesmo, "Tipo de requisito"classifica-o (funcional, usabilidade, software, desempenho, dados escalabilidade) e, por fim, a coluna "Prioridade"exibe sua relevância. Os níveis de prioridade impostos são:

- **Essencial:** é o requisito que o sistema não consegue operar em sua ausência;
- **Importante:** é o requisitos que o sistema consegue funcionar em sua ausência, mas seu funcionamento fica comprometido;
- **Desejável:** é o requisito opcional, o sistema não é comprometido mesmo em sua ausência.

Tabela 1 – Requisitos funcionais

ID	Requisitos funcionais	Tipo de Requisito	Prioridade
[RF01]	O sistema deve armazenar informações presentes em acórdãos para consulta.	Funcional	Essencial
[RF02]	O sistema deve armazenar informações presentes em súmulas para consulta.	Funcional	Essencial
[RF03]	O sistema deve armazenar informações presentes em orientações jurisprudenciais para consulta.	Funcional	Essencial
[RF04]	O sistema deve atender palavras-chaves para a busca.	Funcional	Essencial
[RF05]	O sistema deve atender filtro por tipo de documento (legislação, acórdão, súmula, etc).	Funcional	Essencial
[RF06]	O sistema deve atender filtro por tribunal (STF, STJ, TJs, etc).	Funcional	Essencial
[RF07]	O sistema deve atender filtro por estado.	Funcional	Essencial
[RF08]	O sistema deve armazenar informações presentes em legislações para consulta	Funcional	Importante
[RF09]	O sistema deve armazenar informações presentes em sentenças para consulta..	Funcional	Importante
[RF10]	O sistema deve armazenar informações presentes em decisões monocráticas para consulta.	Funcional	Importante
[RF11]	O sistema deve armazenar informações presentes em decisões interlocutórias para consulta.	Funcional	Importante
[RF12]	O sistema deve armazenar informações presentes em despachos para consulta.	Funcional	Importante
[RF13]	O sistema deve atender filtro por nome do relator	Funcional	Importante
[RF14]	O sistema deve atender filtro por tipo de processo (trabalhista, penal, etc).	Funcional	Importante
[RF15]	O sistema deve atender filtro por data.	Funcional	Importante
[RF16]	O sistema deve elaborar modelos de peças jurídicas.	Funcional	Importante
[RF17]	O sistema deve retornar a identificação dos documentos com suas ementas resumidas.	Funcional	Importante
[RF18]	O sistema deve retornar a descrição das legislações usadas numa decisão.	Funcional	Importante
[RF19]	O sistema deve resumir os argumentos de uma decisão.	Funcional	Importante
[RF20]	O sistema deve resumir a conclusão de uma decisão.	Funcional	Importante
[RF21]	O sistema deve auxiliar na interpretação de trechos de texto	Funcional	Importante
[RF22]	O usuário deve poder ajustar o tema da interface (tema claro ou escuro)	Funcional	Importante
[RF23]	O usuário deve poder fazer o download dos documentos recuperados.	Funcional	Importante
[RF24]	O usuário deve poder visualizar o histórico de conversas.	Funcional	Importante
[RF25]	O sistema deve retornar a identificação dos documentos com os trechos que tenham proximidade com a busca.	Funcional	Desejável
[RF26]	O sistema deve ter a opção dos usuários enviarem feedback (sugestões de melhorias e correções).	Funcional	Desejável
[RF27]	O sistema deve conter recursos text-to-speech (texto para fala) e speech-to-text (fala para texto).	Funcional	Desejável
[RF28]	O sistema deve ter versão mobile (acessível no celular).	Funcional	Desejável
[RF29]	O usuário deve poder visualizar o histórico de conversas.	Funcional	Desejável
[RF30]	O usuário deve poder apagar e arquivar conversas	Funcional	Desejável

Fonte: Próprios autores

Tabela 2 – Requisitos não funcionais

ID	**Requisitos não funcionais**	**Tipo de Requisito**	**Prioridade**
[RNF01]	O sistema deve ser integrado com uma interface web simples com o prompt de entrada em destaque	Usabilidade	Essencial
[RNF02]	O sistema deve ser compatível com diferentes navegadores web (Mozilla, Chrome, Opera, etc.)	Software	Essencial
[RFN03]	O sistema deve utilizar apenas os dados com proximidade igual ou maior a 50% do texto de entrada e dar uma saída de "informações insuficientes" caso não tenha.	Dados	Essencial
[RNF04]	O sistema deve lidar com no mínimo 3 acessos simultâneos.	Escalabilidade	Essencial
[RNF05]	O sistema deve ser integrado com um serviço de banco de dados fornecido pela AWS para armazenar e consultar dados.	Dados	Essencial
[RNF06]	O sistema deve ser gerenciável para adição de novos dados de fontes externas.	Dados	Essencial
[RNF07]	O sistema deve ter login de contas para salvar as configurações e conversas.	Usabilidade	Essencial
[RNF08]	O código deve ser bem documentado, de modo a facilitar futuras atualizações e manutenção do sistema.	Software	Importante
[RNF09]	O sistema deve fazer uso de criptografia para proteger as conversas dos usuários.	Segurança	Importante
[RNF10]	O sistema deve ter um tempo de resposta de no máximo 20 segundos.	Usabilidade	Importante
[RNF11]	O sistema deve ser aprovado pelo PageSpeed Insights.	Desempenho	Importante
[RNF12]	O sistema deve ser responsivo, ou seja, que se adapta automaticamente ao dispositivo do utilizador, seja um desktop, tablet ou telemóvel.	Usabilidade	Importante
[RNF13]	O sistema deve evitar cores relacionadas ao daltonismo.	Usabilidade	Desejável

Fonte: Próprios autores

4 WIREFRAMES

Com base nos requisitos funcionais descritos no capítulo 2, elaboramos protótipos de baixa fidelidade (wireframes) das telas da aplicação.

4.1 Tela de Login

Assim que o usuário entrar no site, ele deverá efetuar o login com seu email de acesso e respectiva senha.

Figura 1 – Tela de Login

4.2 Tela inicial (iniciar conversa)

Após efetuar o login com sucesso, ele é convidado a iniciar uma nova conversa com a IA através de um chat, em que o usuário pode interagir escrevendo uma mensagem na caixa de entrada de texto no canto inferior da tela. Para enviar, ele deve clicar no ícone de enviar mensagem.

Além disso, é também possível fazer upload de arquivos e mandá-los clicando no "clips"à esquerda. Também, ao invés de escrever, pode-se falar por voz a mensagem a ser enviada clicando no microfone à direita.

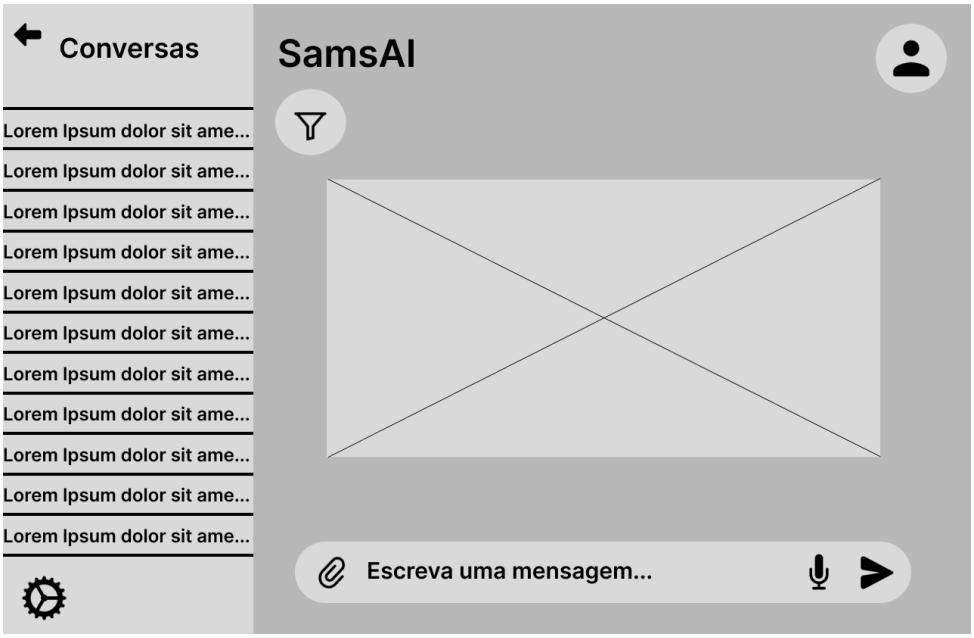

Figura 2 – Tela para iniciar uma conversa com o chatbot

4.3 Tela de conversa com a IA

Assim que o usuário envia uma mensagem ao modelo, ele retorna uma resposta, que pode ser ouvida clicando-se no ícone do alto-falante.

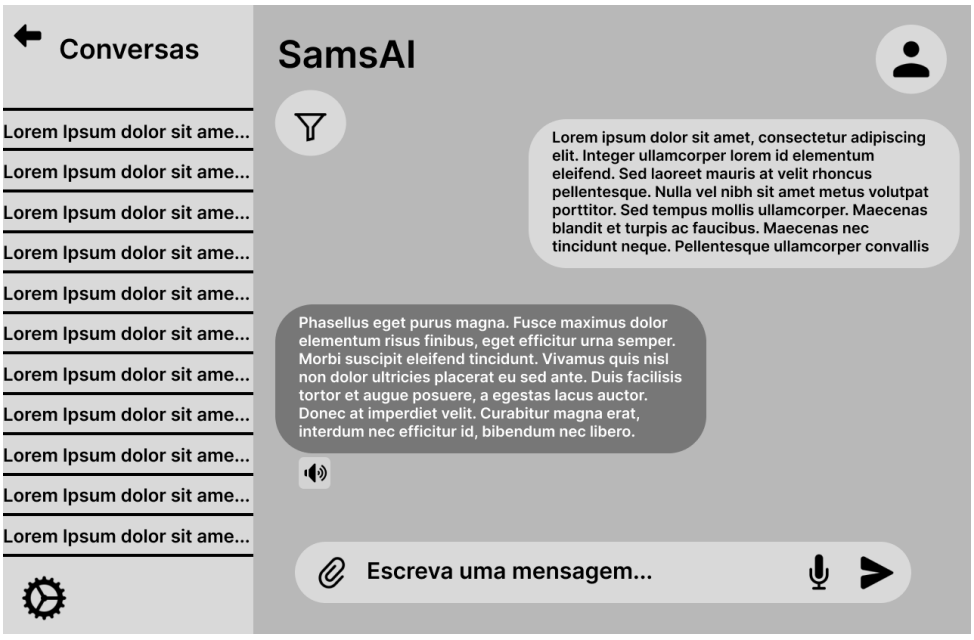

Figura 3 – Tela exibindo a conversa entre o usuário e a IA

4.4 Tela com filtro para respostas

Para obter respostas mais precisas, o usuário pode utilizar um filtro, clicando no ícone localizado no canto superior esquerdo. Nele, é possível filtrar os processos pelos: tipos documentos, os tipos de decisão, os tipos de processo, tribunais, estado, período, relator e advogado do caso.

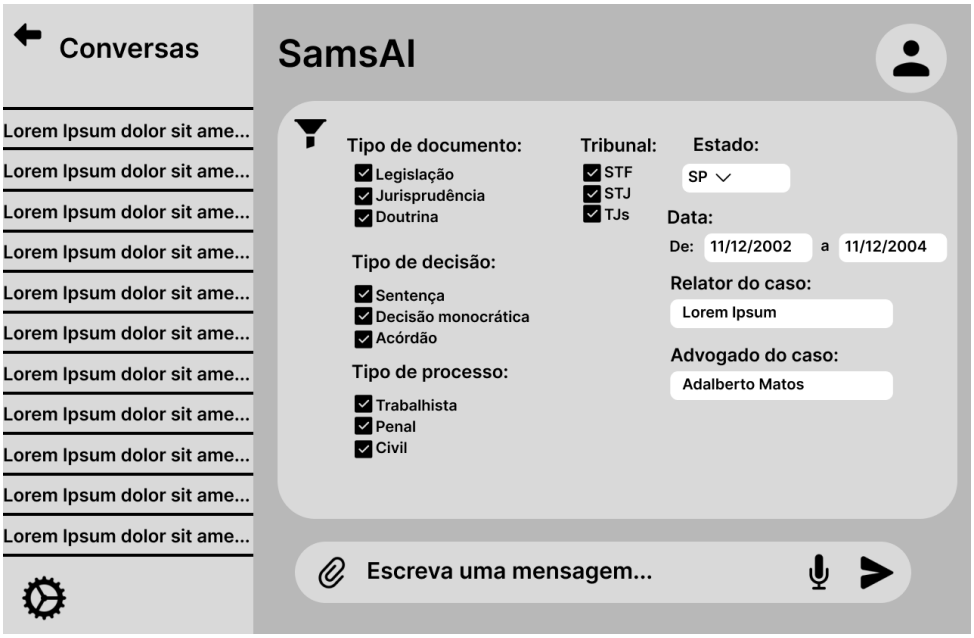

Figura 4 – Tela exibindo as opções de filtragem para pesquisa

4.5 Tela de configurações

Caso o usuário sinta necessidade de configurar a aplicação, ele pode acessar as configurações pelo ícone no canto esquerdo inferior da tela. Nesta tela, ele será capaz de alternar o tema da aplicação entre claro e escuro, além de poder descrever

Figura 5 – Tela para o usuário configurar a interface e as respostas da IA

4.6 Tela para dar feedback ou sair

O usuário pode, a qualquer momento, fornecer um feedback sobre sua experiência com o SamsAI ou sair da aplicação clicando em sua imagem de perfil e escolhendo uma destas opções.

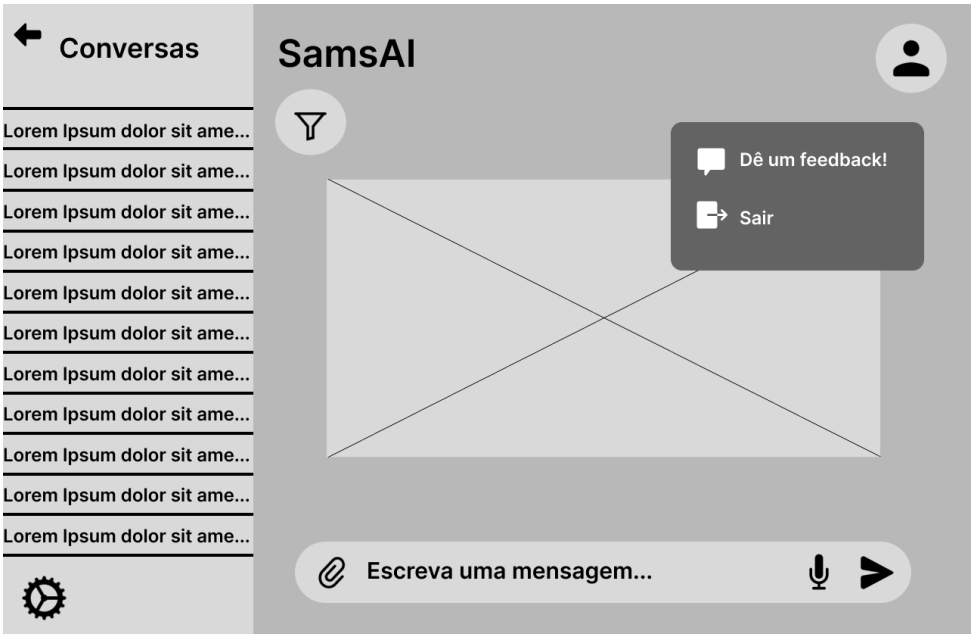

Figura 6 – Tela para dar um feedback ou sair da aplicação

5 MODELAGEM

Neste capítulo, a fim de aprofundar a interpretação acerca do projeto, apresentaremos: diagrama de casos de uso, especificações dos principais casos de uso, diagrama de classes de domínio e, por fim, um diagrama de sequência baseado nos principais casos de uso.

5.1 Diagrama de caso de uso

Neste diagrama, apresentamos os casos de uso que podem ser feitos pelo único ator existente, o usuário.

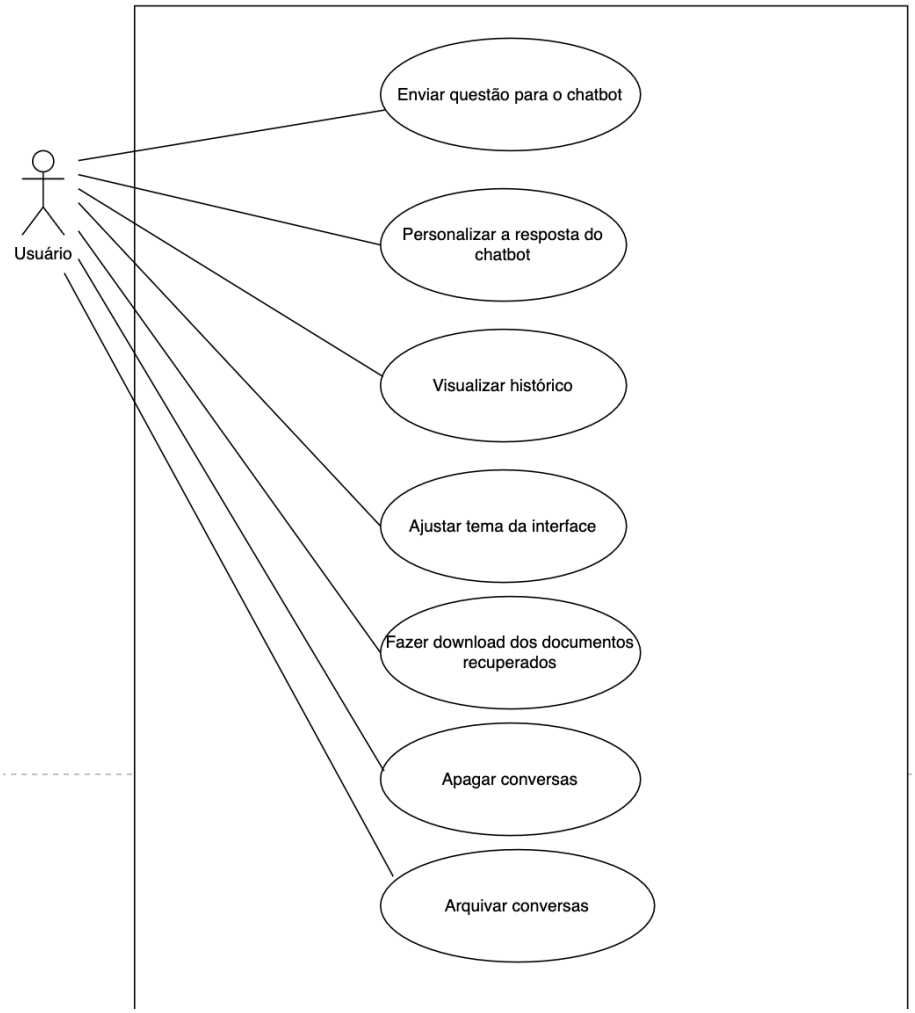

Figura 7 – Diagrama de casos de uso do projeto

5.2 Especificação dos principais casos de uso

Nesta subseção, apresentaremos a especificação dos principais casos de uso do projeto: enviar questão para o chatbot, ajustar interface e apagar uma conversa.

5.2.1 Enviar questão para o chatbot

Tabela 3 – Tabela do caso de uso enviar questão para o chatbot

Identificador	UC001
Nome	Enviar questão para o chatbot
Atores	Usuário
Sumário	O usuário envia uma questão ao sistema e este lhe responde com as informações solicitadas
Pré-condições	-
Pós-condições	Uma resposta é fornecida pelo sistema
Regras de Negócio	1) O texto inserido pelo usuário pode ter no máximo 4000 caracteres. 2) A resposta deve ser baseada em documentos com pelo menos 50% de proximidade.

Fonte: Próprios autores

Tabela 4 – Tabela de fluxo principal do UC001

Fluxo Principal	
Ações do ator	Ações do sistema
1) O usuário digita e envia um texto pelo prompt.	2) O texto aparece no chat, o prompt é bloqueado e o sistema gera uma notificação que a resposta está sendo gerada.
	3) O sistema faz a conversão da entrada de texto para uma representação numérica e consulta dados próximos da fonte externa.
	4) O sistema retorna os dados com maior proximidade e aplica no LLM.
	5) Com base nos dados recebidos, o LLM gera uma resposta embaixo do texto de entrada.
	6) O sistema indica a disponibilidade para uma nova questão.

Fonte: Próprios autores

Tabela 5 – Tabela de fluxo alternativo no passo 2 do UC001

Fluxo alternativo - Passo 2: O usuário quer interromper a geração de resposta	
Ações do ator	Ações do sistema
1) Entre os passos 1 e 6, o usuário clica na notificação para "parar de responder".	2) O sistema interrompe o processo de geração de resposta.
	3) O sistema indica a disponibilidade para uma nova questão.

Fonte: Próprios autores

Tabela 6 – Tabela de fluxo de exceção no passo 1 do UC001

Fluxo de exceção - Passo 1: Quantidade máxima de caracteres no prompt atingida	
Ações do ator	Ações do sistema
1) No passo 1, o usuário insere mais de 4000 caracteres no prompt	2) O sistema indica que a quantidade máxima de caracteres no prompt foi atingida e limita o texto até 4000 caracteres

Fonte: Próprios autores

Tabela 7 – Tabela de fluxo de exceção no passo 3 do UC001

Fluxo de exceção - Passo 3: Não há dados com pelo menos 50% de proximidade	
Ações do ator	Ações do sistema
1) No passo 1, o usuário insere um texto	2) O sistema não encontra dados com pelo menos 50% de proximidade com o texto de entrada
	3) O sistema instrui ao LLM gerar uma resposta de "informações não encontradas".
	4) O LLM gera a resposta embaixo do texto de entrada.
	5) O sistema indica a disponibilidade para uma nova questão.

Fonte: Próprios autores

5.2.2 Ajustar tema da interface

Tabela 8 – Tabela do caso de uso ajustar tema da interface

Identificador	UC002
Nome	Ajustar tema da interface
Atores	Usuário
Sumário	O usuário altera o tema da interface de escuro para claro ou de claro para escuro
Pré-condições	-
Pós-condições	O tema da interface se torna aquele definido pelo usuário
Regras de Negócio	1) Os dois únicos temas disponíveis para a interface são o claro e escuro

Fonte: Próprios autores

Tabela 9 – Tabela de fluxo principal do UC002

Fluxo Principal	
Ações do ator	Ações do sistema
1) Usuário clica no ícone de configurações	2) Sistema exibe a tela de configurações
3) Usuário muda o tema da interface para o tema desejado	4) Sistema adota o novo tema escolhido

Fonte: Próprios autores

Tabela 10 – Tabela de fluxo alternativo no passo 2 do UC002

Fluxo alternativo - Passo 2: O usuário desiste de trocar o tema	
Ações do ator	Ações do sistema
1) No passo 2, o usuário sai das configurações clicando no X no canto superior direito	2) Sistema retorna à tela com a conversa em que se encontrava

Fonte: Próprios autores

5.2.3 Apagar uma conversa

Tabela 11 – Tabela do caso de uso apagar uma conversa

Identificador	UC003
Nome	Apagar uma conversa
Atores	Usuário
Sumário	O usuário apaga uma conversa do sistema que considera inútil
Pré-condições	O usuário deve estar logado na aplicação
Pós-condições	A conversa deve estar excluída do banco de dados com as conversas daquele usuário
Regras de Negócio	1) Todas as conversas dos usuários devem ser armazenadas num banco de dados 2) O usuário deve ser capaz de apagar ou arquivar qualquer conversa a qualquer momento

Fonte: Próprios autores

Tabela 12 – Tabela de fluxo principal do UC003

Fluxo Principal	
Ações do ator	Ações do sistema
1) Usuário visualiza o histórico de conversas e escolhe uma das conversas para deletar	
2) Usuário passa o mouse sobre conversa	3) Sistema exibe ícone de mais opções ao lado do nome da conversa
4) Usuário clica sobre o ícone de opções	5) Sistema exibe opções de: Renomear, Arquivar ou Excluir uma conversa
6) Usuário escolhe opção de Excluir	7) Sistema pergunta se o usuário tem certeza de que deseja excluir aquela conversa
8) Usuário confirma que sim clicando em "Excluir"	9) Sistema exclui conversa de seu banco de dados

Fonte: Próprios autores

Tabela 13 – Tabela de fluxo alternativo no passo 6 do UC003

Fluxo alternativo - Passo 6: O usuário desiste de excluir a mensagem	
Ações do ator	Ações do sistema
1) No passo 6, o usuário desiste de clicar sobre o botão excluir	

Fonte: Próprios autores

Tabela 14 – Tabela de fluxo alternativo no passo 8 do UC003

Fluxo alternativo - Passo 6: O usuário desiste de excluir a conversa	
Ações do ator	Ações do sistema
1) No passo 6, o usuário desiste de clicar sobre o botão excluir	

Fonte: Próprios autores

5.3 Diagrama de classes de domínio

Nesta seção, exibimos a seguir o diagrama de classes de domínio contendo as classes identificadas no projeto (Usuário, Requisição, Modelo de IA, Codificador, Documento e Banco de Dados) e as relações entre elas.

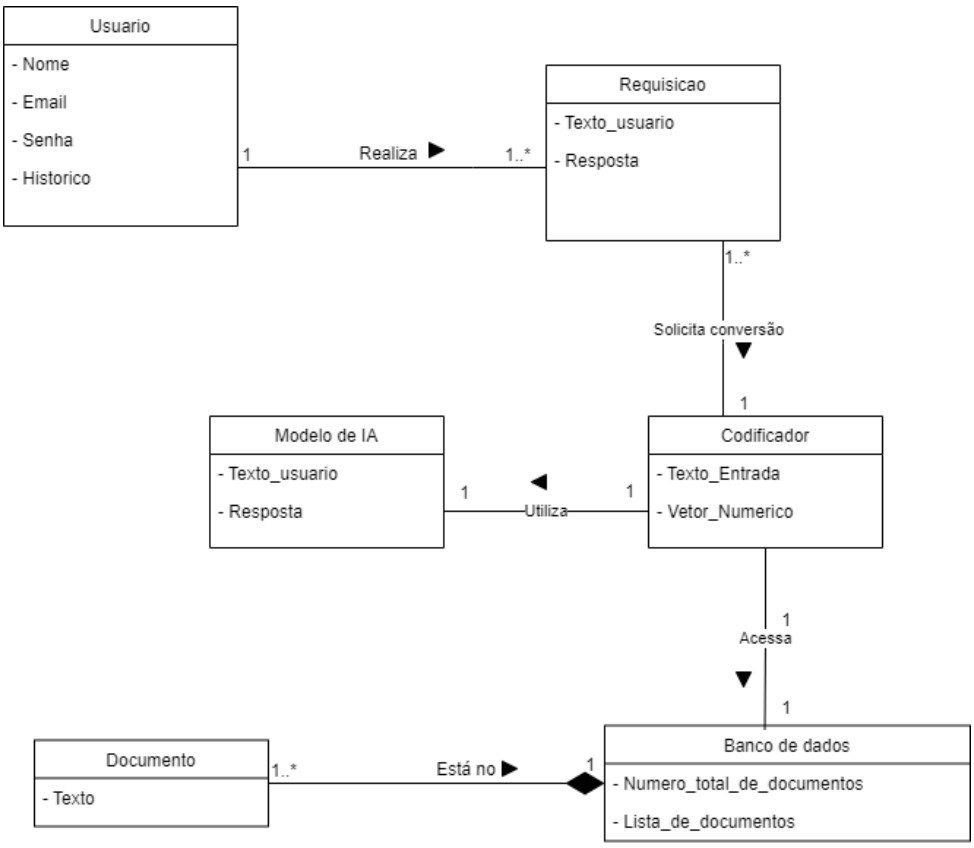

Figura 8 – Diagrama de classes de domínio

5.4 Diagrama de sequência

Nesta seção, exibimos a seguir o diagrama de sequência do caso de uso principal, enviar questão para o chatbot.

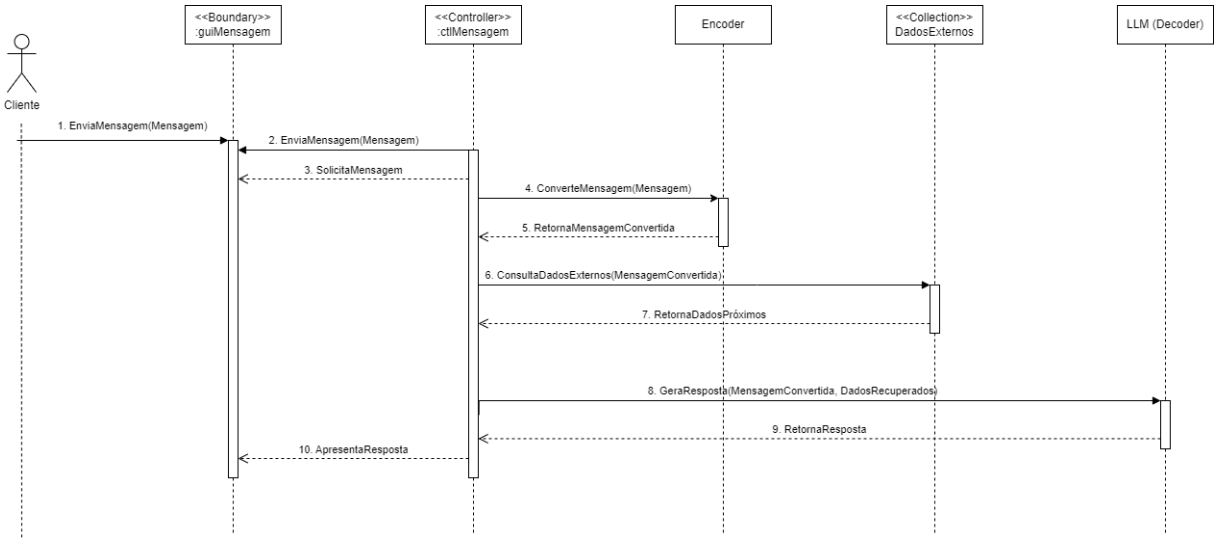

Figura 9 – Diagrama de Sequência

6 DESCRIÇÃO DA ARQUITETURA

6.1 Arquitetura em camadas

A arquitetura em camadas é composta por quatro níveis:

Camada de Apresentação: Responsável pela interação com o usuário, inclui o prompt de texto e a interfaces do chat.

Camada de Negócios: Gerencia a lógica de aplicação, incluindo o codificador para pré-processamento de dados, o decodificador para gerar respostas, histórico de conversas e configurações de interface.

Camada de Persistência: Cuida do armazenamento e recuperação de dados, utilizando um objeto para acesso de dados de documentos armazenados fora do sistema.

Camada de Dados: Envolve a gestão de dados, abrangendo bancos de dados, arquivos PDF e arquivos Markdown.

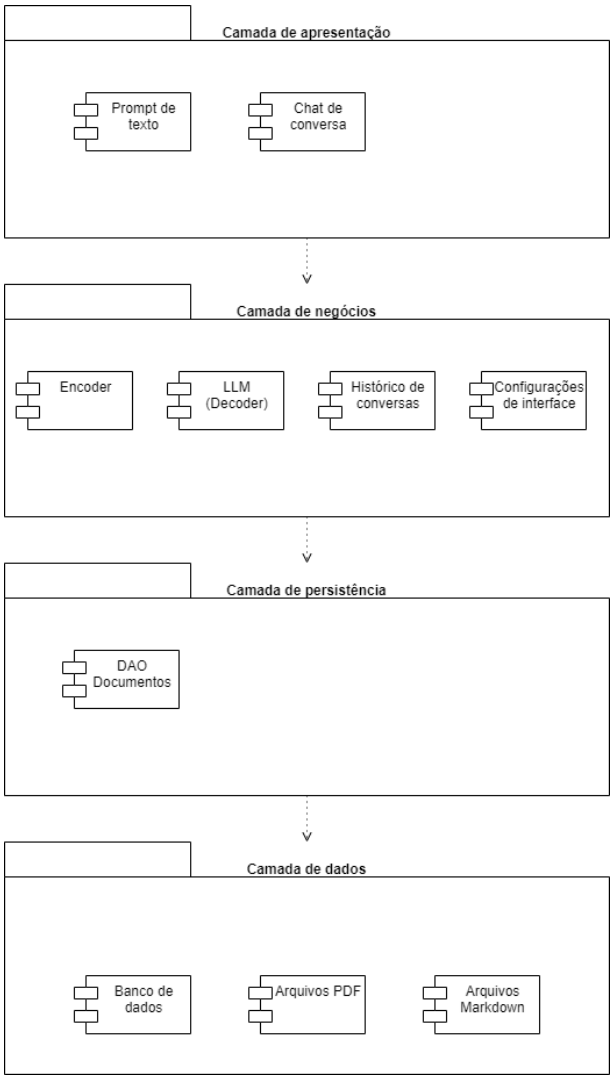

Figura 10 – Diagrama UML da arquitetura em camadas

6.2 Pipeline

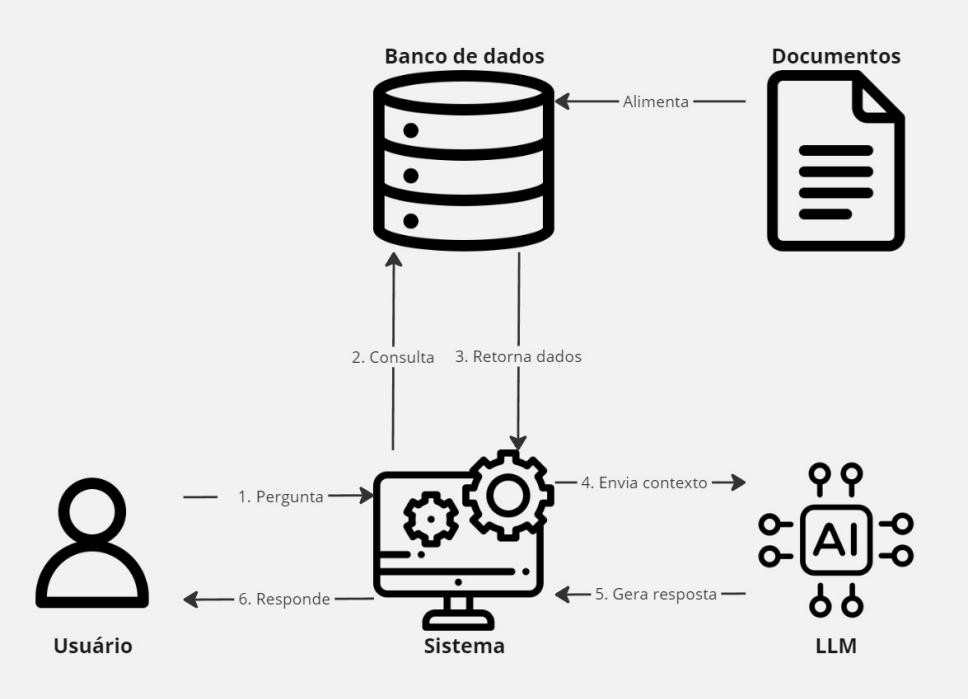

Figura 11 – Pipeline

6.3 Linguagens de Programação

A ideia de nosso projeto é construir o sistema em Python por meio do uso de frameworks adequados para desenvolvimento de aplicações com LLMs.Também será usado HTML/CSS e Javascript para construir a estrutura da interface da aplicação. Além disso, o Javascript também será usado para integrar a interface com o sistema.

6.4 Frameworks

6.4.1 LangChain

O LangChain é um framework para desenvolvimento de aplicações que utilizam LLMs. Ele pode ser usado para criar chatbots e agentes virtuais personalizados, analisar grandes volumes de texto para extração de informação e integração com diversas APIs.(LangChain, 2024a)

6.4.2 LangGraph

Já o LangGraph é uma extensão do LangChain focada para a construção de aplicações mais robustas com multi-agentes. Sua biblioteca permite a criação de sistemas

complexos com vários agentes que interagem entre si, cada um executando tarefas específicas e trocando informações conforme necessário. Além disso, oferece gerenciamento automático de estado, permitindo rastrear e manter informações ao longo de várias interações, garantindo que o sistema mantenha o contexto e responda adequadamente a novas entradas.(LangChain, 2024b)

6.4.3 LlamaIndex

O LlamaIndex é um framework para conectar fontes de dados personalizadas a LLMs. Sua biblioteca oferece ferramentas para ingerir, estruturar e consultar dados de fontes estruturadas e não-estruturadas para a aplicações com LLMs.(LlamaIndex, 2024)

7 DESENVOLVIMENTO

O desenvolvimento de nosso projeto foi dividido em duas grandes partes: o *front-end*, desenvolvido com *React App*, que foi depois dockerizado; e o *back-end*, desenvolvido com o auxílio da ferramenta *no-code Langflow*. Nas duas próximas seções, abordaremos o desenvolvimento de cada uma dessas partes, respectivamente.

7.1 *Front-end*

Inicialmente, no desenvolvimento do *front-end*, escolhemos fazer uma aplicação que utilizasse *React App* devido à familiaridade que temos com essa biblioteca do *javascript* e, principalmente, pela possibilidade que ela oferece de criar componentes fáceis de usar, reutilizar e excluir, o que facilita a manutenção do código.

A primeira tela que começamos a implementar foi a de Login, porém ela ainda não foi integrada ao sistema, pois pretendemos focar no desenvolvimento da tela de conversa com a IA bem como suas funcionalidades. Por esse motivo, começamos a desenvolvê-la iniciando pelo principal, o *chat* de conversas. Para desenvolver o *chat*, utilizamos a biblioteca *@chatscope/chat-ui-kit-react*, pois ela providencia o *input* para o usuário digitar e enviar as mensagens, além de exibí-las. Por falta de tempo para realizar este projeto, foi possível fazer somente essas duas telas.

O código em CSS (*Cascading Style Sheets*) para estilização do *chat* dessa biblioteca foi sobrescrito, a fim de deixá-lo semelhante ao protótipo realizado no Figma. Para diferenciar visualmente as mensagens da IA e do usuário, optamos por colorir as mensagens da IA com um tom de branco e as do usuário com um tom de vinho.

A fim de realizar a integração com o *back-end*, foi implementada a classe javascript *LangflowClient* fornecida pela própria plataforma do *Langflow*. Nela, é realizada uma requisição HTTP POST com o objetivo de obter a resposta da IA para a mensagem enviada pelo usuário. Embora a API (*Application Programming Interface*) fornecida pelo *Langflow* funcionasse corretamente, no início, enfrentamos problemas com o CORS (*Cross-Origin Resource Sharing*), pois nosso aplicativo web, que faz uso dessa API, só pode fazer solicitações para recursos de mesma origem da qual o aplicativo foi carregado, a menos que a resposta da outra origem inclua os cabeçalhos CORS corretos (Mozilla Developer Network Web Docs, 2024).

Para resolver este problema, foi construída uma *proxy* em javascript para "burlar"a política do CORS, uma vez que não havia como alterar os cabeçalhos HTTP das respostas da API no *back-end*. Desse modo, foi possível obter as respostas da IA e exibí-las nas

mensagens da interface do *front-end*.

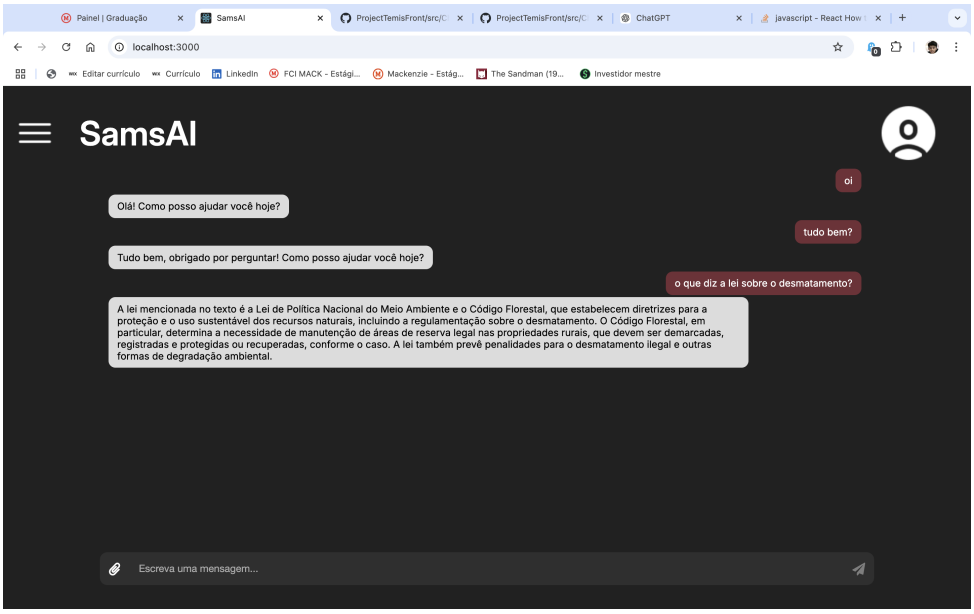

Figura 12 – Tela de conversa entre IA e o usuário

O código ainda estava funcionando apenas localmente, porém ainda era necessário transferí-lo para a nuvem. Após tentarmos utilizar diferentes serviços como o AWS Amplify e Vercel, mas sem sucesso, decidimos *dockerizar* a aplicação e movê-la para uma instância EC2 da AWS (*Amazon Web Services*). Embora a aplicação rodasse localmente nessa máquina, essa instância foi configurada para permitir o acesso de qualquer outro IP pela porta 80 (a porta que a aplicação usa para conectar o *front-end* com IP do usuário). Sendo assim, para acessar a aplicação, basta usar o IPv4 público da máquina virtual EC2: 52.45.15.19.

7.2 *Back-end*

Através da plataforma de desenvolvimento *low-code* Langflow, foi possível criar um sistema inicial e testar fluxos de trabalho usando os componentes já predefinidos da aplicação.

Para a construção da fonte externa de dados, foi selecionado quatro documentos públicos de jurisprudência no site do TJSP. Esses documentos foram segmentados em *chunks* com 1000 de tamanho e *overlap* de 200. Em seguida, para fazer os *embeddings* foi usado o modelo *encoder* NV-Embed-QA, que foi disponibilizado sem custo no Langflow. Depois, tanto os trechos de texto, quanto os respectivos *embeddings*, foram inseridos numa única coleção dentro de um banco de dados vetorial fornecido pela Astra DB, que também foi disponibilizado sem custo no Langflow.

O modelo Sabiá-3 disponibilizado pela Maritaca AI foi o *decoder* escolhido para gerar as respostas, devido ao seu bom desempenho e baixo custo.

8 RESULTADOS

A fim de avaliar o que foi desenvolvido até o momento, foi aplicado um teste de usabilidade através de questionários pré-teste e pós-teste para observar a realização de tarefas no sistema por parte dos usuários. Os usuários eram estudantes de direito e/ou profissionais na área.

O questionário pré-teste tinha o intuito de coletar apenas dados demográficos e informações sobre a experiência dos usuários com IAs como o ChatGPT, Copilot, Gemini, etc. Já o questionário pós-teste continha perguntas do SUS (*System Usability Scale*) a fim de avaliar a usabilidade do sistema. Além disso, ao seu fim, o segundo questionário continha um espaço em que os participantes do teste poderiam escrever o que acharam do sistema, mencionando pontos positivos e negativos.

Com os resultados obtidos pelo segundo questionário, pôde-se observar que a interface é intuitiva e fácil de se interagir, provavelmente isso se deve ao fato da interface ser bem simples contendo somente um *chat* entre a IA e o usuário. Contudo, os usuários também afirmaram que ainda é preciso alimentar o banco de dados com mais dados. Isso se deve ao fato de que o banco de dados foi alimentado com um volume muito pequeno de dados.

Também foi possível notar durante o desenvolvimento da aplicação que o modelo não tinha uma boa capacidade de memorizar mensagens enviadas anteriormente durante uma conversa, o que pode causar um impacto negativo

O código do *front-end* deste projeto pode ser acessado aqui.

9 CONCLUSÃO

Para a versão final do projeto, o *back-end* da aplicação deverá ser desenvolvido em Python, de forma que reduza as dependências por serviços de terceiro. A linguagem escolhida dispõe de bibliotecas e *frameworks* para desenvolvimento de aplicações de LLMs, como o Langchain e Langgraph. A arquitetura do projeto também deverá ser revisada, de modo que transpareça os dados usados nas consultas e seja escalável para uma base maior de dados. Desse modo, será possível solucionar o problema, apontado pelos usuários, de poucos dados disponíveis para consulta, o que fará a aplicação atingir seu objetivo proposto.

A interface do *front-end* conseguiu se mostrar agradável para os usuários. No futuro, pretende-se adicionar mais funcionalidades a aplicação, como as opções de: criar diferentes conversas, aplicar filtros, alternar entre modo claro e escuro, personalizar as respostas, poder enviar arquivos à IA, poder realizar um *feedback*, conforme planejado no início do projeto. Além disso, pretende-se adicionar outras funções que garantam a boa usabilidade do sistema, como a possibilidade de escutar as mensagens recebidas da IA por áudio e de dar comandos por voz.

Referências

Conselho Nacional de Justiça. *Relatório de atividades de 2024*. Brasília: [s.n.], 2024. Acesso em: 10 set. 2024. Disponível em: <https://www.cnj.jus.br>. 9

LangChain. *LangChain Introduction*. 2024. Acesso em: 11 set. 2024. Disponível em: <https://python.langchain.com/v0.2/docs/introduction/>. 30

LangChain. *LangGraph Overview*. 2024. Acesso em: 11 set. 2024. Disponível em: <https://langchain-ai.github.io/langgraph/>. 31

LlamaIndex. *LlamaIndex Introduction*. 2024. Acesso em: 11 set. 2024. Disponível em: <https://docs.llamaindex.ai/en/stable/#llamaindex-is-the-framework-for-context-augmented-llm-applications>. 31

Mozilla Developer Network Web Docs. *Cross-Origin Resource Sharing (CORS)*. 2024. Acesso em: 2 nov. 2024. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/CORS>. 32

Tribunal de Justiça de São Paulo. *Função do Poder Judiciário e Órgãos da Justiça*. 2024. Acesso em: 10 set. 2024. Disponível em: <https://www.tjsp.jus.br/PoderJudiciario/PoderJudiciario/OrgaosDaJustica>. 9