INSTITUTO FEDERAL
Rondônia

MINISTÉRIO DA EDUCAÇÃO
Secretaria de Educação Profissional e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia

Campus Vilhena

Coordenação do Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas

GUILHERME KRAUSE RAMOS

Chatbot para consulta em documentos jurídicos

Vilhena – RO
2025

GUILHERME KRAUSE RAMOS

Chatbot para consulta em documentos jurídicos

Artigo entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor mestre José Lucas Brandão Montes.

Vilhena – RO
2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Ramos, Guilherme Krause.
Chatbot para consulta em documentos jurídicos / Guilherme Krause Ramos. - Vilhena, 2025.
27 f. : il.

Orientador(a): Prof. Me. Jose Lucas Brandao Montes.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Vilhena, 2025.

1. Chatbot. 2. RAG. 3. Inteligencia artificial. 4. Jurídico. I. Montes, Jose Lucas Brandao (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Rosilene Maria do Couto Marques, CRB-11/321

GUILHERME KRAUSE RAMOS

Chatbot para consulta em documentos jurídicos

Artigo entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor mestre José Lucas Brandão Montes.

Aprovado em 25/11/2025 pela banca examinadora.

gov.br
Documento assinado digitalmente
MARCO ANTONIO AUGUSTO DE ANDRADE
Data: 02/12/2025 17:27:50-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Marco Antônio Augusto de Andrade
Examinador interno

gov.br
Documento assinado digitalmente
ROBERTO SIMPLICIO GUIMARAES
Data: 03/12/2025 00:29:18-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Roberto Simplício Guimarães
Examinador interno

gov.br
Documento assinado digitalmente
JOSE LUCAS BRANDAO MONTES
Data: 03/12/2025 15:28:52-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre José Lucas Brandão Montes
Orientador

Resumo. *Este trabalho apresenta o desenvolvimento de um chatbot para consulta em documentos jurídicos, utilizando técnicas de Inteligência Artificial, Processamento de Linguagem Natural (PLN) e a arquitetura Retrieval-Augmented Generation (RAG). Considerando o cenário brasileiro de alta demanda judicial, grande volume de processos e crescente complexidade normativa, o projeto busca demonstrar a viabilidade de uma solução acessível e baseada em tecnologias de código aberto e planos gratuitos para apoiar profissionais do Direito. O sistema foi estruturado em uma arquitetura de três camadas, composta por um backend desenvolvido em Python com FastAPI, um frontend em JavaScript com ReactJS e uma camada de serviços externos integrando modelos de linguagem, bancos vetoriais e APIs especializadas. Ao longo do desenvolvimento, adotou-se uma abordagem iterativa e incremental, com organização via Kanban, além de um fluxo de automação com CI/CD, conteinerização com Docker e deploy em uma máquina virtual disponibilizada pela instituição. O chatbot permite o upload de documentos PDF, extração de texto, divisão do texto, geração de embeddings, indexação no ChromaDB e recuperação semântica para respostas fundamentadas. Foram implementados mecanismos de autenticação, controle de sessões, histórico persistente, filtragem por documento, reranqueamento e integração com modelos LLM de alta performance hospedados na Groq Cloud. Os resultados indicam que o sistema é funcional, escalável e capaz de responder de forma coerente e contextualizada, ainda que apresente desafios relacionados à precisão semântica e ao desempenho. Como trabalhos futuros, propõe-se a inclusão de novos modelos de linguagem, otimizações no pipeline RAG, testes de desempenho, suporte ao envio de documentos em lote e aprimoramentos na robustez arquitetural.*

Palavras-chave: *Chatbot; RAG; Inteligência artificial, Jurídico.*

Abstract. *This work presents the development of a chatbot designed for querying legal documents, employing Artificial Intelligence techniques, Natural Language Processing (NLP), and the Retrieval-Augmented Generation (RAG) architecture. Considering the Brazilian context of high judicial demand, large volume of cases, and increasing normative complexity, the project aims to demonstrate the feasibility of an accessible solution based on open-source technologies and free-tier services to support legal professionals. The system was structured using a three-layer architecture composed of a Python backend with FastAPI, a JavaScript frontend built with ReactJS, and an external services layer integrating language models, vector databases, and specialized APIs. Throughout the development process, an iterative and incremental approach was adopted, supported by Kanban-based task management, CI/CD automation pipelines, Docker containerization, and deployment on a virtual machine provided by the institution. The chatbot enables PDF upload, text extraction, text splitting, embedding generation, semantic indexing with ChromaDB, and contextual retrieval for grounded answers. Additional features include authentication mechanisms, session control, persistent conversation history, document-based filtering, reranking, and integration with high-performance LLMs hosted on Groq Cloud. The results indicate that the system is functional, scalable, and capable of generating coherent and context-aware responses, although challenges remain related to semantic accuracy and performance optimization. Future work includes incorporating new language models, improving the RAG pipeline, conducting performance tests, enabling batch document processing, and strengthening the system's architectural robustness.*

Keywords: *Chatbot; RAG; Artificial intelligence; Legal.*

1. Introdução

A prática jurídica no Brasil enfrenta desafios significativos devido ao elevado volume de processos e à complexidade normativa. De acordo com o Supremo Tribunal Federal, no ano de 2023 o sistema judiciário brasileiro recebeu aproximadamente 35 milhões de novos processos, um aumento de 9,4% em relação ao ano anterior, totalizando cerca de 84 milhões de processos em tramitação (SUPREMO TRIBUNAL FEDERAL, 2023). Esse cenário sobrecarrega advogados, juízes e servidores, tornando o trabalho jurídico cada vez mais exaustivo e propenso a erros.

O Brasil possui uma das maiores concentrações de advogados do mundo, com cerca de 1,3 milhão de profissionais registrados em 2023 (OAB). Apesar desse número expressivo, muitos advogados enfrentam desafios relacionados à remuneração, com uma parcela significativa recebendo menos de cinco salários mínimos mensais (OAB). Além disso, a grande quantidade de processos judiciais e a necessidade de constante atualização sobre legislações e jurisprudências tornam o trabalho jurídico ainda mais desafiador.

Nesse contexto, a tecnologia tem se mostrado uma aliada importante na modernização do setor jurídico. Em 2023, o mercado brasileiro de tecnologia jurídica movimentou R$ 2,8 bilhões, evidenciando um crescimento de 35% em relação ao ano anterior (Barbieri Advogados). Ferramentas como softwares de gestão, assinatura digital e plataformas de pesquisa jurídica têm sido amplamente adotadas para otimizar processos e reduzir custos. No entanto, atividades que exigem interpretação de textos complexos, como a análise de contratos e pareceres, ainda dependem majoritariamente da intervenção humana.

Os *chatbots*, sistemas baseados em Inteligência Artificial (IA) e Processamento de Linguagem Natural (PLN), surgem como uma solução promissora para auxiliar na consulta e interpretação de documentos jurídicos. Esses sistemas são capazes de interagir com usuários em linguagem natural, compreender perguntas contextuais e fornecer respostas precisas com base em grandes volumes de texto. No entanto, o desenvolvimento de um chatbot eficiente, com capacidade de interagir sobre assuntos e documentos jurídicos, requer conhecimento técnico em diversas áreas, como IA, PLN e integração de sistemas, além de um planejamento cuidadoso para garantir sua eficácia e confiabilidade.

Diante do contexto apresentado, este artigo propõe o desenvolvimento e a avaliação de um chatbot voltado à consulta de documentos jurídicos, utilizando tecnologias de código aberto, recursos de inteligência artificial e serviços com planos gratuitos, com objetivo de demonstrar a viabilidade de uma solução acessível e eficaz para apoiar a prática jurídica, contribuindo para a democratização do acesso à informação e redução da sobrecarga de trabalho dos profissionais da área.

2. Fundamentação Teórica

2.1. Inteligência Artificial: conceitos e evolução

A Inteligência Artificial (IA) vem se consolidando como uma das áreas mais dinâmicas e influentes da ciência contemporânea, com impacto direto em diversos setores da sociedade. O termo foi utilizado pela primeira vez em 1956, durante a Conferência de Dartmouth, por John McCarthy, que definiu a IA como a ciência e engenharia de produzir máquinas inteligentes capazes de realizar tarefas que, até então, dependiam da inteligência humana (McCARTHY, 1956). Desde então, a IA evoluiu de sistemas baseados

em regras e raciocínio simbólico para modelos complexos que aprendem a partir de dados e experiências, transformando a forma como interagimos com a informação (RUSSELL; NORVIG, 2021).

Nas décadas seguintes, o desenvolvimento de algoritmos de aprendizado de máquina (machine learning) e redes neurais artificiais revolucionou o campo. Esses sistemas são capazes de identificar padrões em grandes volumes de dados e tomar decisões baseadas em probabilidades e correlações, o que ampliou significativamente o escopo das aplicações da IA (GOODFELLOW; BENGIO; COURVILLE, 2016). O avanço da capacidade computacional e o acesso a bases de dados massivas impulsionaram uma nova era, marcada pelo surgimento do deep learning, técnica que utiliza múltiplas camadas de redes neurais para aprender representações complexas de informações (LECUN; BENGIO; HINTON, 2015).

Atualmente, a IA é parte integrante de tecnologias que permeiam o cotidiano, como sistemas de recomendação, reconhecimento de imagem, análise preditiva e veículos autônomos. No campo das comunicações e da informação, o impacto é ainda mais expressivo, especialmente com o avanço dos modelos de linguagem e do Processamento de Linguagem Natural (PLN), que buscam compreender e reproduzir a linguagem humana. Essa evolução permitiu a criação de sistemas interativos, como assistentes virtuais e *chatbots*, que simulam a comunicação humana com um alto grau de precisão e contexto (JURAFSKY; MARTIN, 2023).

Além dos avanços técnicos, a IA também tem despertado debates éticos e sociais. Questões como transparência algorítmica, privacidade de dados e responsabilidade sobre decisões automatizadas estão no centro das discussões sobre seu uso (FLORIDI, 2020). No campo jurídico, tais questões se tornam ainda mais sensíveis, dado o caráter interpretativo das normas e a necessidade de garantir imparcialidade nas decisões automatizadas. Esse contexto cria um ambiente fértil para o desenvolvimento de soluções de IA voltadas à interpretação e consulta de textos legais, abrindo caminho para os chamados *chatbots* jurídicos.

2.2. Processamento de Linguagem Natural (PLN)

O Processamento de Linguagem Natural (PLN) é uma subárea da Inteligência Artificial que tem como objetivo possibilitar a interação entre seres humanos e computadores por meio da linguagem natural, seja ela falada ou escrita. Em essência, o PLN busca fazer com que as máquinas compreendam, interpretem e gerem linguagem humana de forma significativa e contextualizada (JURAFSKY; MARTIN, 2023). Essa capacidade é fundamental para diversas aplicações modernas, como tradutores automáticos, mecanismos de busca, assistentes virtuais e, mais recentemente, sistemas de perguntas e respostas baseados em modelos de linguagem.

Historicamente, o PLN passou por diferentes abordagens. Inicialmente, predominavam métodos baseados em regras linguísticas, em que o conhecimento era explicitamente programado por especialistas. Com o avanço do aprendizado de máquina, surgiram técnicas estatísticas que passaram a analisar grandes conjuntos de textos para identificar padrões linguísticos e semânticos (MANNING; SCHÜTZE, 1999). Mais recentemente, com o advento do deep learning, os modelos de PLN evoluíram para representar palavras e sentenças em espaços vetoriais multidimensionais, possibilitando a criação de sistemas

com compreensão contextual e capacidade de geração coerente de texto (VASWANI et al., 2017).

Modelos de linguagem de grande escala, como BERT, GPT e LLaMA, revolucionaram o campo ao empregar arquiteturas baseadas em transformers, capazes de compreender relações de dependência entre palavras em longos contextos (BROWN et al., 2020). Esses modelos são treinados com bilhões de parâmetros e grande volume de dados, resultando em um desempenho sem precedentes em tarefas como tradução automática, sumarização de textos e respostas a perguntas. Tais avanços são o alicerce de tecnologias contemporâneas de *chatbots* e assistentes jurídicos inteligentes, capazes de consultar e resumir documentos complexos de forma automatizada.

No contexto jurídico, o PLN tem papel estratégico. A linguagem legal é caracterizada por alta densidade semântica, formalismo e vocabulário técnico, o que a torna um desafio especial para os modelos de linguagem. Assim, técnicas de PLN permitem não apenas extrair informações relevantes de jurisprudências, contratos e petições, mas também apoiar o raciocínio jurídico e a pesquisa de precedentes. Desse modo, o PLN se apresenta como um pilar essencial para o desenvolvimento de sistemas de apoio à decisão e para a construção de *chatbots* voltados à consulta em documentos jurídicos, tema central deste estudo.

2.3. Retrieval-Augmented Generation (RAG)

A arquitetura Retrieval-Augmented Generation (RAG) tem se destacado como uma das abordagens mais inovadoras para aprimorar a capacidade dos modelos de linguagem. Diferente das soluções tradicionais baseadas apenas em geração textual, o RAG combina técnicas de recuperação de informações e geração de texto, possibilitando que os modelos produzam respostas mais contextualizadas e fundamentadas em dados externos. Essa combinação permite mitigar uma das principais limitações dos modelos de linguagem convencionais: a tendência a gerar informações imprecisas ou desatualizadas quando não possuem acesso a uma base de conhecimento confiável (FINARDI et al., 2024).

O avanço de sistemas de inteligência artificial cada vez mais complexos tornou necessária a integração entre métodos de recuperação e de geração de informações, especialmente em aplicações que exigem respostas rápidas, precisas e contextualizadas (GAO et al., 2023; LEWIS et al., 2020). Nesse cenário, o RAG surge como uma arquitetura capaz de unir o melhor dos dois mundos, a busca eficiente de informações relevantes e a capacidade de redigir respostas coerentes, resultando em sistemas mais robustos e informativos.

Em sua estrutura, o RAG é composto por três módulos principais: o *Retriever*, o *Chunk* e o *Generator*. O componente *Retriever* é responsável por buscar passagens ou trechos de texto relevantes em uma base de dados previamente construída, que pode incluir documentos, artigos científicos ou registros jurídicos. Já o *Chunk* atua como a unidade de segmentação do conhecimento, dividindo os documentos em partes menores e semanticamente significativas, o que facilita a recuperação de informações específicas. Por fim, o *Generator* utiliza o material recuperado para produzir respostas contextualizadas e linguisticamente coerentes (KARPUKHIN et al., 2020; FINARDI et al., 2024).

O funcionamento colaborativo entre esses componentes faz do RAG uma arquitetura especialmente adequada para tarefas que envolvem o tratamento de grandes volu-

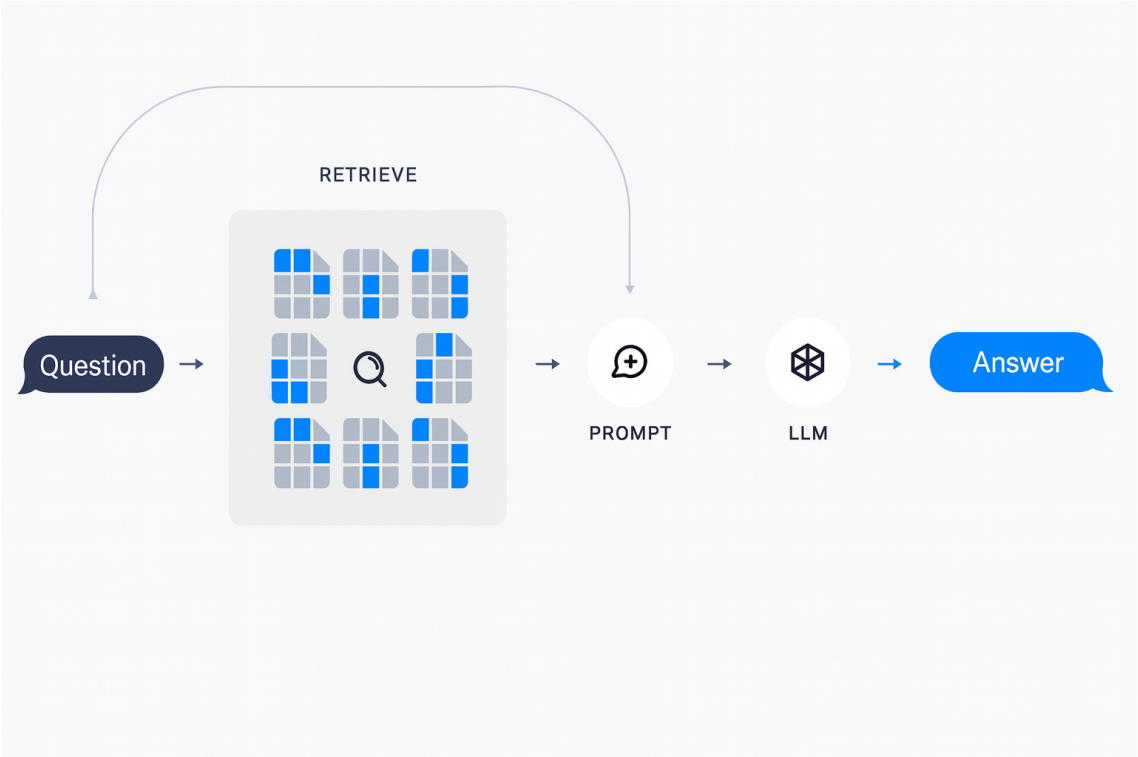

Figura 1. Fluxo de trabalho de RAG. Fonte: LANGCHAIN. Retrieval-Augmented Generation (RAG) Tutorial. LangChain Documentation, 2024. Disponível em: https://python.langchain.com/docs/tutorials/rag/. Acesso em: 16 out. 2025.

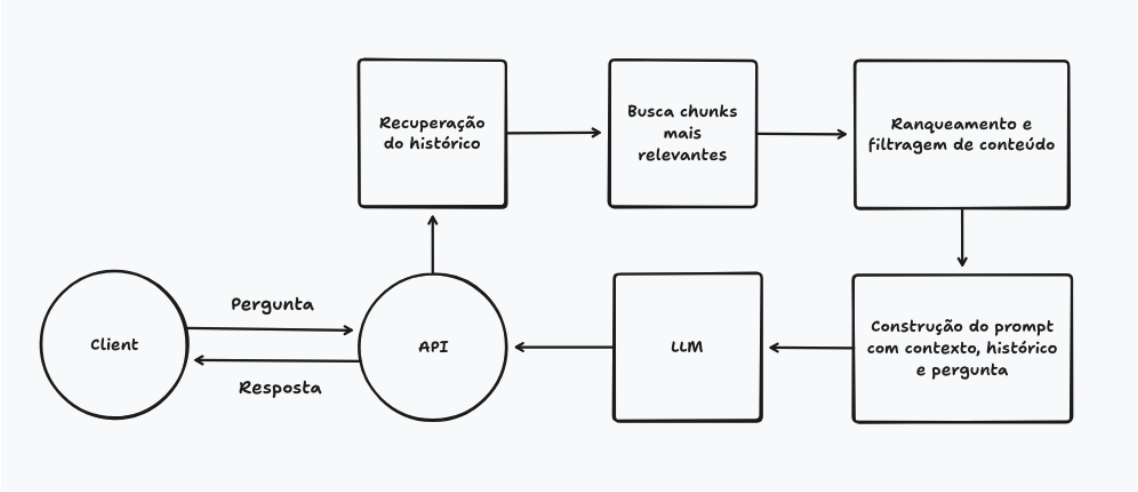

Figura 2. Fluxo de Envio de pergunta. Fonte: autoria própria.

mes de texto e a necessidade de precisão contextual, como no caso de consultas a bases jurídicas, científicas ou técnicas. Assim, o RAG representa um avanço significativo no campo da Inteligência Artificial aplicada à geração de linguagem, oferecendo uma alternativa eficiente e flexível para a construção de sistemas de resposta baseados em conhecimento (FINARDI et al., 2024; GAO et al., 2023; LEWIS et al., 2020).

A Figura 1 ilustra o fluxo de funcionamento de um sistema RAG convencional, no qual a interação entre o usuário e o modelo segue um caminho linear. Inicialmente, o texto de entrada é processado e enviado ao módulo *Retriever*, que realiza a busca por fragmentos relevantes em uma base vetorial previamente construída. Esses fragmentos são então passados ao *Generator*, caracterizado pelo *prompt* e *LLM (Large Language Model)*, que combina o contexto recuperado com sua capacidade de linguagem para formular uma resposta coerente e fundamentada. Esse fluxo evidencia a natureza integrada do RAG, onde a recuperação e a geração ocorrem de forma colaborativa e complementar, garantindo maior precisão e contextualização nas respostas geradas.

Já a Figura 2 apresenta o fluxo de funcionamento do sistema desenvolvido neste projeto, que expande a estrutura tradicional do RAG ao incorporar mecanismos adicionais de gerenciamento de contexto, histórico de conversas e vinculação dinâmica entre documentos e usuários. Nesse modelo, cada interação do usuário é processada com base em um histórico de mensagens associado a uma sessão específica, enquanto o módulo de recuperação atua filtrando os resultados conforme o documento selecionado. Essa arquitetura personalizada permite que o sistema mantenha coerência ao longo de múltiplas interações e assegure que as respostas estejam restritas às informações provenientes do documento ativo, garantindo assim maior controle, rastreabilidade e relevância no processo de geração de respostas.

2.4. Bancos Vetoriais e Recuperação Semântica com Embeddings

Os bancos de dados vetoriais têm se consolidado como uma das principais tecnologias para o armazenamento e recuperação de informações em aplicações baseadas em Inteligência Artificial. Diferentemente dos bancos relacionais tradicionais, que operam com

registros tabulares e consultas por igualdade de valores, os bancos vetoriais armazenam dados como vetores de alta dimensionalidade que são representações numéricas que capturam as características semânticas de textos, imagens ou outros tipos de conteúdo (TAIPALUS, 2023). Essa estrutura permite buscas por similaridade, o que é essencial em domínios como o jurídico, onde o significado contextual das palavras é mais relevante que sua forma literal.

O princípio central desses sistemas é a vetorização, processo responsável por transformar informações brutas, como trechos de petições ou artigos de lei, em representações numéricas que podem ser interpretadas por modelos de inteligência artificial. Esse procedimento, conhecido como *embedding*, gera vetores densos capazes de expressar relações semânticas entre palavras, frases ou documentos (TAIPALUS, 2023; HAN; LIU; WANG, 2023). Assim, textos com significados próximos permanecem próximos entre si nesse espaço, permitindo uma busca baseada em contexto, e não apenas em palavras-chave.

As representações vetoriais de palavras, conhecidas como *word embeddings*, foram popularizadas por Mikolov et al. (2013), que demonstraram como modelos treinados em grandes volumes de texto conseguem aprender relações semânticas complexas entre termos. Esses modelos superaram as antigas representações esparsas, como os vetores *one-hot*, ao capturar nuances linguísticas de maneira contínua. Por exemplo, enquanto métodos baseados em frequência apenas indicam a presença de termos, os *embeddings* associam palavras como "juiz" e "magistrado" por sua similaridade semântica, refletindo o uso análogo desses termos em contextos jurídicos.

Além disso, as relações vetoriais permitem fazer inferências por analogia, como a observação de que a diferença entre "advogado" e "cliente" pode se comportar de modo semelhante à diferença entre "médico" e "paciente". Essa capacidade é aproveitada por modelos modernos de vetorização de documentos jurídicos, capazes de identificar padrões de significado entre expressões legais diferentes, mas conceitualmente relacionadas, o que os torna úteis em sistemas de análise e busca jurídica.

A recuperação de informações em bancos vetoriais baseia-se em algoritmos de busca por similaridade. Quando uma consulta é submetida, por exemplo, uma pergunta sobre "direito do consumidor em contratos de prestação de serviço", o sistema transforma a consulta em um vetor no mesmo espaço de embeddings dos documentos armazenados. Em seguida, calcula-se a similaridade entre esse vetor e os vetores dos documentos disponíveis, retornando aqueles mais próximos semanticamente (HAN; LIU; WANG, 2023; PINECONE, 2023).

Para garantir eficiência em bases extensas, utilizam-se algoritmos aproximados de vizinhos mais próximos (*Approximate Nearest Neighbors* – ANN), que evitam a comparação exaustiva entre todos os vetores. Esses métodos aplicam técnicas de indexação e particionamento do espaço vetorial, possibilitando desempenho escalável e tempos de resposta adequados, mesmo em coleções jurídicas compostas por milhares de documentos (TAIPALUS, 2023).

Em contraste, sistemas de busca convencionais, baseados em índices invertidos, dependem da correspondência literal de termos, o que os torna limitados frente ao chamado problema do vocabulário divergente. Por exemplo, uma busca por "rescisão con-

tratual" pode não retornar documentos que utilizem "dissolução de contrato", apesar do sentido equivalente (ROY et al., 2019). A busca vetorial supera essa limitação ao operar com representações semânticas: documentos e consultas que compartilham significados semelhantes ocupam regiões próximas no espaço de embedding, permitindo que o sistema identifique e retorne textos juridicamente relevantes mesmo que as expressões não coincidam exatamente (ROY et al., 2019).

Essa propriedade faz dos bancos vetoriais uma tecnologia essencial para aplicações jurídicas baseadas em IA, como *chatbots* e sistemas de apoio à decisão, que precisam compreender consultas em linguagem natural e localizar respostas juridicamente precisas em grandes acervos documentais.

3. Metodologia

3.1. Materiais

O desenvolvimento do chatbot jurídico foi realizado utilizando uma arquitetura de três camadas, composta por um front-end responsável pela interface do usuário, um back-end que gerencia a lógica de negócio e a comunicação com o banco de dados, e a camada de serviços externos, responsável pelo consumo de APIs e modelos de linguagem.

O backend foi desenvolvido com a linguagem de programação Python, escolhida pela ampla compatibilidade com ferramentas de Inteligência Artificial, Modelos de Linguagem de Grande Escala (*LLMs*), e Bancos de Vetores. O framework[1] adotado foi o FastAPI, que se destaca entre as alternativas disponíveis por sua sintaxe enxuta e intuitiva, geração automática de documentação e suporte nativo à criação de APIs RESTful assíncronas.

Na camada de frontend foi utilizada a linguagem de programação JavaScript, em conjunto com a biblioteca React.js, o que possibilitou a criação de uma interface dinâmica, responsiva e modular, com componentes reutilizáveis e integração eficiente com o backend por meio de requisições HTTP (Hypertext Transfer Protocol) à API principal.

A camada de serviços externos foi desenvolvida com base na biblioteca LangChain, que oferece uma estrutura modular para o desenvolvimento de sistemas baseados em Retrieval-Augmented Generation (RAG). Essa abordagem permite combinar a geração de texto por modelos de linguagem com a recuperação semântica de informações relevantes em um repositório especializado. Além disso, foi utilizada a plataforma GroqCloud, que disponibiliza recursos de modelos de linguagem de grande porte (LLMs) em um plano gratuito, permitindo o uso de modelos que normalmente exigem máquinas de alta performance para execução. Também foi empregada a plataforma Hugging Face, um ambiente colaborativo da comunidade de machine learning que disponibiliza modelos de IA/ML, conjuntos de dados (datasets) e aplicações. Nessa plataforma, foi utilizado um modelo especializado para a geração de embeddings.

O armazenamento vetorial das informações jurídicas foi realizado com o ChromaDB, um banco de dados vetorial de código aberto otimizado para buscas semânticas de alta performance. Os documentos jurídicos, como leis, petições, pareceres e decisões

[1] De acordo com Binder et al. (2013), um framework estabelece um conjunto de pressupostos, conceitos, valores e práticas que orientam determinado domínio de estudo.

judiciais, foram processados em formato PDF, convertidos em texto, segmentados em chunks e convertidos em *embeddings* onde permaneceram no espaço vetorial criado pelo banco de vetores.

O modelo de linguagem responsável pela geração das respostas foi o Llama 3.3 70B Versatile, desenvolvido pela empresa Meta. Trata-se de um modelo avançado e multilíngue de grande porte, otimizado para uma ampla variedade de tarefas de processamento de linguagem natural (NLP). Com 70 bilhões de parâmetros, o modelo oferece alto desempenho em diversos benchmarks, mantendo eficiência e versatilidade para diferentes aplicações.

Foram utilizados o Docker e o Docker Compose para a criação e orquestração de contêineres, o GitHub Actions para automação de deploy e integração contínua, e uma máquina virtual disponibilizada pela instituição, acessada remotamente via SSH para a hospedagem e execução do sistema.

Por fim, a persistência dos dados de usuários, histórico de conversas e autenticação foi implementada utilizando o PostgreSQL, por meio da plataforma NeonDB, a qual permite a criação de instâncias de bancos de dados hospedadas em nuvem. Essa plataforma oferece planos gratuitos e garante alta disponibilidade e estabilidade, assegurando, assim, a integridade e a segurança das informações. Dessa forma, a utilização desse banco de dados teve como finalidade viabilizar o armazenamento e o gerenciamento estruturado das informações, abrangendo usuários, sessões, documentos consultados e mensagens.

3.2. Métodos

O processo metodológico adotado seguiu uma abordagem de desenvolvimento iterativa e incremental, estruturada em três etapas principais: implementação do sistema backend, desenvolvimento da interface web e deploy com automação por meio de *CI/CD (Continuous Integration/Continuous Delivery)*. Para a organização e acompanhamento das tarefas, foram utilizados os quadros de Kanban do GitHub Projects, centralizando o desenvolvimento em um único ambiente e garantindo maior visibilidade e controle sobre as etapas do projeto. Além disso, a plataforma GitHub foi utilizada como repositório remoto para o código-fonte, com a separação dos repositórios entre frontend e backend, o que facilitou a distribuição das tarefas, o gerenciamento de versões e a colaboração entre as partes do sistema. O ambiente de hospedagem foi disponibilizado pela instituição por meio de uma máquina virtual, acessada via SSH, onde o sistema foi implantado utilizando Docker e Docker Compose para conteinerização e orquestração dos serviços. O processo de deploy foi automatizado com o GitHub Actions, garantindo maior eficiência, rastreabilidade e integração contínua entre as etapas de desenvolvimento e publicação.

4. Desenvolvimento

4.1. Concepção e objetivos iniciais

O desenvolvimento do Chatbot Jurídico teve início a partir da necessidade de criar uma ferramenta capaz de auxiliar na consulta e interpretação de documentos legais de forma automatizada e acessível. A proposta inicial consistia em disponibilizar uma aplicação que, por meio de técnicas de Processamento de Linguagem Natural (PLN) e Recuperação de Informação, pudesse responder a perguntas com base em arquivos jurídicos, reduzindo o tempo de pesquisa e ampliando a precisão na obtenção de informações.

A fase inicial teve caráter exploratório, buscando compreender as limitações e potencialidades das tecnologias envolvidas. O foco esteve na construção de um protótipo mínimo viável (MVP), executado via terminal, sem interface gráfica, com o objetivo de validar o funcionamento básico de geração e consulta a embeddings em documentos de texto. Essa versão inicial visava apenas responder perguntas gerais, sem funcionalidades adicionais, permitindo avaliar o fluxo entre entrada de dados, pré-processamento e resposta do modelo.

A primeira versão do sistema não contemplava uma API, era apenas funções que criavam um *client* de llm para testar o envio de perguntas e recebimento de uma resposta, com tentativas de limitar as respostas ao contexto jurídico apenas pelo *prompt*. O modelo de llm utilizado inicialmente foi o `tiiuae/falcon-7b-instruct` da Hugging Faces. A escolha desse modelo foi motivada por sua simplicidade e facilidade de uso, características que o tornaram adequado para a criação de um chatbot minimamente viável.

O modelo de embeddings utilizado foi o `sentence-transformers/all-MiniLM-L6-v2`, escolhido por sua eficiência e ampla utilização em aplicações de geração de embeddings, tendo como foco inicial compreender o funcionamento desse processo.

O processo de extração do texto e de divisão em trechos menores foi planejado com maior cuidado. Como o envio de documentos ainda não havia sido implementado, criou-se uma pasta para armazenar o arquivo, possibilitando alternar entre os documentos nela contidos durante os testes. A extração do texto foi realizada por meio da biblioteca PyMuPDF, que oferece recursos para extração, análise, conversão e manipulação de arquivos em formato PDF, entre outros.

Com o texto extraído, o passo seguinte consistiu em dividi-lo em trechos menores, a fim de facilitar a busca no banco de vetores após a conversão dos trechos em embeddings. Para essa etapa, foi utilizada a biblioteca LangChain em conjunto com a técnica RecursiveCharacterTextSplitter, que realiza uma divisão hierárquica do texto, respeitando delimitadores naturais como parágrafos, pontos e espaços, além de manter uma sobreposição entre os chunks para preservar o contexto. Caso algum trecho ainda ultrapasse o tamanho máximo permitido, a divisão é realizada de forma recursiva, até que todos os segmentos estejam devidamente ajustados e prontos para a geração de embeddings precisos.

Por exemplo, considerando o seguinte texto jurídico:

”O Tribunal Superior reconheceu a necessidade de revisão das cláusulas contratuais que impliquem em desequilíbrio econômico entre as partes. Em casos de inadimplemento, a parte prejudicada pode requerer a resolução do contrato ou a revisão judicial das condições pactuadas. Além disso, a legislação consumerista garante proteção adicional ao consumidor frente a práticas abusivas, permitindo que cláusulas consideradas leoninas sejam anuladas. O juiz, ao avaliar a lide, deve considerar não apenas a letra da lei, mas também os princípios da boa-fé e da função social do contrato, equilibrando os interesses das partes envolvidas.”

O *splitter* gerou os seguintes *chunks*:

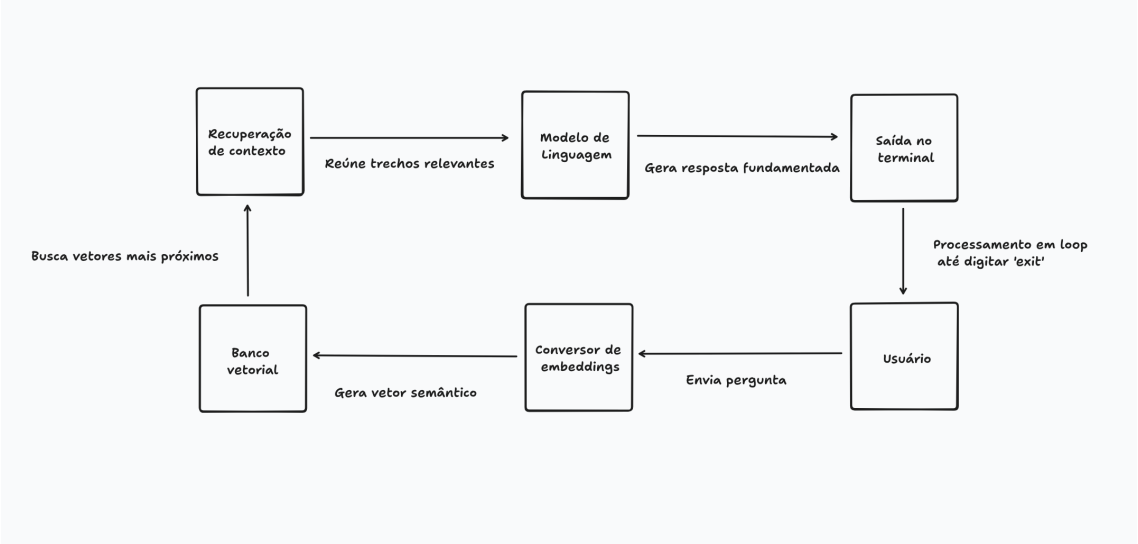

Figura 3. Fluxo de execução do produto no estágio inicial. Fonte: autoria propria

Chunk 1: O Tribunal Superior reconheceu a necessidade de revisão das cláusulas contratuais que impliquem em desequilíbrio econômico entre as partes.

Chunk 2: entre as partes. Em casos de inadimplemento, a parte prejudicada pode requerer a resolução do contrato ou a revisão judicial das condições pactuadas.

Chunk 3: condições pactuadas. Além disso, a legislação consumerista garante proteção adicional ao consumidor frente a práticas abusivas, permitindo que cláusulas consideradas leoninas sejam anuladas.

Chunk 4: sejam anuladas. O juiz, ao avaliar a lide, deve considerar não apenas a letra da lei, mas também os princípios da boa-fé e da função social do contrato, equilibrando os interesses das partes envolvidas.

Os parâmetros de configuração dessa função foram definidos com base no tamanho do texto. Optou-se por um valor de 500 caracteres por trecho, com uma sobreposição de 50 caracteres, o que significa que cada chunk continha 500 caracteres e que os 50 caracteres finais de um trecho eram repetidos no início do seguinte, a fim de preservar a continuidade do contexto.

O banco de vetores empregado foi o Facebook AI Similarity Search (FAISS). A etapa de indexação vetorial tem como finalidade organizar os embeddings em uma estrutura que possibilite medir a similaridade entre diferentes trechos de texto. Para isso, utiliza-se uma base de vetores na qual cada embedding ocupa uma posição em um espaço de múltiplas dimensões, representando semanticamente o conteúdo textual. O método de busca por distância euclidiana (L2) é comumente utilizado para identificar os vetores mais próximos entre si, ou seja, os fragmentos com maior relação semântica. Essa estrutura permite que, ao ser realizada uma consulta, o sistema localize de forma eficiente os trechos mais relevantes e semanticamente semelhantes ao texto pesquisado.

Nesta etapa final da concepção do sistema, foi implementado o fluxo completo de interação entre o usuário, o mecanismo de recuperação de informações e o modelo

de geração de linguagem natural, caracterizando o núcleo funcional do chatbot jurídico, assim como apresentado na Figura 3. A interface consistia em um terminal, de modo que, após a inicialização do sistema, um laço de repetição era ativado, permitindo que o usuário encerrasse o processo por meio da palavra *exit*. O processo tinha início quando o usuário inseria uma pergunta, a qual era convertida em um vetor de embeddings por meio do modelo de representação semântica previamente configurado. Em seguida, esse vetor era comparado aos vetores armazenados no banco vetorial, possibilitando a identificação dos trechos de texto mais semanticamente relevantes para a consulta. Os fragmentos recuperados eram então reunidos e utilizados como contexto informativo, servindo de base para que o modelo de linguagem gerasse uma resposta textual coerente e fundamentada. Essa integração entre as etapas de recuperação e geração constitui o princípio central da arquitetura Retrieval-Augmented Generation (RAG), na qual a geração de texto não ocorre de forma isolada, mas orientada por conhecimento extraído de fontes previamente indexadas. Apesar da implementação deste fluxo e das técnicas empregadas, o sistema ainda não apresentava respostas plenamente contextualizadas, precisas e semanticamente consistentes com os documentos jurídicos utilizados como base informacional, evidenciando a necessidade de aprimoramento na acurácia das respostas.

4.2. Evolução do projeto

Com o funcionamento básico validado, o projeto passou por uma sequência de incrementos estruturados. Inicialmente, foram implementadas funções de carregamento e divisão dos documentos em blocos menores (chunks), fundamentais para que o modelo pudesse indexar o conteúdo de forma semântica. Em seguida, adicionou-se a geração de embeddings com o uso de modelos da Hugging Face, criando representações vetoriais que foram armazenadas em um banco de dados vetorial (vector store) por meio da biblioteca ChromaDB. A primeira grande evolução foi a implementação de um chatbot com um modelo criado localmente, utilizando os modelos da Ollama, que é uma ferramenta de código aberto para executar llms. Por fim, foi criada uma interface desenvolvida no Gradio. Esse chatbot permitiu uma interação mais eficiente com o sistema, tornando possível consultar o conteúdo de forma mais dinâmica.

Na sequência, foi explorado a opção de gerar embeddings diretamente a partir de frases, o que possibilitou classificar o contexto do conteúdo. Para isso, as frases foram classificadas em duas categorias: ”jurídicas”e ”não jurídicas”, usando um sistema binário de marcação (0 para não jurídicas e 1 para jurídicas). Essa classificação ajudou a refinar a indexação e otimizar a relevância das respostas fornecidas pelo sistema.

A adição de funcionalidades também representou um marco importante na evolução do projeto. Entre as melhorias, destacam-se a implementação do upload de arquivos `.docx` e `.pdf` por meio da interface do Gradio, o que ampliou as fontes de documentos acessíveis ao sistema. Em conjunto, foi implementado o método RAG (Retrieval-Augmented Generation), permitindo ao modelo buscar informações relevantes dentro dos documentos antes de gerar respostas, enriquecendo o desempenho geral do chatbot, posteriormente o envio foi limitado apenas para arquivos `.pdf`.

Para garantir que os arquivos fossem processados corretamente, foi desenvolvido um mecanismo para salvar os PDFs em uma pasta local. Esse fluxo foi essencial para que o banco ChromaDB conseguisse acessar e organizar o conteúdo dos documentos. Além

disso, foram criadas funções específicas para carregar e dividir os documentos em chunks menores, otimizando o processo de indexação. Com essas melhorias, foi possível criar embeddings mais precisos e realizar a indexação dos chunks de forma eficaz, estabelecendo uma base sólida para o uso do sistema em diferentes contextos.

Embora tenha sido iniciado um experimento de classificação manual de frases, com a marcação de trechos como jurídicos ou não jurídicos, essa abordagem acabou não sendo levada adiante. A continuidade dessa etapa exigiria um esforço significativo de rotulação manual, o que poderia desviar o foco e comprometer os objetivos centrais do projeto, voltados à construção de um sistema funcional de recuperação semântica e geração de respostas automatizadas. Optou-se, portanto, por priorizar a evolução da arquitetura e da integração dos componentes, mantendo a proposta inicial de um sistema escalável e de fácil adaptação a diferentes conjuntos documentais.

A implementação de uma interface web utilizando o framework Gradio, aliada à modularização das funções do sistema, proporcionou uma visão mais clara sobre o funcionamento geral do chatbot e sobre os caminhos possíveis para sua evolução. Essa estrutura experimental permitiu validar interações em tempo real, testar diferentes fluxos de recuperação e resposta, além de favorecer uma organização mais coerente entre os módulos de indexação, recuperação e geração. A partir dessa consolidação conceitual e técnica, surgiu a necessidade de expandir o sistema para além do ambiente local, levando à concepção de uma API dedicada que tornasse o chatbot acessível a outras aplicações e interfaces.

Com a infraestrutura de indexação consolidada, o projeto avançou para a etapa de consulta e geração de respostas. Foi então implementada uma rota de API dedicada, denominada /question, responsável por receber a pergunta do usuário, recuperar os vetores mais relevantes no banco de dados vetorial e, a partir deles, gerar uma resposta textual. Esse estágio marcou a introdução do conceito de Retrieval-Augmented Generation (RAG), que combina a recuperação semântica do conteúdo com a geração de linguagem natural, formando uma pipeline unificada e mais precisa na entrega de respostas contextualizadas.

A sequência de aprimoramentos realizados ao longo do desenvolvimento do projeto evidenciou uma evolução sólida e progressiva na arquitetura do chatbot jurídico, refletindo o amadurecimento técnico e conceitual da solução. Inicialmente, foram efetuados ajustes na indexação dos chunks, com o intuito de promover uma segmentação mais semântica e precisa dos documentos jurídicos, o que resultou em ganhos expressivos na qualidade e relevância das informações recuperadas. Em seguida, o sistema passou a contar com integração ao GroqChat do LangChain, o que viabilizou a utilização de modelos de linguagem de maior capacidade e número de parâmetros, algo inviável no ambiente do Ollama, uma vez que sua execução estava limitada aos recursos de processamento da máquina local, capaz de suportar apenas modelos com até 7 bilhões de parâmetros, ampliando significativamente o potencial de geração de respostas contextualizadas e tecnicamente consistentes. Na sequência, foi realizada a reescrita do prompt de teste, voltada à avaliação do desempenho do chatbot diante de perguntas fora do contexto documental. Essa modificação teve como propósito conter o escopo das respostas ao domínio jurídico por meio de técnicas de engenharia de prompts, mantendo-se fiel à proposta inicial de simplicidade arquitetural e adesão a ferramentas de código aberto, sem a necessidade de

processos complexos de classificação manual. Por fim, a implementação de um retriever dedicado ao banco vetorial, aliado a um mecanismo de memória, consolidou uma etapa de maturidade no sistema, tornando as respostas mais coerentes, contextualmente sensíveis e alinhadas ao histórico das interações com o usuário.

Após a implementação das rotas principais /question e /uploadfile, o sistema atingiu um ponto de maturidade que permitia a interação completa entre o usuário e o modelo de linguagem, possibilitando tanto o envio de perguntas quanto o processamento e armazenamento de novos documentos no banco vetorial. Com essas duas funcionalidades essenciais consolidadas, o foco do desenvolvimento se voltou à estruturação do sistema de autenticação e gerenciamento de usuários, com o objetivo de transformar o protótipo em uma aplicação multiusuário.

Paralelamente ao avanço no backend, iniciou-se o desenvolvimento de uma interface web utilizando React.js, com o intuito de proporcionar uma camada visual intuitiva para interação com o sistema. Nessa fase inicial, foi implementado um chat funcional básico, composto por uma barra de perguntas, um botão para envio de documentos e a área de exibição das respostas geradas pelo modelo. Essa estrutura simples permitiu testar a integração direta com as rotas /question e /uploadfile, garantindo a comunicação fluida entre o frontend e a API. Além disso, essa etapa foi essencial para preparar o ambiente para a futura implementação do sistema de autenticação, estabelecendo a base necessária para o gerenciamento de sessões e a personalização da experiência do usuário.

Uma importante etapa dessa nova fase consistiu na implementação de um banco de dados PostgreSQL, responsável por armazenar de forma persistente as informações de usuários, sessões e histórico de conversas. Essa camada de persistência foi fundamental para garantir a continuidade das interações e a personalização da experiência de uso. Em seguida, foram desenvolvidas rotas específicas para autenticação e controle de acesso, incluindo o registro e login de usuários.

Para aprimorar a acessibilidade, o sistema passou a contar com integração de login via Google, oferecendo uma alternativa prática de autenticação externa e reduzindo a necessidade de gerenciamento manual de credenciais. Além disso, foram implementadas rotas complementares de manutenção e controle do chat, como as de renomear e deletar conversas, que permitiram ao usuário organizar melhor suas sessões.

Por fim, o desenvolvimento avançou com a criação da rota para recuperação de senha e verificação de conta pelo email, tornando o sistema o mais próximo de um sistema comercial, completando o ciclo de autenticação e assegurando uma experiência mais robusta e confiável. Paralelamente, foram realizados ajustes nos parâmetros da llm, visando otimizar o desempenho e garantir a coerência entre entre as perguntas e respostas. Essa etapa consolidou a transição do projeto de um protótipo experimental para uma plataforma completa de chatbot jurídico, com autenticação segura, persistência de dados e gestão personalizada das interações.

4.3. Etapa Final de Estruturação e Implantação do Sistema

Na fase final do desenvolvimento, o sistema já contava com todas as funcionalidades centrais plenamente consolidadas, abrangendo o fluxo completo de upload de documentos, envio de perguntas e recuperação de informações por meio da arquitetura RAG. Também estavam implementados os mecanismos de autenticação de usuários, incluindo registro

com verificação de conta por e-mail, recuperação de senha via e-mail e login social com Google, além da persistência do histórico de conversas no banco de dados. Com essas etapas concluídas, o foco do trabalho passou a ser o aperfeiçoamento da arquitetura do projeto, visando aprimorar a organização estrutural, aumentar a escalabilidade e facilitar a manutenção e evolução futura do código.

Um dos principais aprimoramentos do sistema foi a criação de novas rotas e lógicas de controle voltadas para levar as informações de forma mais precisa ao frontend. A implementação de uma rota para recuperar detalhes de chats e documentos permitiu que cada interação fosse contextualizada, refletindo com exatidão o vínculo entre usuário, chat e arquivo associado. Paralelamente, a adição de um filtro no ChromaDB por documento foi essencial para a logística de usuários e chats ativos, garantindo que cada chat estivesse vinculado a um único documento. Dessa forma, o mesmo arquivo pode ser utilizado por diferentes usuários sem gerar conflitos, tanto no banco de vetores quanto no banco de dados relacional, promovendo organização e consistência em todo o sistema. Além disso, foram realizados ajustes de integridade entre os bancos de dados, assegurando que os registros no banco vetorial e no relacional fossem devidamente excluídos quando um chat fosse removido, evitando inconsistências e acúmulo de dados redundantes.

Com a base funcional consolidada, iniciou-se a preparação para o ambiente de implantação e distribuição. Foram criados os arquivos Dockerfile e Docker Compose, que automatizaram a configuração do ambiente de execução, facilitando tanto o desenvolvimento local quanto a futura implantação em servidores. Essa etapa marcou a transição do projeto de um protótipo em ambiente de testes para uma aplicação pronta para ser disponibilizada em produção.

Por fim, o processo de organização da arquitetura do código representou o passo conclusivo da etapa de desenvolvimento. O projeto foi reestruturado de modo a dividir as responsabilidades em módulos independentes. A pasta app/ passou a reunir os principais componentes da aplicação, organizados em diretórios como core, db, models, repositories, routes, schemas, services e utils, refletindo uma arquitetura limpa e modular. Essa nova estrutura não apenas aprimorou a legibilidade e manutenção do código, mas também estabeleceu uma base sólida para futuras expansões do sistema, encerrando o ciclo de desenvolvimento com uma implementação robusta, escalável e devidamente organizada.

4.4. Arquitetura do sistema

A arquitetura do Chatbot Jurídico, representada na figura 4, foi concebida de forma modular, priorizando a separação de responsabilidades entre os componentes e a escalabilidade da aplicação. O sistema foi dividido em dois módulos principais: o backend, desenvolvido em Python com FastAPI, e o frontend, implementado em ReactJS. O backend é responsável pelo processamento de linguagem natural, comunicação com os modelos de linguagem e gerenciamento de dados, enquanto o frontend gerencia a interação com o usuário e a visualização das respostas.

Os fluxos de envio do documento e pergunta, ilustrados na figura 4, ocorrem da seguinte forma: o usuário realiza o upload de um documento jurídico em formato PDF; o servidor armazena o arquivo localmente e o submete a um processo de leitura e fragmentação em *chunks*. Esses fragmentos são então convertidos em *embeddings*. Os vetores resultantes são indexados no ChromaDB, onde permanecem disponíveis para

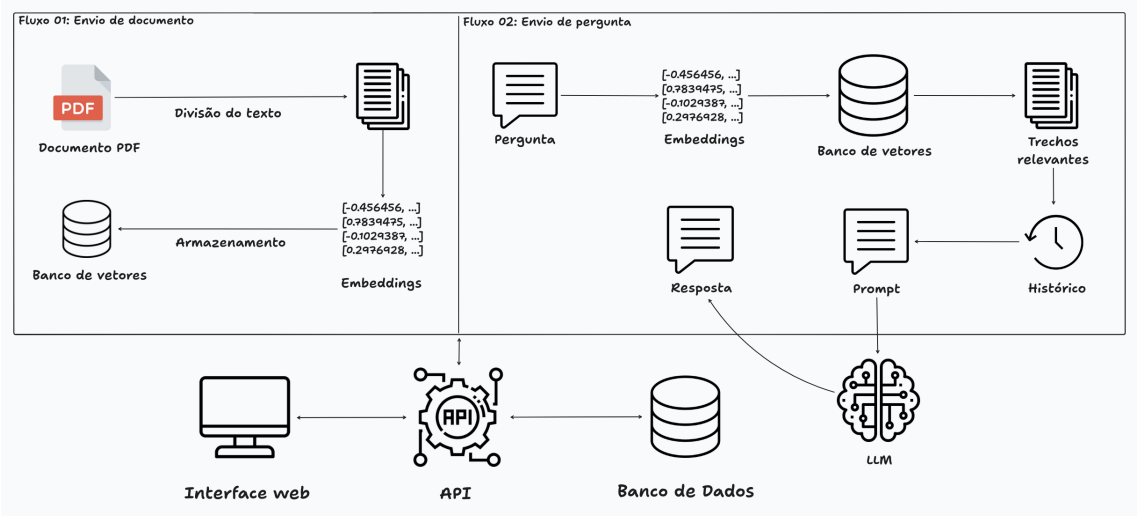

Figura 4. Fluxo de execução do sistema. Fonte: autoria propria

consultas futuras. Quando uma pergunta é enviada, o sistema busca os vetores semanticamente mais próximos, reordena, filtra os resultados e envia o contexto relevante ao modelo de linguagem, que gera a resposta final.

A figura 5 detalha o fluxo interno desse processo, evidenciando como o cliente interage com a *API* e como os diferentes componentes do backend cooperam para gerar as respostas. Esse diagrama ilustra de maneira clara o percurso da requisição até a produção da resposta final, permitindo compreender as etapas envolvidas no mecanismo de Recuperação e Geração (RAG) adotado pelo sistema.

O processo inicia-se com a pergunta enviada pelo cliente através de uma requisição ao servidor, que é recebida pela função `query_rag()`, responsável por dar início ao processo de geração da resposta ao instanciar o *retriever*, componente encarregado de encontrar informações relevantes no banco de vetores. A pergunta também é convertida em um vetor. Esse vetor é comparado com os vetores armazenados para identificar os trechos mais semanticamente similares à pergunta. Em seguida, as três últimas interações do usuário com o chatbot são recuperadas do banco de dados. Com essas informações, o sistema compõe um *prompt* que integra o histórico recente, os trechos recuperados e a pergunta atual, além das instruções que orientam o modelo. O *prompt* estruturado organiza todas as informações relevantes em um formato coerente e interpretável para o modelo. Esse *prompt* é então encaminhado a uma LLM externa à API. A LLM recebe o *prompt*, converte cada token em vetores numéricos e utiliza essas representações para realizar os cálculos necessários à previsão do próximo token. Esse processo é repetido iterativamente até que todo o texto seja gerado, resultando na resposta final. Após gerar a resposta, a LLM devolve à API, que por sua vez a envia para o cliente.

Além do armazenamento vetorial, a aplicação também utiliza um banco de dados relacional PostgreSQL, responsável por persistir informações de usuários, histórico de conversas, documentos e metadados dos chats. Essa combinação de bancos (relacional e vetorial) garante tanto a integridade dos dados estruturados quanto a eficiência na recuperação semântica.

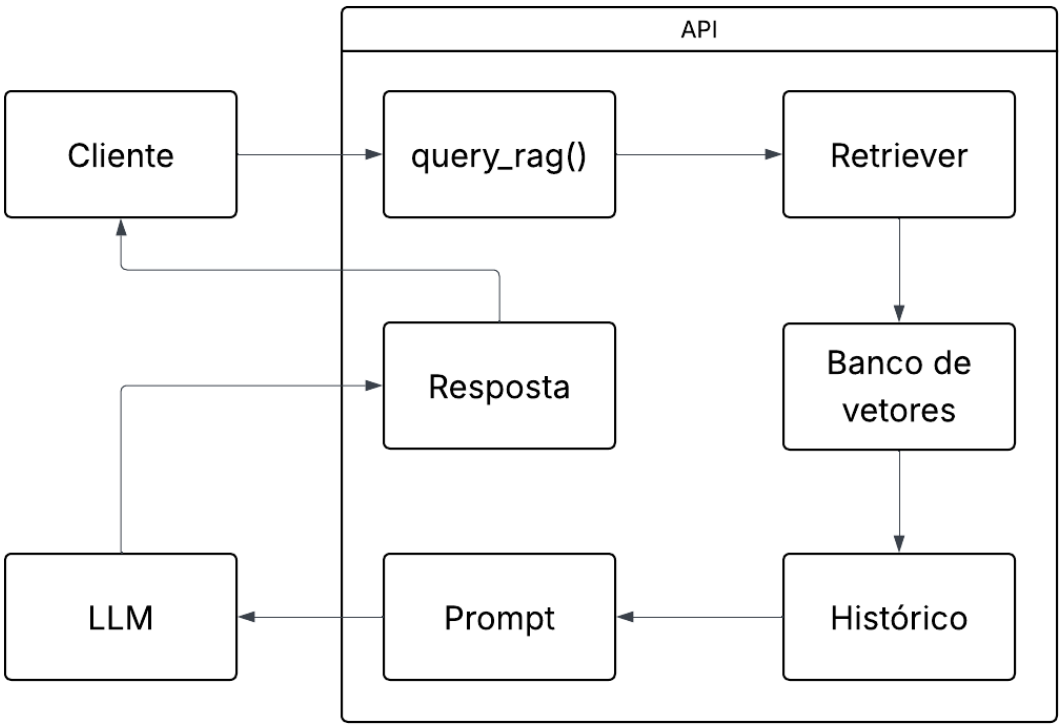

Figura 5. Arquitetura do sistema. Fonte: autoria propria

4.5. Backend e principais componentes

O backend foi desenvolvido com base na estrutura assíncrona do FastAPI, possibilitando alta performance e modularização. As rotas foram divididas em grupos conforme suas finalidades. Entre as principais, destacam-se:

- `/uploadfile`: responsável pelo recebimento e armazenamento dos arquivos PDF enviados pelos usuários, restringindo a entrada apenas a documentos compatíveis.
- `/question`: encarregada de processar as consultas, buscar os vetores relevantes no banco de embeddings e gerar respostas utilizando o modelo hospedado na Groq.
- `/users`, `/auth`, `/chats`: relacionadas à autenticação, gerenciamento de usuários e controle das sessões de conversa.

Para o processamento dos documentos, foram criadas funções específicas para leitura, divisão em *chunks*, geração de embeddings e indexação. Cada etapa foi encapsulada em módulos independentes, permitindo fácil manutenção e substituição de componentes. Também foi implementada uma função de limpeza do banco de dados vetorial, a fim de evitar inconsistências e otimizar o espaço de armazenamento.

O retriever com memória foi adicionado posteriormente, permitindo que o chatbot mantivesse o contexto da conversa sem perder eficiência. Essa funcionalidade foi integrada à API por meio do *LangChain*, que coordenou a comunicação entre as etapas de recuperação e geração de resposta. O histórico de mensagens foi persistido no banco relacional, associando cada entrada ao usuário e ao documento correspondente.

4.6. Frontend e interface do usuário

O frontend foi construído com *ReactJS*, utilizando componentes reutilizáveis e uma arquitetura voltada à experiência do usuário. A interface de chat permite o envio de mensagens e exibição de respostas em tempo real, bem como o upload de documentos PDF para indexação. Também foram implementadas seções adicionais, como a visualização de histórico de conversas e a seleção de documentos ativos.

O layout foi desenvolvido com base em um modelo de aplicação moderna, contendo uma barra lateral para navegação entre chats e configurações, e uma área principal dedicada ao diálogo com o assistente. A integração com o Google foi adicionada para facilitar o login e o registro de usuários, ampliando a acessibilidade e reduzindo a necessidade de cadastros manuais.

A comunicação entre o frontend e o backend é realizada por meio de requisições HTTP utilizando *fetch* e autenticação via token JWT. Para melhorar a organização do código, foi empregado o *React Context* para o gerenciamento de estados globais, incluindo a persistência do documento ativo durante as conversas.

4.7. Estratégias de RAG e otimização

O núcleo de funcionamento do Chatbot Jurídico baseia-se no paradigma de *Retrieval-Augmented Generation* (RAG). Nessa abordagem, as respostas não são geradas unicamente a partir do modelo de linguagem, mas sim com base em informações recuperadas

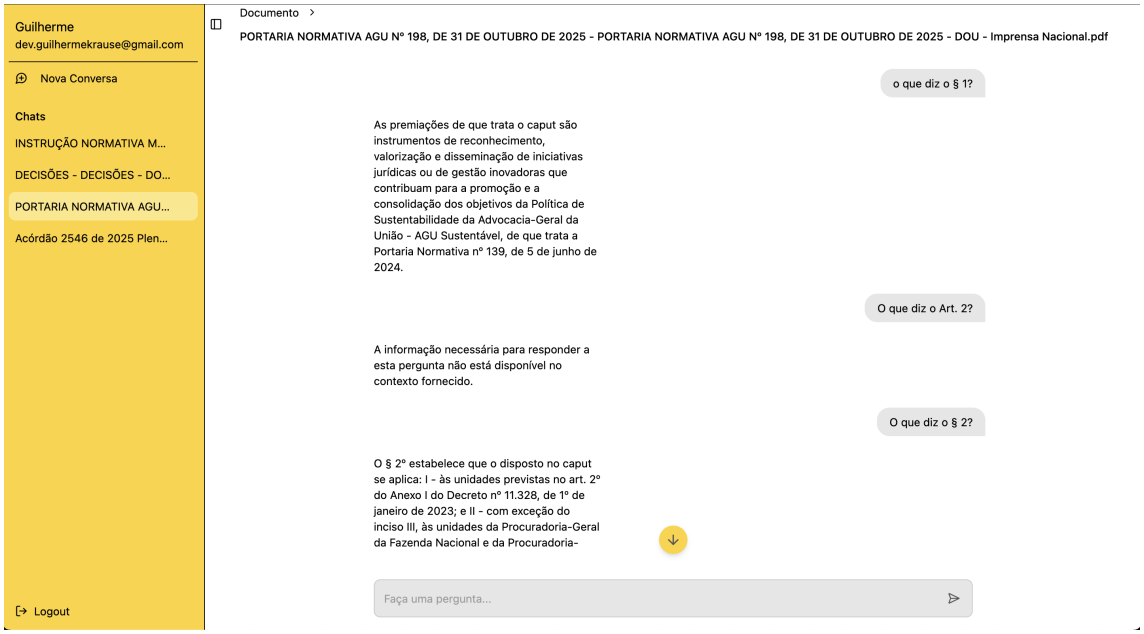

Figura 6. Interface do chatbot. Fonte: autoria propria

de uma base de conhecimento vetorial. Isso garante que o conteúdo das respostas esteja alinhado ao material jurídico fornecido pelo usuário.

Para aprimorar a qualidade das respostas, foram testadas diferentes estratégias de divisão de texto, variando o tamanho e a sobreposição dos fragmentos de texto. Também foram implementadas técnicas de truncamento do histórico de conversas, com o objetivo de limitar o contexto transmitido ao modelo e reduzir a latência das respostas. O reranqueamento de trechos com a API da Cohere mostrou-se uma etapa relevante na melhoria da precisão semântica, permitindo que os fragmentos mais contextualmente relevantes fossem priorizados na geração de resposta.

A combinação entre embeddings da *Hugging Face*, indexação via *ChromaDB* e reranqueamento da *Cohere* compôs um pipeline robusto e modular. Essa estrutura favorece futuras substituições de componentes, como a troca do modelo de embeddings ou ajustes no mecanismo de busca, sem necessidade de reescrever o sistema central.

4.8. Deploy e automação

Com o sistema funcional, a etapa seguinte consistiu na implantação e automação do ambiente de execução. Foram criados arquivos *Dockerfile* e *docker-compose* para contêinerização dos serviços, permitindo o empacotamento padronizado do backend, frontend e banco de dados. Essa abordagem garantiu portabilidade e simplificou a replicação do ambiente em outros servidores.

A integração contínua foi configurada por meio do *GitHub Actions*, permitindo a execução automática sempre que novas versões do código fossem enviadas ao repositório. O deploy final foi realizado em uma máquina virtual com acesso via SSH, configurada para hospedar os contêineres e gerenciar a comunicação entre os serviços. O uso de variáveis de ambiente e volumes persistentes assegurou a integridade dos dados e a esca-

labilidade do sistema.

4.9. Síntese do desenvolvimento

O desenvolvimento do Chatbot Jurídico demonstrou a viabilidade técnica da integração entre um banco de vetores, modelos de linguagem diferentes uma da Groq Cloud e outro da Ollama e uma interface web feita com ReactJS para aplicações no campo jurídico. O processo, conduzido de forma iterativa e incremental, permitiu a construção de uma solução funcional, escalável e alinhada às demandas contemporâneas de automação e acessibilidade da informação legal. Ainda que etapas de validação e aprimoramento estejam em andamento, o sistema já representa um avanço significativo na interseção entre Inteligência Artificial e Direito.

5. Desafios e trabalhos futuros

Durante o desenvolvimento, alguns desafios se destacaram. Entre eles, a necessidade de equilibrar a qualidade das respostas com o tempo de resposta do modelo, especialmente em consultas longas ou com documentos extensos. Ajustes nos parâmetros da llm e na quantidade de contexto transmitido ao modelo foram realizados para mitigar esses problemas.

A integração de múltiplas APIs, incluindo LangChain, Cohere, Hugging Face e Groq Cloud, demandou atenção especial à compatibilidade e ao correto aproveitamento das ferramentas, especialmente no que se refere às versões das bibliotecas utilizadas na implementação.

Outro ponto desafiador foi a persistência de histórico e a filtragem de contextos por documento, de modo que o chatbot respondesse apenas com base no material selecionado pelo usuário. A solução envolveu o uso combinado de identificadores únicos por sessão e um controle mais rigoroso sobre a indexação de vetores.

Como trabalhos futuros, prevê-se a realização de testes de desempenho, incluindo medições de tempo médio de resposta e acurácia semântica, além da implementação de testes de interface (E2E – End-to-End) para identificar possíveis falhas no frontend e testes de integração para assegurar a correta funcionalidade e a integridade do sistema.

Também poderá ser realizada a inclusão de novos modelos de linguagem com o objetivo de aprimorar a acurácia semântica, considerando que diferentes modelos apresentam capacidades distintas de compreensão e geração de linguagem natural. Além disso, pretende-se otimizar o desempenho do sistema, uma vez que modelos de maior porte podem gerar respostas mais precisas, porém com maior latência e custo computacional.

Outra oportunidade de melhoria consiste na implementação do envio de documentos em lote, visando maior objetividade e otimização do tempo, permitindo que consultas sejam realizadas simultaneamente em diversos documentos de um mesmo processo.

6. Resultados

6.1. Avaliação do chatbot

A análise das respostas geradas pelas duas configurações do chatbot, denominadas Adaptado (temperatura 0) e Padrão, demonstrou que a variação da temperatura não

produziu diferenças significativas no conteúdo obtido. Foram avaliadas 15 perguntas, cada uma respondida nas duas configurações, totalizando 15 pares de respostas comparáveis. Em 13 desses pares (86,67%), as respostas foram textualmente idênticas, indicando comportamento praticamente invariável independentemente da temperatura utilizada. Esses resultados sugerem que, para o tipo de conteúdo avaliado, predominantemente informativo e baseado em normas/portarias, a temperatura teve impacto mínimo, possivelmente devido à natureza determinística do contexto e ao *prompt* utilizado. O quadro completo pode ser acessado por meio do seguinte endereço: https://docs.google.com/spreadsheets/d/1u2Anv_k0s8OBfZha5yV46e_boRq46WeFYG3Nd1Ikvvw/edit?usp=sharing.

As perguntas foram classificadas nas categorias fácil, médio e difícil, entretanto, nenhuma delas apresentou desempenho superior às demais. No conjunto das 15 perguntas analisadas, verificou-se uma média de três respostas corretas por categoria, o que sugere que o nível de dificuldade atribuído às questões não exerceu influência significativa sobre a capacidade do modelo em recuperar as informações solicitadas. Cabe destacar que essa categorização foi realizada pela plataforma ChatGPT, utilizada como referência de benchmark devido à sua ampla adoção mundial em tarefas de processamento de linguagem natural.

6.2. Métricas do Kanban

O processo de desenvolvimento também foi analisado por meio das métricas extraídas do quadro Kanban. No período compreendido entre maio e agosto, foram identificadas 32 tarefas concluídas, distribuídas da seguinte forma: cinco em maio, nenhuma em junho, vinte e duas em julho e cinco em agosto, caracterizando o throughput total do processo. As métricas de Lead Time e Cycle Time indicam um fluxo de trabalho geralmente ágil, ainda que com certa variabilidade ao longo do período. O Lead Time apresentou média de 3,2 dias, enquanto o Cycle Time registrou média de 1,1 dias. A diferença relativamente pequena entre essas médias sugere um fluxo de desenvolvimento organizado, com poucos atrasos e manutenção de um ritmo consistente entre a criação e a execução das tarefas.

6.3. Conclusão

O desenvolvimento de um chatbot utilizando técnicas como *Retrieval-Augmented Generation* (*RAG*), aliado ao uso de bancos vetoriais, apresenta-se como uma solução eficaz para sistemas que dependem da recuperação de informações a partir de fontes externas, especialmente quando é necessário restringir o contexto apenas a documentos selecionados. Apesar de o uso de tecnologias de código aberto e de plataformas gratuitas impor limitações de desempenho, dado o custo envolvido na manutenção de sistemas robustos e de alta performance, o sistema desenvolvido cumpre adequadamente seu objetivo: permitir consultas confiáveis em documentos jurídicos.

Tabela 1. Quadro comparativo de perguntas e repostas. Fonte: autoria propria

Pergunta	Resposta	Configuração	Categoria
O edital refere-se a qual processo seletivo?	Processo Seletivo Unificado – PSU 2026/1.	Adaptado	Fácil
Quem é o responsável legal pela publicação do edital?	A informação necessária para responder a esta pergunta não está disponível no contexto fornecido.	Adaptado	Fácil
Qual documento o candidato deve ler completamente antes de se inscrever?	Termo de Consentimento de Coleta e Tratamento dos Dados Pessoais de Candidatos aos Cursos do Instituto Federal de Rondônia – IFRO.	Adaptado	Fácil
A inscrição pode ser feita presencialmente no campus?	Não, as inscrições serão realizadas exclusivamente on-line, no Portal Oficial do IFRO, no período estabelecido no ”ANEXO I – CRONOGRAMA”deste Edital.	Adaptado	Fácil
O candidato é responsável por verificar seus dados no formulário de inscrição?	A informação necessária para responder a esta pergunta não está disponível no contexto fornecido.	Adaptado	Fácil
Que tipo de comunicação o IFRO pode enviar ao candidato durante o processo?	O IFRO pode enviar comunicação pessoal dirigida ao candidato por e-mail, telefone, correio, SMS e/ou quaisquer aplicativos de comunicação instantânea.	Adaptado	Médio
Em qual situação o candidato deve apresentar o “Termo de Compromisso para Participar das Aulas Presenciais”?	No ato da matrícula.	Adaptado	Médio
O que deve ser apresentado para comprovar escolaridade integral em escola pública?	A informação necessária para responder a esta pergunta não está disponível no contexto fornecido.	Adaptado	Médio
O que acontece se o candidato não apresentar	A informação necessária para responder a esta	Adaptado	Médio

Referências

SUPREMO TRIBUNAL FEDERAL. STF divulga estatísticas do Poder Judiciário em 2023. Brasília: STF, 2023. Disponível em: ¡https://portal.stf.jus.br/noticias/verNoticiaDetalhe.asp?idConteudo=542620&ori=1¿. Acesso em: 13 out. 2025.

ORDEM DOS ADVOGADOS DO BRASIL. Perfil da advocacia brasileira 2023. Brasília: OAB, 2023. Disponível em: ¡https://www.oab.org.br/noticia/61715/oab-divulga-dados-ineditos-sobre-o-perfil-da-advocacia-brasileira¿. Acesso em: 13 out. 2025.

BARBIERI ADVOGADOS. IA na advocacia: como transformar seu escritório jurídico com inteligência artificial. São Paulo: Barbieri Advogados, 2023. Disponível em: ¡https://www.barbieriadvogados.com/ia-na-advocacia-como-transformar-seu-escritorio-juridico-com-inteligencia-artificial/¿. Acesso em: 13 out. 2025.

Russell, S. J., & Norvig, P. (2021). Artificial intelligence: A modern approach (4th ed.). Pearson.

MCCARTHY, John. A proposal for the Dartmouth Summer Research Project on Artificial Intelligence. Hanover, New Hampshire: Dartmouth College, 1956.

Heaton, J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genet Program Evolvable Mach 19, 305–307 (2016).

LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015).

Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Pearson.

Manning, C.D. and Schütze, H. (1999) Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA.

Vaswani, A., et al. (2017) Attention Is All You Need. Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, 4-9 December 2017, 6000-6010.

BROWN, T. et al. Language models are few-shot learners. In: advances in neural information processing systems, v. 33, 2020.

Ferrucci, David & Brown, Eric & Chu-Carroll, Jennifer & Fan, James & Gondek, David & Kalyanpur, Aditya & Lally, Adam & Murdock, J William & Nyberg, Eric & Prager, John & Schlaefer, Nico & Welty, Christopher. (2010). Building Watson: An Overview of the DeepQA Project. AI Magazine. 31. 59-79. 10.1609/aimag.v31i3.2303.

Weizenbaum, J. (1966) ELIZA—A Computer Program for the Study of Natural Language Communication between Man and Machine. Communications of the ACM, 9, 36-45. https://doi.org/10.1145/365153.365168

Morley, J., Floridi, L., Kinsey, L., & Elhalal, A. (2020). From What to How: An Initial Review of Publicly Available AI Ethics Tools, Methods and Research to Translate Principles into Practices. Science and Engineering Ethics, 26, 2141-2168. https://doi.org/10.1007/s11948-019-00165-5

FINARDI, P. et al. The Chronicles of RAG: The Retriever, the Chunk and the

Generator. 15 jan. 2024.

GAO, Y. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. 18 dez. 2023.

LEWIS, P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 22 maio 2020.

KARPUKHIN, V. et al. Dense Passage Retrieval for Open-Domain Question Answering. 10 abr. 2020.

TAIPALUS, T. Vector Database Management Systems: Fundamental Concepts, Use-Cases, and Current Challenges. 2023.

HAN, Y.; LIU, C.; WANG, P. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge. 2023.

MIKOLOV, T. et al. Efficient Estimation of Word Representations in Vector Space. 2013.

PINECONE. Pinecone Vector Database Documentation. 2023.

ROY, D. Word Embeddings and Information Retrieval. Kolkata: Indian Statistical Institute, 2019.

Binder CR, Hinkel J, Bots PWG, Pahl-Wostl C (2013) Comparison of Frameworks for analyzing social-ecological systems. Ecol Soc 18(4):26. https://doi.org/10.5751/ES-05551-180426