UNIVERSIDADE FEDERAL DO PARÁ

FACULDADE DE COMPUTAÇÃO

CAMPUS UNIVERSITÁRIO DE CASTANHAL

ARIDAN SILVA PANTOJA

HARPIA: CHATBOT INTELIGENTE PARA APOIO A PROCESSOS SELETIVOS NA UFPA

Castanhal-PA

2025

UNIVERSIDADE FEDERAL DO PARÁ
FACULDADE DE COMPUTAÇÃO
CAMPUS UNIVERSITÁRIO DE CASTANHAL

ARIDAN SILVA PANTOJA
Orientador: Dra.Yomara Pinheiro Pires

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação do Campus de Castanhal da Universidade Federal do Pará, como requisito para obtenção do grau de Bacharel em Engenharia de Computação.
Professor: Dra.Yomara Pinheiro Pires

Castanhal- PA
2025

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE CASTANHAL
FACULDADE DE COMPUTAÇÃO

Harpia: Chatbot Inteligente para Apoio a Processos Seletivos na UFPA

Autor: ARIDAN SILVA PANTOJA

Monografia apresentada à Faculdade de Computação e aprovada para a obtenção do título de Bacharel em Engenharia de Computação.

APROVADO EM: 16 de setembro de 2025.

Banca Examinadora:

Profª. Drª. Yomara Pinheiro Pires
Orientador FACOMP/CCAST/UFPA

Prof. Drº. Igor Ruiz Gomes
(Avaliador Interno – FACOMP/CCAST/UFPA)

Prof. Drº. José Jailton Henrique Ferreira Júnior
(Avaliador Interno – FACOMP/CCAST/UFPA)

Visto:

Profª. Drª. Yomara Pinheiro Pires
(Diretora da Faculdade de Computação/CCAST/UFPA)

AGRADECIMENTOS

Obrigado aos meus pais, Divania e Júnior! Vocês são incríveis e eu tenho o maior orgulho do mundo em ser filho dos dois. Obrigado à Beatriz por tanta paciência e por estar ao meu lado nos momentos mais complicados. Obrigado a todos os meus amigos que sempre estão comigo, me apoiando. Obrigado pela oportunidade de me deixarem ser eu mesmo e por torcerem por mim. Amo todos vocês!

Resumo

As instituições de ensino superior públicas enfrentam desafios na gestão de seus processos seletivos, gerando sobrecarga para servidores e dificultando o acesso à informação para candidatos. Este trabalho objetivou desenvolver e aprimorar a Harpia, um chatbot inteligente de código aberto integrado a Modelos de Linguagem de Grande Escala (LLMs) para automatizar o suporte nos processos seletivos da Universidade Federal do Pará (UFPA). A metodologia, de natureza aplicada e qualitativa, combinou o desenvolvimento tecnológico com uma análise comparativa da evolução do sistema. A solução, construída sobre uma arquitetura modular, emprega a técnica de Geração Aumentada por Recuperação (RAG), permitindo que o chatbot gere respostas precisas e contextualizadas, estritamente ancoradas em documentos oficiais. Como resultado, a nova versão superou as limitações do protótipo inicial, garantindo maior precisão e confiabilidade nas respostas, que passaram a incluir as fontes consultadas. A implementação de um painel administrativo e a melhoria na interface do usuário também foram bem-sucedidas. Conclui-se que a Harpia se consolida como uma ferramenta estratégica que otimiza a comunicação e a eficiência operacional, contribuindo com uma solução robusta e escalável, com potencial para futuras expansões, como a integração com outros canais de atendimento.

Palavras-chave: *Chatbot, Inteligência Artificial, Modelos de Linguagem de Grande Escala, Geração Aumentada por Recuperação, Processos Seletivos.*

Abstract

Public higher education institutions face challenges in managing their admission processes, which generates an increased workload for administrative staff and hinders candidates' access to information. This study aimed to develop and enhance Harpia, an open-source intelligent chatbot integrated with Large Language Models (LLMs), to automate support for the admission processes at the Federal University of Pará (UFPA). The methodology, applied and qualitative in nature, combined technological development with a comparative analysis of the system's evolution. The solution, built on a modular architecture, employs the Retrieval-Augmented Generation (RAG) technique, enabling the chatbot to generate precise and contextualized responses strictly grounded in official documents. As a result, the new version overcame the limitations of the initial prototype, ensuring greater precision and reliability in its responses, which now include citations of the consulted sources. The implementation of an administrative dashboard and improvements to the user interface were also successful. In conclusion, Harpia stands as a strategic tool that optimizes communication and operational efficiency, providing a robust and scalable solution with potential for future expansions, such as integration with other support channels.

Keywords: *Chatbot, Artificial Intelligence, Large Language Models, Retrieval-Augmented Generation, Admission Processes.*

Lista de Figuras

Lista de Tabelas

Sumário

1 INTRODUÇÃO

1.1 Contexto

A gestão universitária nas instituições federais brasileiras é frequentemente marcada por características burocráticas, como estruturas administrativas complexas e uma notável rigidez normativa, que dificultam a implementação de mudanças e inovações (FALQUETO; FARIAS, 2013). Esse desafio estrutural se estende a diversos procedimentos, tornando-se especialmente evidente na gestão dos processos seletivos em universidades de grande porte como a Universidade Federal do Pará (UFPA). Nesses processos, os desafios são amplificados pelo volume expressivo de candidatos e pelas múltiplas etapas que demandam coordenação precisa entre diferentes setores, desde a publicação de editais até a divulgação dos resultados finais (HECK et al., 2018).

A gestão tradicional desses processos seletivos caracteriza-se pela dependência de métodos manuais e comunicação fragmentada entre os diversos atores envolvidos: candidatos, docentes, servidores técnico-administrativos, bolsistas e gestores. Esta abordagem convencional resulta frequentemente em sobrecarga das equipes responsáveis, que precisam lidar simultaneamente com grande volume de consultas, esclarecimento de dúvidas, processamento de documentação e cumprimento de prazos rigorosos estabelecidos pelos órgãos regulamentadores da educação superior, o que contribui para o aumento da pressão psicológica e da carga de trabalho dos servidores técnico-administrativos, podendo gerar sofrimento e até adoecimento, conforme evidenciado por Vieira et al. (2018)

No cenário atual da educação superior brasileira, verifica-se uma crescente demanda por transparência, eficiência e acessibilidade nos processos administrativos das universidades públicas, impulsionada pelas políticas de democratização do acesso ao ensino superior e pela obrigatoriedade de prestar contas à sociedade sobre o uso de recursos públicos, conforme aponta o estudo de Lima et al. (2025), que avalia a qualidade das informações veiculadas nos portais de transparência das universidades federais brasileiras e destaca a importância de critérios de qualidade na disponibilização dessas informações.

No contexto específico dos processos seletivos, a tecnologia emerge como ferramenta fundamental para automatizar tarefas repetitivas, centralizar informações e facilitar o acesso dos candidatos aos dados relevantes sobre os processos em andamento, melhorar a eficiência operacional, reduzir custos e aprimorar a experiência dos usuários (PEYTON et al., 2025).

1.2 Justificativa

A motivação para o desenvolvimento de uma solução tecnológica voltada para a gestão de processos seletivos na UFPA fundamenta-se nas limitações identificadas no modelo atual de gestão e na necessidade de modernização dos serviços oferecidos pela instituição. A análise das dificuldades enfrentadas pelos candidatos e pela administração universitária revela oportunidades significativas de melhoria através da implementação de ferramentas de automação e inteligência artificial (IA) (CHINIMILLI; SADASIVUNI, 2024).

Um dos principais problemas identificados refere-se à dificuldade de acesso às informações pelos candidatos, que frequentemente estão dispersas em sites institucionais, editais extensos e comunicados administrativos, sobrecarregando os setores administrativos e comprometendo a qualidade do atendimento (FALQUETO; FARIAS, 2013). Além disso, editais podem conter informações dispersas, termos técnicos ou regras complexas, dificultando a compreensão do que é exigido (ROSA, 2024), e a ausência ou excesso de detalhes sobre critérios de avaliação e procedimentos administrativos pode deixar os candidatos em situação de vulnerabilidade (DIAS, 2023).

Ademais, a sobrecarga de trabalho dos servidores técnico-administrativos representa outro aspecto crítico que justifica a implementação de soluções automatizadas. Durante os períodos de processos seletivos, os profissionais responsáveis precisam dedicar considerável parcela de seu tempo ao esclarecimento de dúvidas rotineiras, respondendo repetidamente às mesmas questões sobre prazos, documentação, critérios de seleção e procedimentos administrativos (VIEIRA et al., 2018). Esta situação reduz a disponibilidade dos servidores para atividades mais estratégicas e complexas que demandam análise especializada e tomada de decisões.

A necessidade de garantir a equidade no acesso às informações constitui aspecto fundamental na justificativa desta proposta. Em uma universidade pública, é essencial assegurar que todos os candidatos tenham acesso igualitário às informações necessárias para participação nos processos seletivos, independentemente de sua localização geográfica, disponibilidade de horário ou familiaridade com os procedimentos institucionais (LAVIGNE, 2024). A implementação de um sistema automatizado de atendimento pode contribuir significativamente para a democratização do acesso às informações, oferecendo suporte contínuo e padronizado a todos os interessados.

Logo, do ponto de vista econômico, a automação de processos administrativos representa oportunidade de otimização de recursos humanos e financeiros. A redução do tempo dedicado ao atendimento de consultas rotineiras permite que os servidores concentrem seus esforços em atividades de maior valor agregado, melhorando a eficiência global dos setores envolvidos na gestão de processos seletivos (ADAMO-

POULOU; MOUSSIADES, 2020b).

1.3 Objetivo

1.3.1 Objetivo Geral

Desenvolver o chatbot Harpia, integrado a LLMs, como solução aberta e flexível para otimizar o suporte e a comunicação nos processos seletivos da UFPA.

1.3.2 Objetivos Específicos

Para alcançar o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

1. Projetar a arquitetura do chatbot Harpia como solução *open-source*, modular, escalável e segura, permitindo integração com sistemas existentes e adaptação a diferentes contextos.
2. Desenvolver uma interface de usuário acessível e intuitiva, garantindo usabilidade, responsividade e experiência positiva em múltiplos dispositivos.
3. Sistematizar e organizar as informações institucionais dos processos seletivos (editais, cronogramas, critérios, FAQ), assegurando atualização contínua e respostas rápidas baseadas em fontes oficiais.
4. Integrar uma Linguagem de Grande Escala (do inglês, Large Language Models ou LLMs) para interpretação de linguagem natural, garantindo respostas precisas, contextuais e alinhadas ao banco de conhecimento da UFPA.
5. Aplicar boas práticas de engenharia de software, assegurando qualidade, segurança, manutenibilidade e evolução futura do sistema.

2 FUNDAMENTAÇÃO TEÓRICA

Com o avanço da IA e o crescimento exponencial de dados, o desenvolvimento de soluções inteligentes baseadas em modelos de linguagem de grande escala (LLMs) tem se tornado essencial. No contexto deste trabalho, que visa criar um chatbot inteligente para auxiliar atividades organizacionais durante os processos seletivos da UFPA, é fundamental compreender os fundamentos teóricos relacionados à Inteligência Artificial (IA), Processamento de Linguagem Natural (PLN), modelos generativos e estratégias como *Retrieval-Augmented Generation* (RAG), além das melhores práticas em engenharia de software e desenvolvimento web.

2.1 Inteligência Artificial (IA)

A Inteligência Artificial (IA) pode ser definida como o campo da ciência da computação dedicado ao desenvolvimento de sistemas capazes de simular a maneira como a inteligência humana funciona (CHINIMILLI; SADASIVUNI, 2024). Assim, a IA engloba a criação de agentes que percebem o ambiente, raciocinam sobre ele e tomam decisões de forma autônoma, destacando que a IA não se restringe a tarefas específicas, mas busca replicar, em alguma medida, a capacidade humana de aprendizado, adaptação e solução de problemas (LAN et al., 2024).

Dentre as abordagens mais estudadas na IA, o aprendizado supervisionado é aquele em que os modelos são treinados a partir de exemplos rotulados, permitindo que aprendam a mapear entradas a saídas conhecidas, sendo amplamente utilizada em tarefas de classificação de documentos, reconhecimento de imagens e análise de sentimentos, pois fornece ao sistema a referência necessária para ajustar seus parâmetros de forma precisa (ALVES et al., 2023).

O aprendizado não supervisionado busca identificar padrões em dados não rotulados, explorando relações intrínsecas sem depender de respostas pré-definidas (ALVES et al., 2023). Técnicas como agrupamento (*clustering*) e redução de dimensionalidade permitem que sistemas descubram categorias, tendências ou anomalias. Já o aprendizado por reforço foca em agentes que aprendem por meio de recompensas ou penalidades durante a interação com o ambiente, sendo útil em problemas dinâmicos e sequenciais, como jogos, controle robótico e navegação autônoma (BRITO et al., 2023).

Um dos campos mais relevantes da IA para este trabalho é o PLN, que permite que sistemas compreendam, interpretem e gerem linguagem humana, sendo fundamental para que chatbots, como o proposto para os processos seletivos da UFPA, consigam

interpretar perguntas formuladas de diferentes maneiras, acessar informações de bases de conhecimento externas e gerar respostas coerentes, precisas e contextualizadas (PEREIRA, 2019).

2.2 Processamento de Linguagem Natural (PLN)

O PLN é um campo da IA voltado para o desenvolvimento de sistemas capazes de compreender, interpretar e gerar linguagem humana (ADAMOPOULOU; MOUSSIADES, 2020a, 2020b). Sua importância cresce na medida em que a comunicação automatizada se torna essencial em diferentes setores, incluindo atendimento ao cliente, educação e serviços administrativos. O histórico do PLN está intimamente ligado à evolução dos chatbots: o primeiro sistema notável, ELIZA, criado por Joseph Weizenbaum em 1966, simulava um psicoterapeuta utilizando padrões de palavras-chave para responder às perguntas dos usuários (ADAMOPOULOU; MOUSSIADES, 2020a).

Desde então, outros chatbots foram desenvolvidos, como PARRY, A.L.I.C.E. e SmarterChild, evoluindo gradualmente para sistemas capazes de compreender contexto e intencionalidade (ADAMOPOULOU; MOUSSIADES, 2020b). Atualmente, chatbots aplicados à educação e aos processos administrativos, como o proposto para os processos seletivos da UFPA, são ferramentas estratégicas para fornecer informações rápidas, automatizar procedimentos burocráticos e interagir de forma personalizada com os usuários (PANTOJA et al., 2025).

Entre as técnicas clássicas de PLN, destacam-se a *tokenização*, que divide o texto em unidades compreensíveis pelo modelo; os *embeddings*, que representam palavras e frases em vetores numéricos preservando relações semânticas; e métodos de análise sintática e semântica, que permitem a compreensão da estrutura e do significado das sentenças (MINAEE et al., 2024). Esses métodos fornecem a base para que os sistemas interpretem perguntas e instruções, facilitando a construção de respostas coerentes e contextualizadas em chatbots, tradutores automáticos, sistemas de análise de sentimentos e assistentes virtuais (ADAMOPOULOU; MOUSSIADES, 2020a).

Os avanços recentes em *deep learning* revolucionaram o PLN, com o surgimento de *word embeddings*, arquiteturas *Transformer* e LLMs, que conseguem processar e gerar texto de forma mais sofisticada, permitindo que os chatbots compreendam contextos complexos, sigam instruções específicas e produzam respostas mais naturais, precisas e adaptadas às necessidades dos usuários (ADAMOPOULOU; MOUSSIADES, 2020b). Para a Harpia, a utilização dessas ferramentas possibilita que o chatbot para processos seletivos da UFPA interprete diferentes formas de perguntas dos candidatos e da equipe técnico-administrativa, fornecendo respostas consistentes e base-

adas em informações atualizadas, tornando a experiência de interação mais eficiente e confiável.

2.2.1 Modelos de Linguagem de Grande Escala (LLMs)

Os Modelos LLMs representam um avanço significativo no campo da IA, destacando-se por sua performance robusta em uma vasta gama de tarefas relacionadas ao processamento de linguagem natural, especialmente desde o lançamento do ChatGPT em 2022 (MINAEE et al., 2024). Nesse contexto, os LLMs se tornaram a tecnologia central que possibilita a criação de assistentes conversacionais sofisticados, como o chatbot proposto neste trabalho para apoiar os processos seletivos da UFPA, sendo capaz de interagir de forma natural e informativa com os candidatos e técnicos administrativos.

Segundo Minaee et al. (2024) e Shao et al. (2024), um LLM pode ser definido como um modelo baseado na arquitetura Transformer, que possui bilhões de parâmetros e é pré-treinado em grandes volumes de dados textuais. O pilar dessa arquitetura é o mecanismo de autoatenção (self-attention), que permite ao modelo analisar todas as palavras de uma sequência simultaneamente, possibilitando a identificação de relações contextuais e padrões complexos da linguagem de forma muito eficiente (PEYKANI et al., 2025). O poder desses modelos é então amplificado por seus bilhões de parâmetros, que são os valores internos ajustados durante o treinamento, sendo através deles que o LLM aprende a prever palavras, gerar textos coerentes e capturar nuances semânticas em diferentes contextos, armazenando um vasto conhecimento sobre a linguagem e o mundo (MINAEE et al., 2024).

O treinamento em escala web confere aos LLMs habilidades emergentes, como o aprendizado em contexto, a capacidade de seguir instruções complexas e o raciocínio de múltiplos passos (PEYKANI et al., 2025). Para o chatbot de processos seletivos de uma universidade, essa capacidade generalista é fundamental, pois ele precisa compreender perguntas formuladas de inúmeras maneiras tanto por candidatos quanto pela equipe técnico-administrativa — desde dúvidas sobre o cronograma do vestibular e a documentação para matrícula, até consultas específicas sobre o número de vagas para cada região — e, a partir disso, gerar respostas coesas e precisas, sem ter sido explicitamente treinado para cada pergunta possível.

Portanto, apesar de seu poder, o uso de LLMs em contextos institucionais, como processos seletivos, é limitado por desafios que nascem de sua própria concepção (PEYKANI et al., 2025). Sua natureza probabilística, e não factual, pode levar a alucinações nas respostas; seu treinamento estático resulta em conhecimento desatualizado sobre editais; e o reflexo dos dados da internet introduz vieses que comprometem

a equidade do processo (SHARMA, 2025). Sendo assim, essas limitações tornam o uso de um LLM ”puro”inadequado para um ambiente institucional, criando a necessidade de arquiteturas que ancorem suas respostas em uma base de conhecimento externa, confiável e atualizada, como os editais e regulamentos oficiais da universidade (SHARMA, 2025).

2.2.2 Geração Aumentada por Recuperação (RAG)

A Geração Aumentada por Recuperação (do inglês, *Retrieval-Augmented Generation* ou RAG) é uma técnica utilizada para superar as limitações intrínsecas dos LLMs, permitindo que o modelo utilize informações externas durante a geração de respostas. Ao combinar modelos de linguagem com mecanismos de recuperação de dados, o RAG torna as respostas mais precisas, confiáveis e contextualmente relevantes, incorporando conhecimento atualizado a partir de fontes externas (GUPTA et al., 2024).

O objetivo central do RAG é mitigar limitações críticas dos LLMs, como alucinações, ao apoiar a geração em dados verificáveis; conhecimento desatualizado, permitindo o acesso a informações recentes; falta de transparência, possibilitando a citação das fontes; e dificuldades em domínios especializados, integrando bases externas de conteúdo específico (GAO et al., 2024). Assim, essa abordagem é especialmente relevante em contextos institucionais, como o chatbot para processos seletivos universitários, garantindo respostas precisas, atualizadas e rastreáveis sobre editais, prazos e documentação.

No contexto dos chatbots, o RAG pode ser implementado por meio de *tools*, que permitem ao LLM executar funções como requisições a APIs ou buscas de dados, sem se limitar apenas à recuperação de informações (MINAEE et al., 2024). Dessa forma, o modelo decide quando acionar a *tool*, garantindo que a função seja executada e os resultados obtidos. Na Harpia, por exemplo, uma *tool* permite recuperar trechos de editais e regulamentos oficiais, incorporando informações confiáveis sempre que o usuário solicita dados específicos.

Para recuperar informações de forma eficiente, o RAG utiliza a busca híbrida (*Hybrid Search*), que combina busca por palavras-chave e busca semântica baseada em embeddings, gerando um ranqueamento próprio dos documentos relevantes, captando tanto correspondências exatas de termos quanto relações semânticas e sinônimos (Supabase Docs, 2025). Em seguida, esses rankings são combinados pelo *Reciprocal Rank Fusion* (RRF), uma técnica que calcula a pontuação final inversamente proporcional à posição de cada documento em cada ranking, priorizando textos que aparecem de forma consistente entre as listas (SHARMA, 2025). Dessa maneira, o chatbot integra informações precisas, atualizadas e contextualmente relevantes, aumentando a

confiabilidade das respostas durante os processos seletivos.

2.3 Engenharia de Software Aplicada ao Desenvolvimento Web

A Engenharia de Software é a área que aplica princípios de engenharia a uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de software, visando à produção de sistemas de alta qualidade de maneira eficiente (VALENTE, 2020). No cenário atual do desenvolvimento web, a aplicação desses princípios é crucial para construir sistemas modulares, escaláveis e de fácil manutenção, englobando a tomada de decisões estratégicas sobre padrões de arquitetura e a utilização de frameworks modernos que aceleram o desenvolvimento e garantem a robustez das aplicações (VALENTE, 2020).

No desenvolvimento web moderno, tecnologias como React e Next.js destacam-se como ferramentas eficientes para a construção de interfaces de usuário dinâmicas e de alto desempenho (PATI; ZAKI, 2025). O React serve como base para a criação de componentes de UI modulares e reutilizáveis, enquanto o *Next.js* estende essa capacidade com recursos avançados como a renderização no lado do servidor (do inglês, *Server Side Rendering* ou SSR) e a geração de sites estáticos (do inglês, *Static Site Generation* ou SSG), que otimizam significativamente o desempenho e a experiência do usuário (CHEN, 2025).

Em complemento, o *TypeScript* tem desempenhado um papel essencial na transformação do ecossistema *JavaScript*, tornando-se um padrão no desenvolvimento web moderno (BOGNER; MERKEL, 2022). O estudo empírico de Bogner e Merkel (2022) analisou milhares de repositórios no GitHub e concluiu que o uso do *TypeScript* melhora a qualidade do código, a manutenibilidade e a consistência entre equipes, ainda que não elimine totalmente a ocorrência de defeitos.

Portanto, a aplicação de princípios de Engenharia de Software, aliada ao uso de tecnologias modernas como *React*, *Next.js* e *TypeScript*, bem como à adoção de boas práticas, arquiteturas bem definidas e ferramentas adequadas, permite a construção da Harpia como uma aplicação modular, customizável e de fácil manutenção, capaz de oferecer respostas precisas e integrar diferentes funcionalidades como uma ferramenta open-source prática e eficiente.

3 METODOLOGIA

3.1 Abordagem Metodológica

A metodologia desta pesquisa foi projetada para orientar o desenvolvimento e aprimoramento de um chatbot destinado a auxiliar candidatos e técnicos administrativos nos processos seletivos da Universidade Federal do Pará (UFPA). O objetivo do sistema é atender às demandas informacionais de dois públicos distintos: candidatos que buscam esclarecimentos sobre etapas, documentação e prazos; e servidores que necessitam de uma ferramenta para otimizar o atendimento e reduzir a sobrecarga de trabalho.

Trata-se de uma pesquisa aplicada, pois propõe um produto tecnológico para solucionar um problema prático identificado no contexto institucional: a recorrência de dúvidas e as dificuldades de comunicação nos processos seletivos. A abordagem é predominantemente qualitativa, centrada na análise da evolução funcional e da arquitetura do sistema, bem como na avaliação de seus benefícios. O caráter qualitativo se aprofunda no exame das percepções registradas em um artigo prévio sobre a primeira versão do chatbot e na análise comparativa entre as duas versões, que foca na descrição de funcionalidades e na evolução metodológica. Futuramente, métricas quantitativas poderão ser exploradas para ampliar a avaliação de desempenho.

O método adotado combina desenvolvimento tecnológico e análise comparativa. O desenvolvimento tecnológico contempla o aprimoramento da ferramenta para entregar um produto robusto e alinhado às necessidades da UFPA, cuja confiabilidade das informações é assegurada pelo uso de documentos oficiais como base de conhecimento. A análise comparativa, por sua vez, constitui o eixo central da investigação, destacando a evolução entre a versão atual e a anterior. Assim, a pesquisa não se limita à simples construção de um protótipo, mas busca compreender o processo evolutivo da solução, identificando melhorias implementadas, limitações superadas e novos desafios.

Por fim, a escolha metodológica garante que os resultados não apenas apresentem um produto funcional, mas também demonstrem sua contribuição prática e acadêmica. A análise crítica entre as duas versões do chatbot oferece subsídios para aprimoramentos futuros, consolidando a proposta como uma base sólida para o desenvolvimento de soluções tecnológicas de suporte informacional em instituições de ensino superior.

3.2 Arquitetura da Aplicação

A arquitetura da aplicação foi concebida com base em um modelo modular e escalável, projetado para assegurar alto desempenho, manutenibilidade e flexibilidade na

integração com diversos serviços e modelos de inteligência artificial. Para garantir a qualidade e a robustez do código, o *TypeScript* foi adotado como linguagem padrão, oferecendo um sistema de tipagem estática que é essencial para maximizar a segurança, a legibilidade e a produtividade da equipe, permitindo a detecção de erros ainda em tempo de desenvolvimento (MICROSOFT, 2025).

Essa organização é estruturada em quatro camadas principais, cada uma com responsabilidades bem definidas. Dessa forma, a separação em camadas favorece a evolução independente de cada componente, reduz o acoplamento e simplifica futuras manutenções:

- **Camada de Apresentação (Front-end):** Responsável pela interação com o usuário e pela entrega de uma experiência eficiente e responsiva.
- **Camada de Lógica e Integração (Back-end):** Centraliza regras de negócio, orquestra fluxos e integra serviços externos e modelos de IA.
- **Camada de Persistência de Dados e Armazenamento:** Gerencia bancos de dados, indexação de conteúdos e soluções de armazenamento de arquivos.
- **Camada de Processamento e Inserção de Documentos:** Realiza o recebimento, segmentação, extração e preparação de documentos para indexação e consulta.

A Figura 1 ilustra o relacionamento entre as camadas, demonstrando como a Camada de Lógica atua como intermediária entre o usuário e os dados, enquanto a Camada de Inserção opera de forma independente, garantindo alta performance na preparação e integração de conteúdos.

Fonte: Elaborado pelo autor.

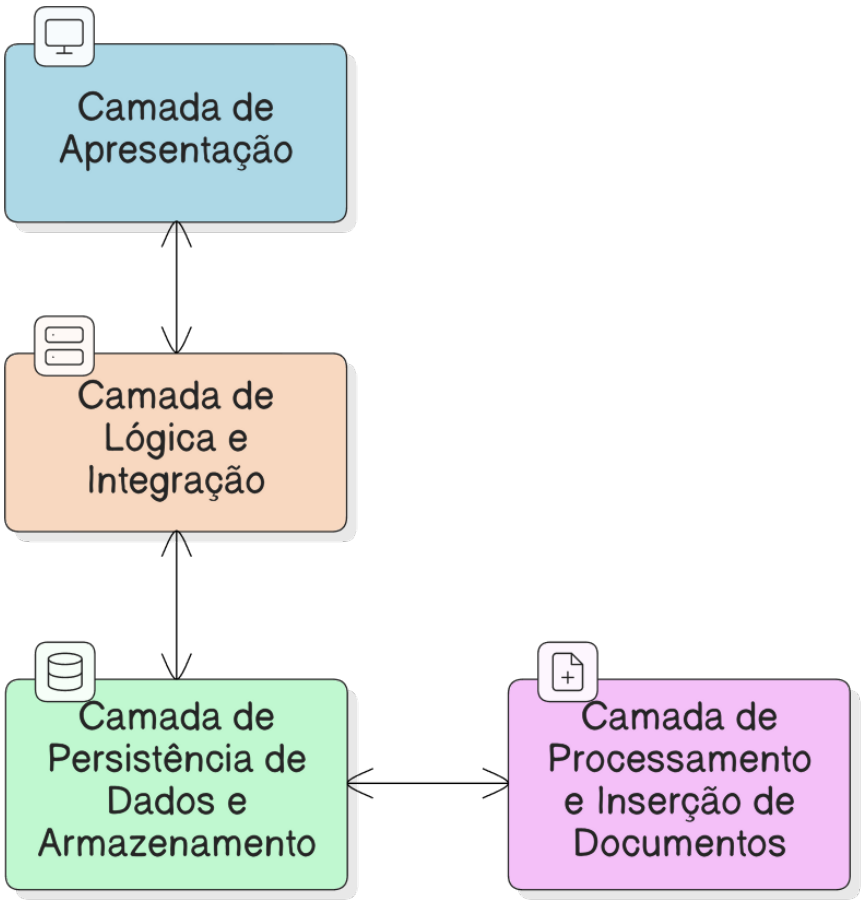

Figura 1: Diagrama da Arquitetura em Camadas da Aplicação

A seguir, são detalhadas as camadas e suas respectivas tecnologias.

3.2.1 Camada de Apresentação (Front-end)

A camada de apresentação é responsável por intermediar a comunicação entre o usuário e o sistema, oferecendo uma experiência fluida, acessível e responsiva. Para atender a esses requisitos, foi utilizado o *Next.js*, um *framework* baseado em *React* que facilita a criação de interfaces modulares, escaláveis e reutilizáveis (SOURCE, 2025; VERCEL, 2025a).

O *Next.js* fornece suporte nativo para *Server-Side Rendering* (SSR) e *Static Site Generation* (SSG), permitindo carregamentos mais rápidos e otimização do SEO. Essa abordagem possibilita melhor experiência ao usuário e maior eficiência no processamento de páginas dinâmicas.

Para a estilização, foi adotado o *Tailwind CSS*, que utiliza classes utilitárias para acelerar o desenvolvimento, promover padronização visual e facilitar a manutenção

da interface (LABS, 2025). A biblioteca *Shadcn/UI* (SHADCN, 2025) complementa o design com uma base de componentes acessíveis e personalizáveis, aplicados tanto na construção do *design system* quanto na criação de elementos específicos para interfaces conversacionais.

Além disso, foi integrada a biblioteca *Vercel AI Elements* (VERCEL, 2025b), que disponibiliza componentes prontos para aplicações com inteligência artificial, permitindo maior flexibilidade na composição da interface e melhorando a experiência de uso em interações com modelos generativos.

3.2.2 Camada de Lógica e Integração (Back-end)

A camada de lógica centraliza as regras de negócio, gerencia fluxos da aplicação e integra os serviços utilizados. Para isso, o próprio *Next.js* foi adotado com suporte a rotas de API nativas, permitindo a criação de endpoints otimizados para consultas, processamento de dados e comunicação com modelos de IA.

Na autenticação, utilizou-se o Clerk (INC., 2025), um provedor moderno que suporta múltiplos métodos de login, incluindo *Google*, *GitHub* e outros provedores via *OAuth*, além de oferecer componentes prontos e APIs simplificadas para integração. Após a autenticação, os usuários com privilégios administrativos têm acesso ao painel administrativo, que centraliza funcionalidades para o acompanhamento das métricas do sistema, como acompanhamento de mensagens, histórico de conversas, documentos na base de conhecimento, permitindo uma análise eficiente sobre o uso da Harpia.

Para orquestrar as interações com modelos de IA, foi integrado o Vercel AI SDK (VERCEL, 2025c), que centraliza o gerenciamento de fluxos conversacionais e a troca de mensagens com os modelos. Foram empregados dois modelos principais: o GPT-5.0 Nano (OPENAI, 2025), responsável por gerar metadados inteligentes, como categorias e descrições de documentos, e o Gemini Flash 2.5 (DEEPMIND, 2025), escolhido por sua alta velocidade de processamento e capacidade de lidar com grandes volumes de dados.

Para otimizar chamadas de API e monitorar custos, foi implementado o *Cloudflare AI Gateway*, que atua como camada de proteção contra abusos e garante maior eficiência na utilização dos modelos (CLOUDFLARE, 2025).

3.2.3 Camada de Persistência de Dados e Armazenamento

A camada de persistência é responsável pela gestão eficiente de dados estruturados, vetoriais, metadados associados e arquivos brutos, tendo como base o PostgreSQL, escolhido por sua robustez e capacidade de lidar com grandes volumes de

informações (GROUP, 2025b). Foram aplicados recursos avançados, como o pgvector, que permite o armazenamento e a busca eficiente de embeddings vetoriais para recuperação semântica de documentos (GROUP, 2025a), e o tsvector, que otimiza buscas textuais, permitindo consultas por palavras-chave (GROUP, 2025c).

Para abstrair consultas e gerenciar o versionamento do banco, foi adotado o *Drizzle ORM*, que oferece integração nativa com *TypeScript*, facilitando a manutenção do esquema e garantindo tipagem completa nas consultas (TEAM, 2025).

Complementarmente, foi adotado um sistema de armazenamento de objetos compatível com *S3*, disponibilizado pelo *Supabase*, permitindo armazenar PDFs e outros tipos de arquivos de forma escalável e segura. Essa abordagem modular possibilita integrar ou migrar facilmente para outros provedores compatíveis, garantindo flexibilidade e independência de fornecedor.

3.2.4 Camada de Processamento e Inserção de Documentos

Esta camada foi projetada para processar, organizar e preparar documentos para indexação e consulta eficiente, utilizando o modelo text-embedding-3-small (OPENAI, 2025) para gerar embeddings semânticos de 1536 dimensões, permitindo uma busca híbrida que combina filtros tradicionais com similaridade semântica via pgvector. Para o processamento, foram empregadas bibliotecas como pdf-parse (AUTOKENT, 2018), para extração de texto de arquivos PDF; langchain (LANGCHAIN, 2022a), para manipulação avançada dos documentos e aplicação de técnicas de segmentação, como o RecursiveCharacterTextSplitter (LANGCHAIN, 2022b); e Drizzle ORM, para registro e versionamento dos documentos processados no banco de dados.

3.3 Modelagem do Banco de Dados

A modelagem dos dados foi desenvolvida utilizando o *Drizzle ORM*, ferramenta que proporciona flexibilidade para adaptação a diferentes sistemas gerenciadores de banco de dados e que facilita o versionamento do *schema* por meio de scripts de migração (TEAM, 2025). A estrutura resultante foi organizada em cinco tabelas principais: `file`, `file_chunk`, `chat`, `message` e `vote`. As duas primeiras estão relacionadas à base de conhecimento do chatbot, enquanto as três últimas concentram as informações referentes à utilização do sistema e às interações realizadas com a Harpia.

A Tabela `file`, cujo código de definição é apresentado na Figura 2, armazena os metadados dos documentos que compõem a base de conhecimento. Seus campos incluem: um identificador único (`id`), a `url` exclusiva do arquivo no serviço de armazenamento de objetos, os campos textuais `category` e `description` (gerados automa-

ticamente por inteligência artificial durante o *upload*), o `filename` original, o número total de páginas (`pages`) e um campo do tipo JSONB (`metadata`) contendo informações adicionais, como autor e data de criação. O código implementado com o *Drizzle ORM* define cada campo com seu tipo, restrições e valores padrão, além de impor unicidade para `url` e `filename`.

Fonte: Elaborado pelo autor.

```
export const file = pgTable('file', {
  id: uuid('id').primaryKey().defaultRandom(),
  url: text('url').notNull().unique(),
  category: text('category').notNull(),
  description: text('description').notNull(),
  filename: text('filename').notNull().unique(),
  pages: bigint('pages', { mode: 'number' }).notNull(),
  metadata: jsonb('metadata').notNull(),
  createdAt: timestamp('created_at').defaultNow().notNull(),
  updatedAt: timestamp('updated_at').defaultNow().notNull(),
});
```

Figura 2: Código utilizado para criação da tabela `file`

Complementando essa estrutura, a Tabela `file_chunk` (Figura 3) funciona como repositório central dos "fragmentos de conhecimento" extraídos dos documentos. Contém, além do identificador único e do `content`, campos para `metadata` (JSONB), `fts` (do tipo `tsvector`, gerado automaticamente para otimizar buscas textuais) e `embedding` (vetor de 1536 dimensões para busca semântica). Possui ainda a chave estrangeira `file_id`, que vincula cada fragmento ao arquivo de origem. Para atender ao requisito de alta performance, foram criados dois índices estratégicos: um índice GIN sobre `fts` e um índice HNSW sobre `embedding`, otimizando respectivamente as buscas por palavra-chave e por similaridade vetorial.

Fonte: Elaborado pelo autor.

```
export const fileChunk = pgTable(
  'file_chunk',
  {
    id: uuid('id').primaryKey().defaultRandom(),
    content: text('content').notNull(),
    metadata: jsonb('metadata').notNull(),
    fts: tsvector('fts')
      .notNull()
      .generatedAlwaysAs(
        (): SQL ⇒ sql`to_tsvector('portuguese', ${fileChunk.content})`
      ),
    embedding: vector('embedding', { dimensions: 1536 }).notNull(),
    fileId: uuid('file_id')
      .references(() ⇒ file.id)
      .notNull(),
    createdAt: timestamp('created_at').defaultNow().notNull(),
    updatedAt: timestamp('updated_at').defaultNow().notNull(),
  },
  (table) ⇒ [
    index('idx_file_chunks_fts').using('gin', table.fts),
    index('idx_file_chunks_embedding').using(
      'hnsw',
      table.embedding.op('vector_ip_ops')
    ),
  ]
);
```

Figura 3: Código utilizado para criação da tabela `file_chunk`

No contexto das interações com os usuários, a Tabela `chat` (Figura 4) registra os metadados de cada sessão, incluindo o identificador único, o campo `user_id` que vincula a conversa a um usuário específico (via serviço Clerk) e o título (`title`), gerado automaticamente por inteligência artificial com base na primeira mensagem. Já a Tabela `message` (Figura 5) armazena cada mensagem individual de um chat, associando-a ao respectivo `chat_id` e contendo o campo `role`, que identifica o autor (`user` ou `assistant`), e o campo `parts`, do tipo JSONB, que permite registrar não apenas o conteúdo textual, mas também dados estruturados relacionados à geração da resposta.

Fonte: Elaborado pelo autor.

```
export const chat = pgTable('chat', {
  id: uuid('id').primaryKey().defaultRandom(),
  userId: text('user_id').notNull(),
  title: text('title').notNull(),
  createdAt: timestamp('created_at').defaultNow().notNull(),
  updatedAt: timestamp('updated_at').defaultNow().notNull(),
});
```

Figura 4: Código utilizado para criação da tabela `chat`

Fonte: Elaborado pelo autor.

```
export const message = pgTable('message', {
  id: uuid('id').primaryKey().defaultRandom(),
  chatId: uuid('chat_id')
    .references(() ⇒ chat.id)
    .notNull(),
  parts: jsonb('parts').notNull(),
  role: text('role').notNull(),
  createdAt: timestamp('created_at').defaultNow().notNull(),
  updatedAt: timestamp('updated_at').defaultNow().notNull(),
});
```

Figura 5: Código utilizado para criação da tabela `message`

Por fim, a Tabela `vote` (Figura 6) foi projetada para registrar o feedback dos usuários quanto à qualidade das respostas do assistente. Ela utiliza uma chave primária composta pelos campos `chat_id` e `message_id`, de forma a impedir votos duplicados para uma mesma mensagem dentro de uma conversa. O campo `is_upvoted`, do tipo booleano, indica se a avaliação foi positiva ou negativa, constituindo dados valiosos para o monitoramento da eficácia do chatbot e a identificação de melhorias necessárias.

Fonte: Elaborado pelo autor.

```
export const vote = pgTable(
  'vote',
  {
    chatId: uuid('chat_id')
      .notNull()
      .references(() ⇒ chat.id),
    messageId: uuid('message_id')
      .notNull()
      .references(() ⇒ message.id),
    isUpvoted: boolean('is_upvoted').notNull(),
    createdAt: timestamp('created_at').defaultNow().notNull(),
    updatedAt: timestamp('updated_at').defaultNow().notNull(),
  },
  (table) ⇒ [primaryKey({ columns: [table.chatId, table.messageId] })]
);
```

Figura 6: Código utilizado para criação da tabela `vote`

Por fim, o diagrama relacional elaborado no `dbdiagram.io`, apresentado na Figura 7, consolida visualmente toda a modelagem do banco de dados desenvolvida. Essa representação resume de maneira clara as relações entre as tabelas, evidenciando chaves primárias, estrangeiras e restrições de integridade. Além de sintetizar a estrutura lógica do sistema, o diagrama serve como referência estratégica para manutenção, documentação e evolução futura da aplicação, garantindo a preservação da coerência e da integridade da arquitetura de dados.

Fonte: Elaborado pelo autor.

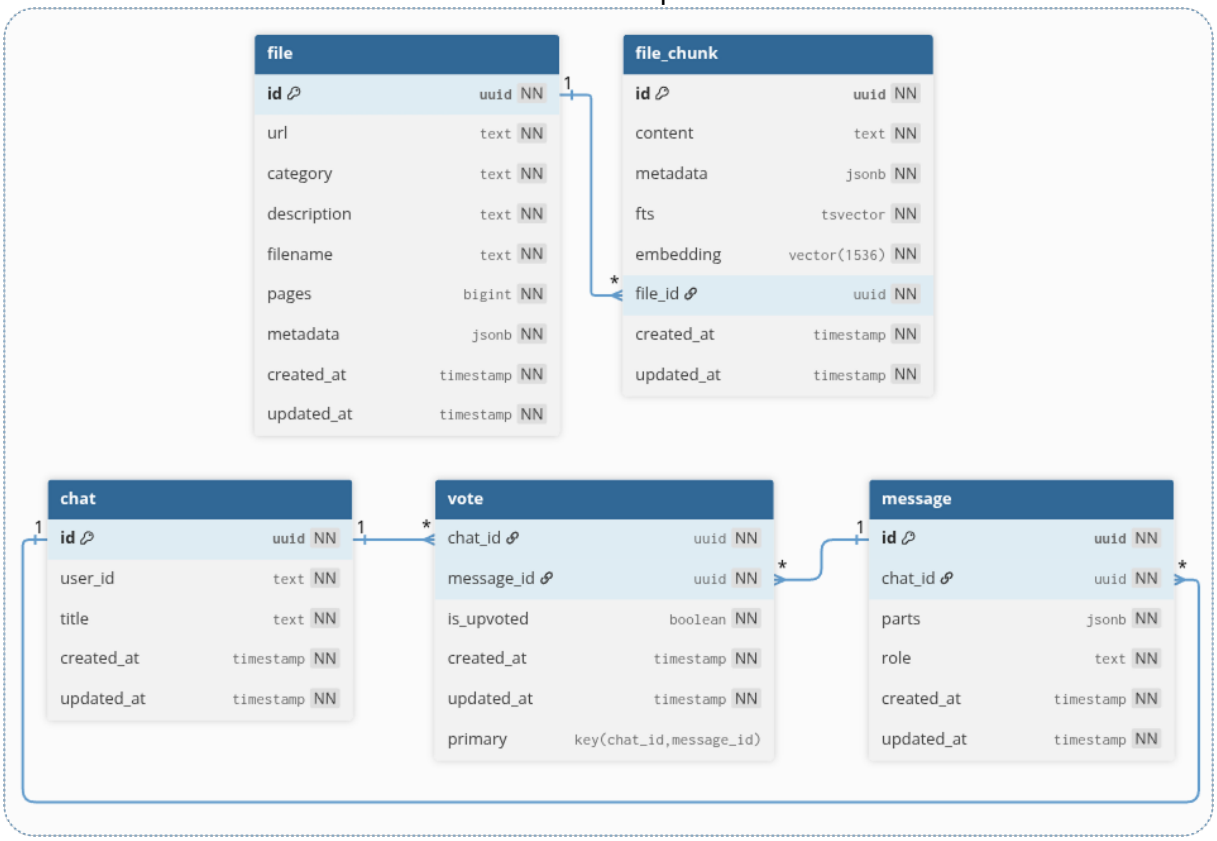

Figura 7: Diagrama relacional das tabelas do banco de dados gerado no `dbdiagram.io`

4 TRABALHOS CORRELATOS

4.1 Explorando chatbots baseados em LLM para criação automática de questões práticas de programação de computadores

O trabalho desenvolvido por Zimerman (2024), intitulado Explorando Chatbots Baseados em LLM para Criação Automática de Questões Práticas de Programação de Computadores, teve como objetivo investigar a utilização de modelos de linguagem de larga escala (LLMs), como o ChatGPT, para apoiar o processo de ensino de programação. A proposta concentrou-se na criação automática de questões didáticas, explorando ferramentas baseadas em processamento de linguagem natural e engenharia de prompt para gerar atividades contextualizadas. Apesar de apresentar contribuições relevantes para o campo educacional, o foco do estudo se restringe à dimensão pedagógica, voltada a professores e alunos de disciplinas iniciais de programação. Nesse sentido, o Harpia diferencia-se por se estruturar em torno de um problema institucional concreto: a recorrência de dúvidas e dificuldades de comunicação durante o processo de ingresso na universidade. Ao contrário de apenas automatizar a geração de conteúdo, o Harpia organiza e sistematiza informações essenciais para candidatos e servidores, atuando de forma estratégica na redução de demandas repetitivas no atendimento.

4.2 PLUTO: um sistema de chatbot que utiliza IA e a abordagem RAG para responder dúvidas sobre o SIGAA

Outro estudo relevante é o de Oliveira (2025), que desenvolveu o PLUTO: um sistema de chatbot que utiliza IA e a abordagem RAG para responder dúvidas sobre o SIGAA. O PLUTO foi concebido para atender estudantes da Universidade Federal de Alagoas no uso do Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA), uma ferramenta essencial no cotidiano acadêmico. O sistema integra técnicas de inteligência artificial com a abordagem Retrieval-Augmented Generation (RAG), garantindo precisão nas respostas a partir de uma base de conhecimento constantemente atualizada por meio de um painel administrativo. O trabalho demonstra a eficácia do modelo. Embora o PLUTO represente um avanço significativo ao automatizar dúvidas relacionadas a sistemas de gestão acadêmica, sua aplicação é restrita a um domínio específico. Nesse ponto, o Harpia apresenta um diferencial ao não se limitar a um sistema acadêmico em particular, mas sim ao abranger múltiplos setores e públicos dentro da universidade, oferecendo suporte tanto a ingressantes quanto a técnicos

administrativos.

4.3 Um Chatbot Especializado para o Contexto da Universidade Federal de Ouro Preto

Na mesma direção, Barbosa (2024) apresentou o trabalho Um Chatbot Especializado para o Contexto da Universidade Federal de Ouro Preto, cujo objetivo foi a construção de um assistente virtual integrado à API do ChatGPT para responder a dúvidas frequentes relacionadas à Pró-Reitoria de Graduação da instituição. O sistema utilizou técnicas de indexação de documentos oficiais para garantir padronização e confiabilidade das respostas. Assim como o Harpia, o chatbot proposto buscou atender demandas institucionais, reforçando o potencial da tecnologia no contexto acadêmico. Entretanto, a principal diferença reside no público-alvo: enquanto o chatbot da UFOP concentrou-se em atender exclusivamente estudantes, o Harpia foi concebido para atuar em duas frentes complementares, ao mesmo tempo em que orienta candidatos e ingressantes sobre etapas do processo seletivo, documentação e prazos, também funciona como ferramenta de apoio para técnicos administrativos, contribuindo para a redução da sobrecarga de atendimentos repetitivos.

4.4 Análise geral das comparações

Ao sintetizar a análise dos trabalhos correlatos, observa-se que cada um apresenta contribuições específicas no uso de chatbots no contexto educacional e institucional. Zimerman (2024) destaca o potencial de LLMs na geração de conteúdos didáticos; Oliveira (2025) enfatiza a integração de inteligência artificial com a abordagem RAG para suporte a sistemas acadêmicos; e Barbosa (2024) demonstra a aplicabilidade de chatbots especializados para atender estudantes a partir de documentos institucionais. O Harpia, contudo, diferencia-se por seu caráter abrangente e estratégico, uma vez que foi concebido para atender simultaneamente dois públicos distintos: os candidatos e ingressantes, que necessitam de informações claras sobre o processo seletivo, e os técnicos administrativos, que se beneficiam da automatização de atendimentos rotineiros. Dessa forma, a solução proposta se posiciona como uma ferramenta tecnológica de impacto direto na melhoria da comunicação institucional e na eficiência dos serviços prestados pela universidade. A seguir apresenta-se a Tabela 1 de comparações e os principais destaques do chatbot Harpia.

Fonte: Elaborado pelo autor.

Trabalho	**Tecnologias Utilizadas**	**Foco Principal**	**Diferencial do Harpia**
Zimerman (2024) – Chatbots LLM	Integração com LLM	Criação automática de questões de programação	Atua em problema institucional concreto, organizando informações críticas para ingresso de alunos.
Oliveira (2025) – PLUTO	Integração com LLM e RAG	Suporte ao sistema SIGAA (gestão acadêmica)	Abrange múltiplos setores e públicos, não limitado a um sistema acadêmico.
Barbosa (2023) – UFOP	Integração com LLM e RAG	Atendimento a dúvidas acadêmicas via documentos institucionais	Define públicos distintos (candidatos e técnicos), reduzindo sobrecarga administrativa.
Harpia – UFPA	Integração com LLM e RAG	Orientação a ingressantes e apoio administrativo	Atende simultaneamente estudantes e servidores, ampliando a eficiência comunicacional.

Tabela 1: Comparação entre trabalhos correlatos e o Harpia

5 PROCESSO DE DESENVOLVIMENTO

5.1 Levantamento de Requisitos

A seguir, na Tabela 2 são apresentados os requisitos funcionais (RFs) e não funcionais (RNFs) da aplicação, onde os RFs descrevem as funcionalidades essenciais do sistema, ou seja, o que o chatbot deve realizar para atender às necessidades dos usuários, como autenticação, envio de mensagens, consultas à base de conhecimento e administração do histórico de conversas. Já os RNFs definem as características de qualidade e restrições do sistema, incluindo desempenho, segurança, responsividade, tecnologias utilizadas e formas de armazenamento. A separação clara entre RFs e RNFs permite compreender tanto as funcionalidades que o sistema deve executar quanto os padrões e condições que devem ser observados para garantir sua eficiência, confiabilidade e usabilidade.

ID	Descrição	Novo
Requisitos Funcionais (RFs)		
RF01	O sistema deve permitir que o usuário envie mensagens de texto para interagir com o chatbot.	
RF02	O chatbot deve processar as mensagens recebidas e gerar respostas adequadas com base em seu modelo de processamento.	
RF03	O sistema deve permitir autenticação do usuário, garantindo acesso seguro e personalizado às funcionalidades do chatbot.	X
RF04	O chatbot deve oferecer suporte a conversas, sempre mantendo o contexto entre mensagens enviadas pelo usuário.	
RF05	O sistema deve permitir que o usuário visualize o histórico de suas conversas anteriores com o chatbot.	X
RF06	O sistema deve permitir que o usuário exclua conversas anteriores de forma permanente.	X
RF07	O chatbot deve fornecer mensagens de erro ou esclarecimento caso não entenda a solicitação do usuário.	
RF08	O chatbot deve permitir a coleta de feedback do usuário sobre a qualidade das respostas fornecidas.	X
RF09	O sistema deve permitir que administradores acessem um painel de métricas com estatísticas de utilização da aplicação.	X
RF10	O sistema deve permitir a geração automática de títulos para novos chats com base nas primeiras interações do usuário.	X

ID	Descrição	Novo
RF11	O sistema deve permitir o upload de documentos para a base de conhecimento, com posterior processamento para consultas futuras.	
RF12	O chatbot deve ser capaz de recuperar informações relevantes de uma base de conhecimento utilizando a abordagem de RAG para enriquecer as respostas.	X
RF13	O sistema deve realizar buscas híbridas, combinando *semantic search* e *keyword search*, para localizar informações mais precisas na base de conhecimento.	X
RF14	A API do chatbot deve implementar mecanismos de segurança, incluindo autenticação, autorização e limitação de requisições.	X
Requisitos Não Funcionais (RNFs)		
RNF01	O sistema deve utilizar Next.js, React, Tailwind CSS e Shadcn/UI para construção da interface do usuário.	
RNF02	A autenticação dos usuários deve ser implementada com o serviço Clerk, garantindo segurança no controle de acesso.	X
RNF03	Os documentos enviados devem ser processados utilizando LangChain para divisão em *chunks* e OpenAI Embeddings para geração de vetores semânticos.	X
RNF04	A base de conhecimento deve utilizar um banco de dados PostgreSQL para armazenar metadados e embeddings.	X
RNF05	O sistema deve armazenar os arquivos em um serviço de object storage compatível com S3.	X
RNF06	O sistema deve ser responsivo e adaptável a diferentes tamanhos de tela e dispositivos.	
RNF07	A aplicação deve oferecer suporte a modo claro e escuro, respeitando a preferência do usuário.	
RNF08	O sistema deve implementar logs e monitoramento para rastrear falhas, erros e métricas de desempenho.	X

Tabela 2: Requisitos Funcionais e Não Funcionais da Harpia com Indicação de Novos Requisitos

5.2 Diagrama de Casos de Uso

O diagrama de casos de uso é uma ferramenta da Engenharia de Software que modela as funcionalidades do sistema Harpia sob a perspectiva do usuário, representando os atores e suas principais interações. Para este projeto, foram definidos dois atores principais. O Candidato representa o público geral, que pode realizar as interações essenciais: enviar perguntas, gerenciar seu histórico de conversas e avaliar as respostas do assistente. O Administrador, por sua vez, é uma especialização do Candidato, representando os servidores técnicos que, além de todas as funções comuns, possuem acesso a funcionalidades exclusivas de gestão.

As interações são representadas pelos principais casos de uso, todos eles dependentes da ”Autenticação”. Funcionalidades exclusivas do Administrador são agrupadas no caso de uso ”Acessar Painel Administrativo”, que permite a visualização de métricas essenciais sobre a utilização do sistema. A Figura 8 apresenta o diagrama que ilustra visualmente essas relações, delimitando o escopo da Harpia.

Fonte: Elaborado pelo autor.

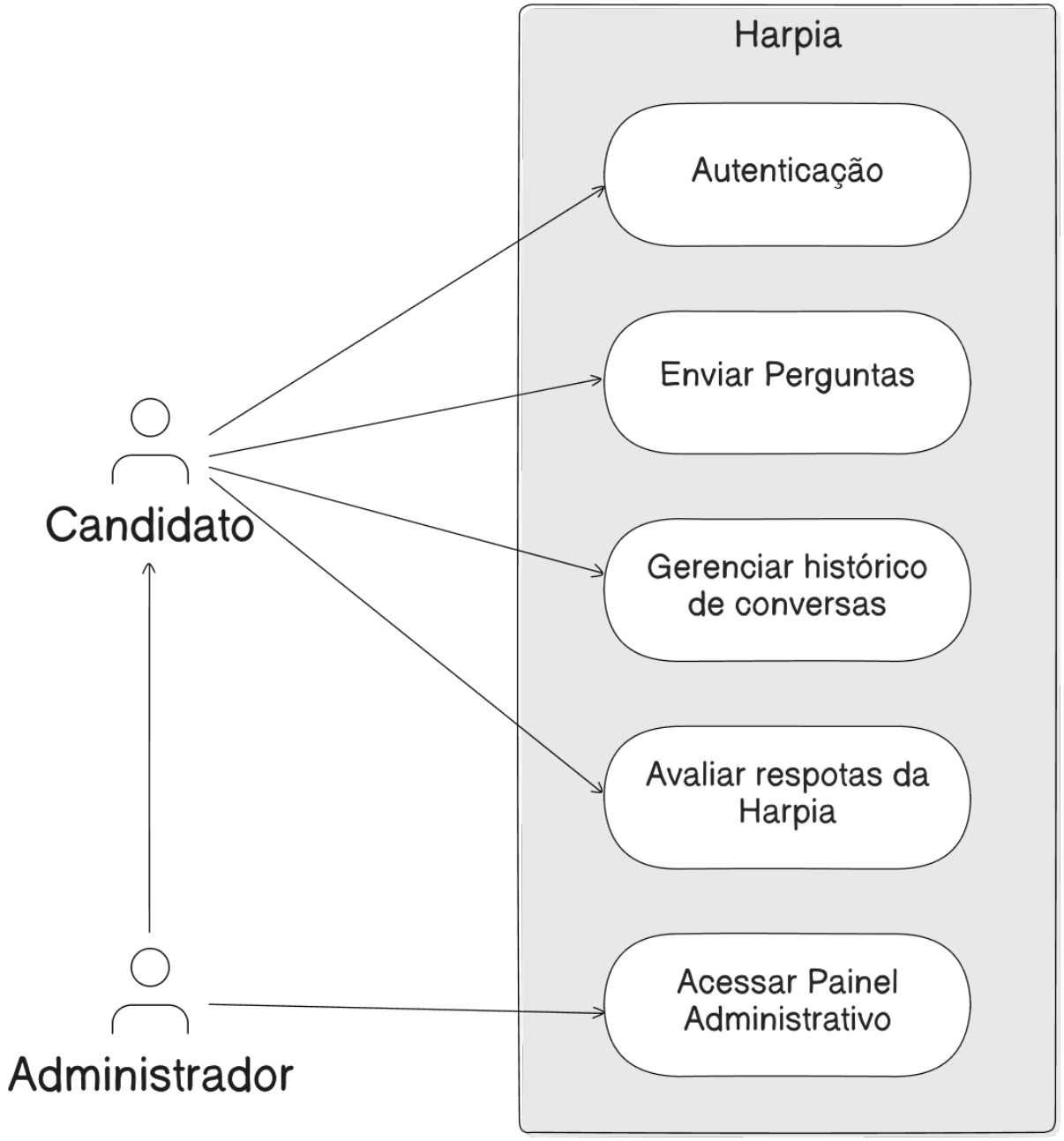

Figura 8: Diagrama de Casos de Uso da Harpia

5.3 Configurações do Ambiente

O ambiente de desenvolvimento da aplicação foi estruturado para garantir portabilidade, consistência e facilidade de manutenção, integrando ferramentas modernas de versionamento, containerização e deploy. O banco de dados *PostgreSQL* utilizado

no desenvolvimento é provisionado via *Docker*, garantindo consistência entre diferentes ambientes e eliminando problemas de configuração local, além de permitir a fácil criação, reinicialização e remoção de containers para testes e desenvolvimento.

As credenciais e configurações sensíveis, como chaves de API da *OpenAI* e *Google Gemini*, credenciais do *S3* e tokens do *Clerk*, são armazenadas em variáveis de ambiente. Essa abordagem evita a exposição de informações sensíveis no código-fonte e facilita a configuração de diferentes ambientes, incluindo desenvolvimento, *staging* e produção.

Todo o código-fonte do projeto é versionado com *Git*, hospedado no *GitHub*. A estratégia de *branches* segue boas práticas de desenvolvimento, permitindo colaboração eficiente, revisão de código e histórico completo de alterações. O *front-end* da aplicação, desenvolvido em *Next.js*, é automaticamente publicado no ambiente de produção através do *Vercel*, integrando-se com o repositório *GitHub*. Isso possibilita atualizações rápidas e automáticas, além de *rollback* seguro em caso de falhas.

Para aumentar a produtividade do desenvolvimento e a organização do código, foram utilizadas ferramentas como o *Cursor*, que auxilia na escrita e navegação de código dentro do editor, e o *Ultracite*, um formatador e *linter* configurável, otimizado para integração com inteligência artificial, que garante consistência e boas práticas no estilo de código da equipe.

O ambiente de desenvolvimento e produção integra de forma segura o armazenamento *S3*, o banco *PostgreSQL*, os modelos de IA da *OpenAI* e *Google Gemini* e o serviço de autenticação *Clerk*. A configuração modular permite alterar provedores ou credenciais sem impactar a lógica principal da aplicação, garantindo flexibilidade e segurança na gestão dos serviços externos.

5.4 Desenvolvimento do *Front-end*

O desenvolvimento do *front-end* da Harpia foi realizado utilizando o *Next.js* com o *App Router*, o que possibilitou a criação de uma estrutura organizada, com páginas agrupadas de acordo com suas funcionalidades. Dentro do diretório `src/app`, foram definidos diferentes grupos de rotas que compõem a interface principal da aplicação.

O primeiro grupo, denominado `(auth)`, reúne as páginas responsáveis pelos fluxos de autenticação e cadastro de usuários, integradas ao serviço *Clerk*. Essas páginas incluem o acesso à tela de entrada (*sign-in*) e à tela de registro (*sign-up*), essenciais para controlar a sessão de cada usuário antes de permitir o acesso às demais áreas da plataforma.

O segundo grupo, `(chat)`, concentra a página raiz do sistema, que representa a

interface principal de interação com o chatbot. Além disso, há uma página dinâmica identificada por `c/[id]`, responsável por renderizar conversas já existentes, permitindo ao usuário retomar interações anteriores de forma simples e intuitiva.

Outro ponto central do *front-end* é o conjunto de páginas localizadas na rota `/admin`, destinadas exclusivamente aos usuários com permissões administrativas. Nesse espaço, a página inicial do painel administrativo `/admin` exibe métricas e informações gerais sobre o uso da aplicação. Além dela, a página `/admin/files` permite visualizar os documentos presentes na base de conhecimento da Harpia, enquanto a página `/admin/messages` possibilita analisar todas as mensagens trocadas entre usuários e o modelo, incluindo perguntas, respostas e os respectivos feedbacks enviados.

Essa organização de rotas, aliada ao uso de layouts persistentes, garante uma navegação consistente, além de facilitar a manutenção e a escalabilidade do projeto. A Figura 9 ilustra a estrutura geral de páginas implementadas no front-end da aplicação.

Fonte: Elaborado pelo autor.

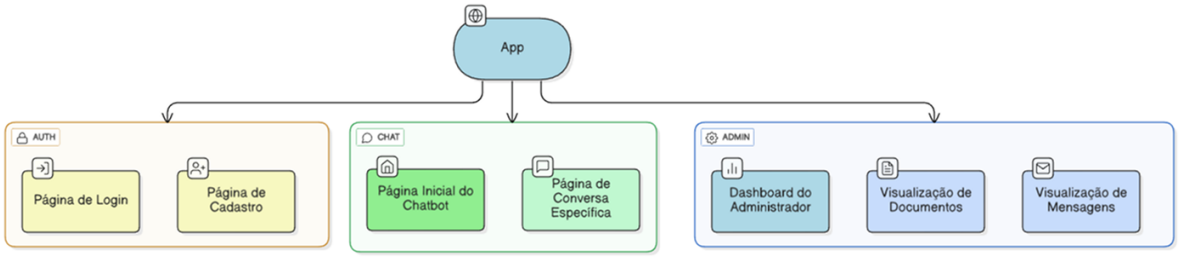

Figura 9: Páginas da Harpia com *App Router*

5.5 Desenvolvimento do *Back-End*

O desenvolvimento do *back-end* da aplicação Harpia foi estruturado utilizando o *App Router* do *Next.js*, concentrando toda a lógica da API na pasta `src/app/api`. As rotas foram planejadas para atender às principais funcionalidades do sistema, garantindo comunicação eficiente entre o *front-end*, o banco de dados e os modelos de inteligência artificial.

O núcleo da aplicação é a rota `/chat`, responsável pelo processamento das mensagens enviadas pelos usuários e pela geração de respostas com auxílio de um modelo de linguagem integrado a uma ferramenta de busca híbrida (*RAG*). O fluxo dessa rota envolve diversas etapas: autenticação do usuário via *Clerk*, validação do limite diário de mensagens, carregamento de mensagens anteriores, processamento pelo modelo generativo e armazenamento das mensagens no banco de dados. A lógica implementada permite criar novos chats dinamicamente, com títulos gerados automaticamente

a partir da primeira mensagem do usuário, enquanto conversas existentes podem ser consultadas e atualizadas.

Um elemento central do processamento é a ferramenta de busca híbrida (*hybrid search*), que combina consultas semânticas e por palavras-chave utilizando embeddings do modelo *text-embedding-3-small* da OpenAI. Essa abordagem permite recuperar informações relevantes da base de conhecimento de forma precisa e eficiente, posicionando o *back-end* como camada intermediária entre o *front-end*, os serviços de IA e o banco de dados, assegurando segurança, integridade dos dados e alta performance.

Rotas complementares foram implementadas para melhorar a experiência do usuário e a gestão do sistema. A rota `/chat/[id]` possibilita operações específicas sobre conversas existentes, incluindo busca por mensagens individuais e exclusão de chats. A rota `/history` permite consultar o histórico completo de interações de um usuário, enquanto a rota `/vote` registra e consulta feedbacks referentes às respostas do modelo, fornecendo dados importantes para monitoramento e aprimoramento contínuo.

A arquitetura do *back-end* foi organizada de forma modular, facilitando manutenção, escalabilidade e integração de novas funcionalidades. Todas as operações relacionadas a dados estão centralizadas em funções reutilizáveis presentes em `queries.ts`, enquanto a lógica de comunicação com os modelos de IA e o processamento de fluxos conversacionais concentra-se na rota `/chat`, garantindo clareza e separação de responsabilidades dentro do código-fonte.

A Figura 10 ilustra a organização das rotas da API e o fluxo de execução das mensagens, mostrando como as solicitações dos usuários percorrem o sistema até serem processadas pelos modelos de IA e registradas no banco de dados, oferecendo uma visão clara do papel de cada rota e da integração entre os componentes do *back-end*.

Fonte: Elaborado pelo autor.

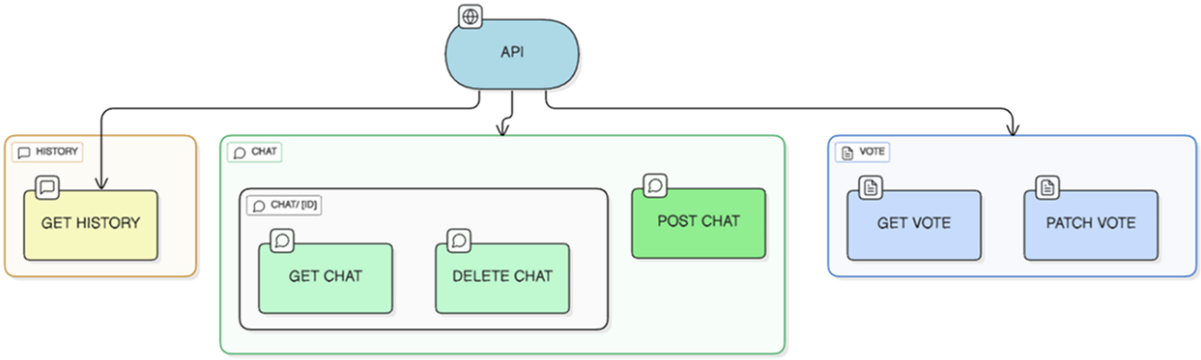

Figura 10: Estrutura das rotas da API com *App Router*

5.6 Inserção de Documentos

O processo de inserção de documentos foi implementado como um script independente (`src/scripts/upload.ts`), responsável por ler arquivos PDF, extrair seu conteúdo, gerar metadados e armazenar todas as informações na base de dados e no serviço de armazenamento S3. O script garante que cada documento carregado seja processado de maneira estruturada, permitindo consultas eficientes e integração com o mecanismo de busca híbrida (*RAG*) utilizado pelo chatbot.

Inicialmente, o script realiza a autenticação e configuração do cliente S3, garantindo a compatibilidade com diferentes provedores de armazenamento. Em seguida, verifica se o arquivo já existe no bucket para evitar duplicidades, realizando o upload apenas quando necessário. Após a confirmação do arquivo no S3, o PDF é carregado e processado por meio do *PDFLoader*, extraindo cada página como documento individual.

Os metadados principais do arquivo, incluindo número de páginas, título, categoria e descrição, são gerados com auxílio de um modelo de linguagem (*GPT-5 Nano*), possibilitando a classificação automática dos documentos e resumindo seu conteúdo para facilitar buscas posteriores. O script, então, divide o texto em fragmentos menores (*chunks*) utilizando o *RecursiveCharacterTextSplitter*, calculando posições de caracteres e linhas para manter a rastreabilidade de cada segmento.

Cada *chunk* é convertido em embedding semântico com o modelo *text-embedding-3-small* da OpenAI, permitindo consultas semânticas rápidas e precisas dentro da base de conhecimento. Por fim, todos os *chunks* e seus embeddings, juntamente com os metadados do arquivo principal, são inseridos no banco de dados, garantindo que o sistema possa fornecer respostas contextualmente relevantes durante as interações com o usuário.

A Figura 11 apresenta o fluxo geral do script de inserção, desde a leitura do arquivo PDF até o armazenamento dos dados no banco e no S3, evidenciando a separação das etapas e a integração com os modelos de IA e serviços externos.

Fonte: Elaborado pelo autor.

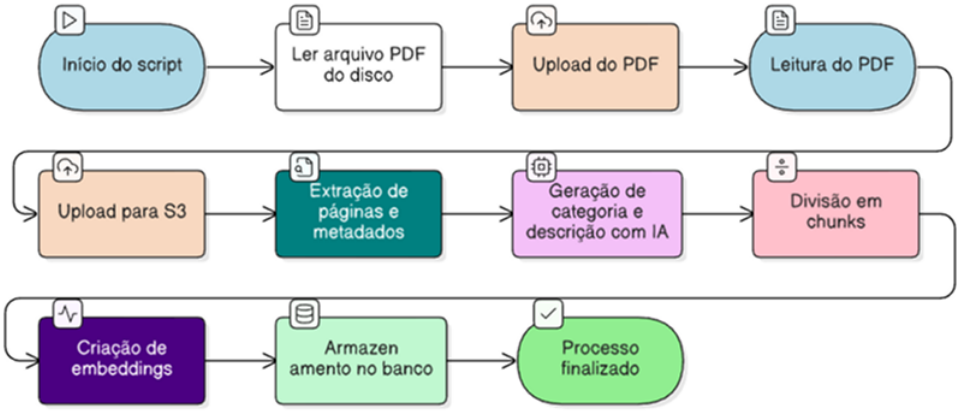

Figura 11: Fluxo de processamento e inserção de documentos no sistema

6 RESULTADOS

A versão inicial da Harpia forneceu uma base sólida para o desenvolvimento do chatbot, permitindo identificar pontos fortes e limitações relevantes. Entre os aspectos positivos, destacaram-se a capacidade de fornecer respostas rápidas e contextualizadas, além da interface intuitiva, que facilitava a interação dos usuários. Por outro lado, a análise dos feedbacks revelou desafios, como dificuldades na localização de informações nos editais, uso excessivo de linguagem técnica e respostas parcialmente precisas em consultas mais complexas. Esses achados serviram de referência para orientar aprimoramentos, incluindo ajustes no modelo de linguagem integrado à ferramenta de busca híbrida baseada em RAG e melhorias na experiência do usuário, visando interações mais claras, precisas e acessíveis.

A avaliação da primeira versão foi conduzida com base em 22 respostas, sendo 10 de discentes e 12 de técnicos administrativos e bolsistas envolvidos nas atividades do Processo Seletivo (PS), com o trabalho publicado no evento CSBC WASHES 2025 como ”Desenvolvimento de um Chatbot Inteligente para auxílio de atividades organizacionais durante o período de Processos Seletivos da UFPA”. Os resultados evidenciaram percepções variadas quanto à utilidade, precisão, clareza das respostas e facilidade de uso, permitindo identificar os pontos prioritários de melhoria para a versão atual.

Em termos de precisão, a primeira versão do Harpia apresentou desempenho satisfatório, mas com limitações: 5 respostas (45,4%) atribuíram nota 4 e 3 respostas (27,2%) avaliaram o desempenho como excelente, enquanto consultas mais complexas receberam avaliações mais baixas, indicando respostas parcialmente corretas (PANTOJA et al., 2025). Na versão atual, a integração com a ferramenta de busca híbrida baseada em RAG permite consultar múltiplos chunks das fontes da base de conhecimento, apresentando informações contextualizadas e referências a páginas ou trechos utilizados, com buscas semânticas e por palavras-chave. Melhorias na interface e no fluxo de interação garantem respostas claras, objetivas e rapidamente verificáveis. A Figura 12 exemplifica uma resposta do Harpia, destacando as fontes consultadas pelo LLM.

Fonte: Elaborado pelo autor.

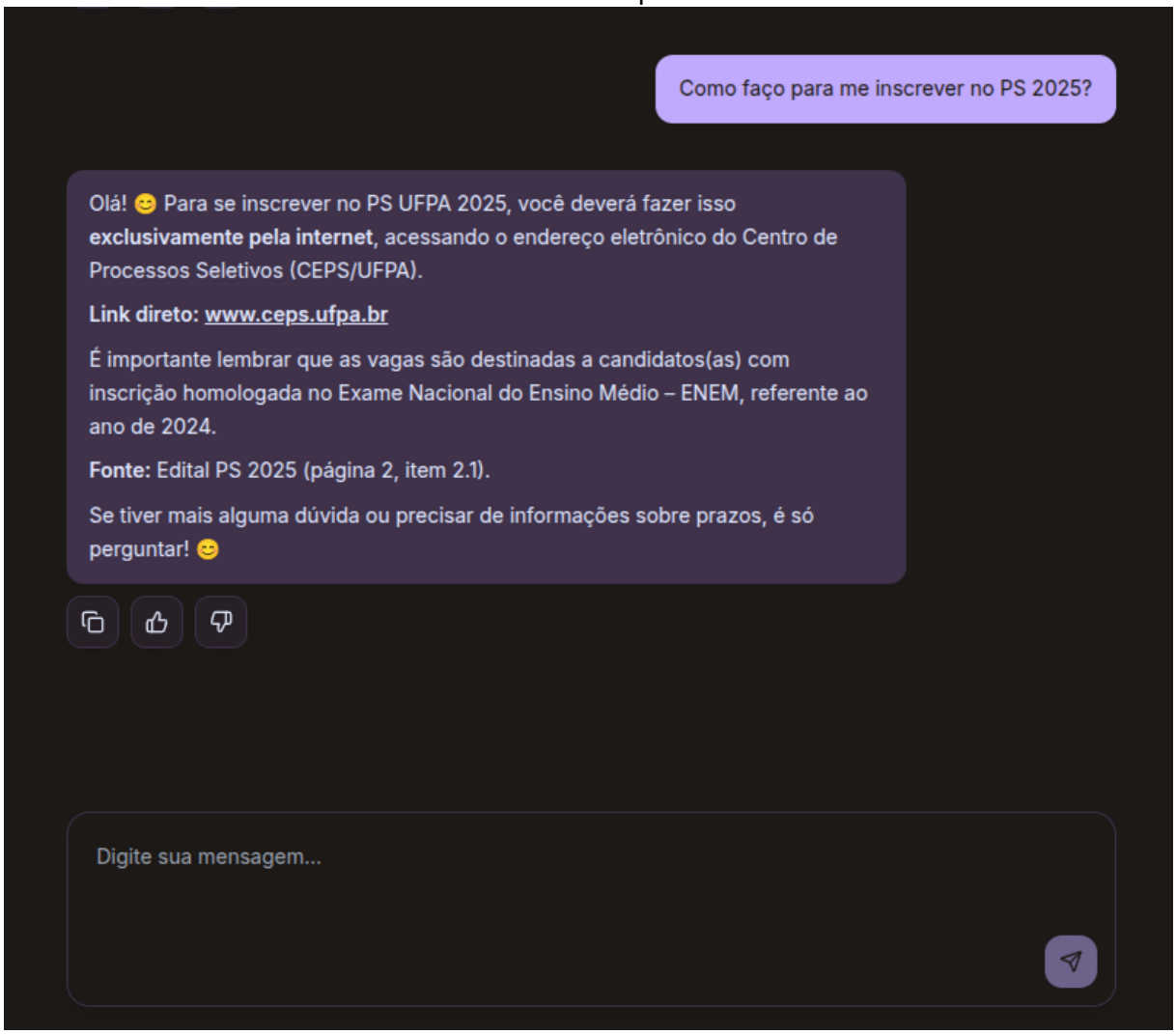

Figura 12: Exemplo de mensagem do Harpia com referência às fontes consultadas.

Quanto à utilidade para a rotina de trabalho, a primeira versão recebeu, na maior parte, nota 3 em uma escala de 1 a 5 (6 respostas, 54,5%), indicando que o sistema facilitava as atividades de forma moderada (PANTOJA et al., 2025). Entre os desafios apontados estavam a localização de informações específicas nos editais, respostas parcialmente precisas e a necessidade de consultar múltiplas fontes. As melhorias implementadas ampliaram a utilidade do Harpia por meio do painel administrativo, que oferece visualização detalhada das interações dos usuários, permitindo identificar tópicos frequentes, padrões de dúvidas recorrentes e respostas avaliadas como satisfatórias ou insatisfatórias. A Figura 13 apresenta a tela do painel administrativo com o histórico de mensagens, exibindo cada interação, o ID do usuário, conteúdo, feedback e data, auxiliando na análise e otimização dos processos de atendimento.

Fonte: Elaborado pelo autor.

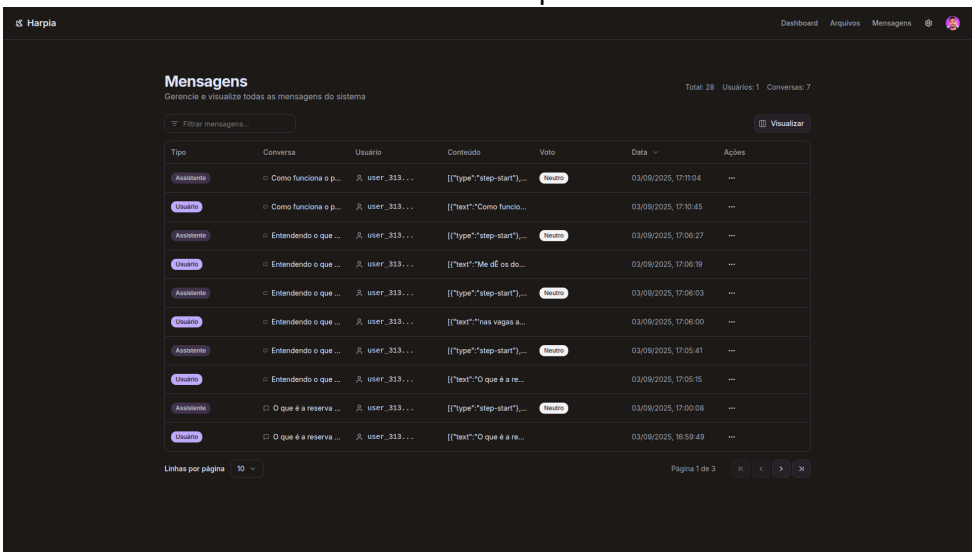

Figura 13: Tela de mensagens do Painel Administrativo.

Em relação ao impacto da Harpia no atendimento ao público durante o PS 2025, a avaliação inicial indicou resultados positivos: 5 respostas (45,4%) atribuíram nota 4, indicando melhorias consideráveis, e 4 respostas (27,2%) atribuíram nota 5, sugerindo impacto transformador (PANTOJA et al., 2025). Para reforçar o suporte e garantir a flexibilidade da ferramenta, a Harpia utiliza um script de inserção de documentos, que processa e indexa editais e anexos escolhidos pelos administradores, mantendo o sistema atualizado e permitindo respostas rápidas e precisas. A Figura 14 mostra a saída do script após a inserção de um documento na base de conhecimento do Chatbot.

Fonte: Elaborado pelo autor.

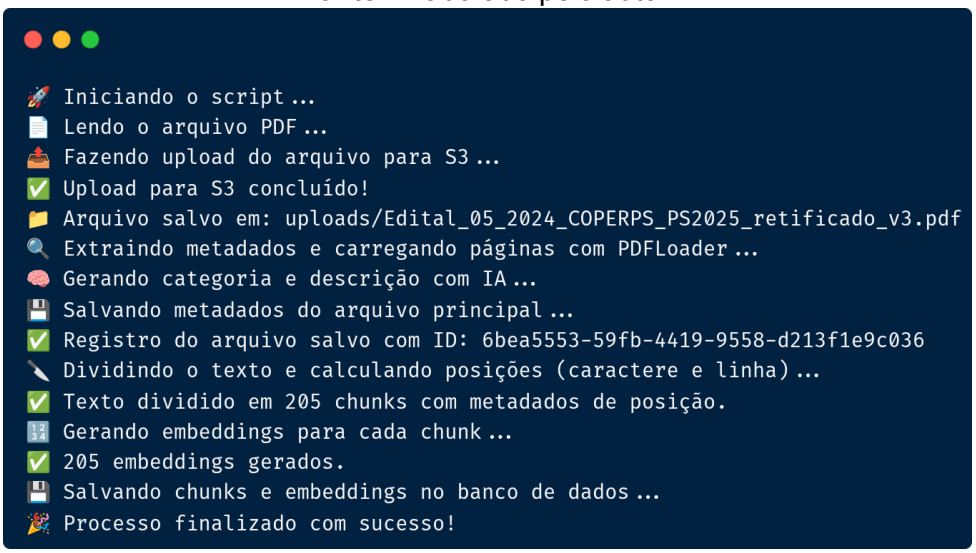

Figura 14: Saída do script de inserção de documentos da Harpia.

Na primeira versão, a clareza das respostas foi avaliada como moderada ou satisfatória em 72,6% dos casos, e 27,2% consideraram as respostas muito claras (PANTOJA et al., 2025). Com melhorias, como novas instruções do sistema, o Harpia passou a fornecer respostas mais objetivas e contextualizadas, com referências a fontes e trechos consultados, facilitando a compreensão, reduzindo ambiguidades e promovendo uma experiência mais confiável.

Quanto à facilidade de uso, 81,7% dos usuários avaliaram a usabilidade como satisfatória ou muito satisfatória, enquanto 18,1% consideraram a facilidade de uso moderada, indicando necessidade de ajustes na interface. Alterações na experiência do usuário incluem acesso ao histórico de conversas, remoção de chats antigos, possibilidade de votar em respostas para feedback e maior customização, garantindo praticidade para diferentes casos de uso. A Figura 15 apresenta um print da tela com o histórico de conversas de cada usuário, incluindo conteúdo das mensagens, ID do usuário, votos e data, facilitando a análise e otimização dos atendimentos.

Fonte: Elaborado pelo autor.

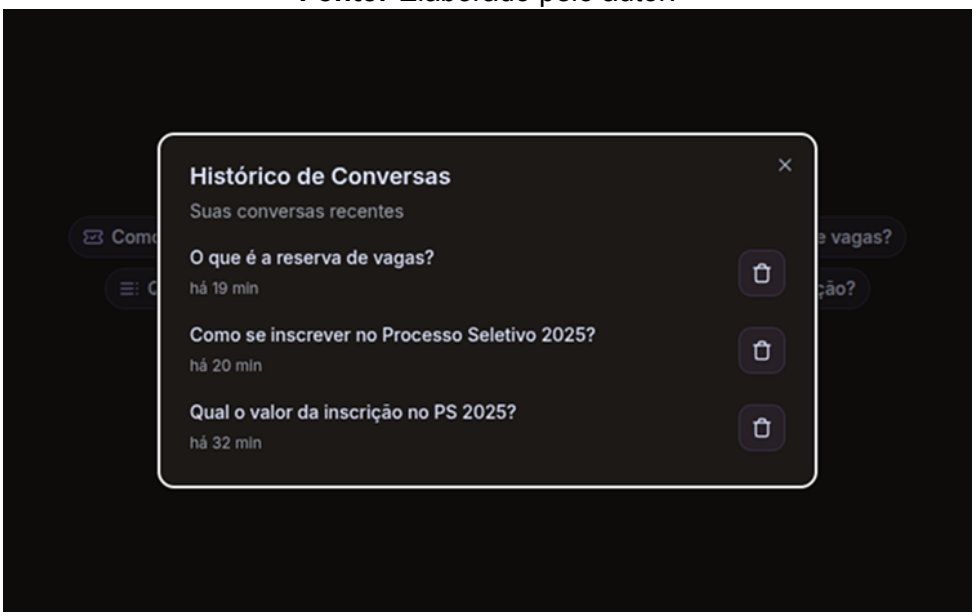

Figura 15: Captura do histórico de conversas do usuário.

Dessa forma, a análise detalhada dos resultados evidencia uma evolução substancial do chatbot Harpia em múltiplas frentes. As melhorias implementadas, orientadas pelo feedback da versão inicial e pela adoção de uma arquitetura mais robusta, não apenas corrigiram as limitações apontadas, mas também introduziram novas capacidades estratégicas ao sistema, especialmente no que tange à confiabilidade das informações e ao suporte administrativo. Para consolidar e visualizar de forma clara essa progressão, a Tabela 3 resume os principais avanços, comparando as características

da primeira versão com as da versão atual aprimorada.

Critério de Análise	Versão Inicial	Versão Atual
Precisão e Confiabilidade	Respostas satisfatórias, mas com limitações em consultas complexas e sem referência direta às fontes.	Respostas geradas via busca híbrida (RAG), com consulta a múltiplos trechos dos documentos e exibição das fontes utilizadas, aumentando a confiabilidade.
Utilidade para Gestão	Considerada moderada. Não possuía ferramentas para análise do uso ou gestão do sistema.	Possui Painel Administrativo, permitindo a visualização de métricas, análise do histórico de mensagens e monitoramento de feedbacks dos usuários.
Gerenciamento de Conteúdo	Base de conhecimento estática, sem um método simples para atualização de documentos.	Implementação de um script de inserção que permite aos administradores processar e indexar novos editais e anexos, mantendo a base de conhecimento sempre atualizada.
Clareza das Respostas	Respostas consideradas claras, mas com eventual uso de linguagem técnica e risco de ambiguidades.	Respostas mais objetivas e contextualizadas, com referências diretas a trechos e páginas dos documentos, facilitando a verificação e a compreensão.
Experiência do Usuário	Interface intuitiva, mas com funcionalidades básicas. Sem gestão avançada de conversas.	Interface aprimorada com novas funcionalidades: acesso ao histórico de conversas, opção para excluir chats, sistema de votação para feedback e maior customização.

Tabela 3: Tabela comparativa entre a versão inicial e a versão atual do chatbot Harpia.

7 CONSIDERAÇÕES FINAIS

7.1 Conclusão

O presente trabalho teve como propósito desenvolver e aprimorar a Harpia, um chatbot inteligente projetado para modernizar e otimizar a comunicação nos processos seletivos da Universidade Federal do Pará (UFPA). A crescente demanda por eficiência, transparência e acessibilidade em instituições de ensino superior públicas motivou a criação de uma solução tecnológica que pudesse mitigar a sobrecarga dos servidores técnico-administrativos e, simultaneamente, aprimorar a experiência dos candidatos ao centralizar e simplificar o acesso a informações cruciais.

Os objetivos propostos foram alcançados com sucesso. O objetivo geral de desenvolver um chatbot de código aberto, customizável e baseado na integração com um LLM e RAG foi plenamente atingido. Além disso, a arquitetura modular e escalável, fundamentada em boas práticas de engenharia de software e tecnologias modernas como Next.js, TypeScript e PostgreSQL com pgvector, garantiu a robustez e a manutenibilidade da aplicação. A implementação da técnica de Geração Aumentada por Recuperação (RAG), combinada com uma busca híbrida, permitiu que o Harpia fornecesse respostas precisas e contextualizadas, ancoradas em documentos oficiais, superando limitações de alucinação e conhecimento desatualizado inerentes aos modelos de linguagem de grande escala.

A comparação com a versão anterior e com trabalhos correlatos evidenciou o caráter inovador e estratégico da solução. Diferentemente de outras iniciativas, o Harpia foi concebido para atender a dois públicos distintos — candidatos e servidores —, posicionando-se não apenas como um canal de informações, mas como uma ferramenta de gestão que contribui diretamente para a eficiência operacional. Os resultados qualitativos, aprimorados a partir da análise da primeira versão, indicam um avanço significativo na precisão, clareza e utilidade do chatbot, especialmente com a adição do painel administrativo, que oferece insights valiosos para a melhoria contínua dos processos de atendimento.

Dessa forma, o Harpia se consolida como uma contribuição relevante para a UFPA, demonstrando o potencial da inteligência artificial para solucionar desafios práticos no ambiente acadêmico e promover um acesso mais democrático e equitativo à informação.

7.2 Dificuldades Encontradas

Durante o ciclo de desenvolvimento da Harpia, diversos desafios técnicos e conceituais foram enfrentados, os quais exigiram soluções criativas e aprofundamento em áreas específicas da engenharia de software e da inteligência artificial.

Uma das principais dificuldades reside na gestão do ciclo de vida dos dados na base de conhecimento. A abordagem RAG depende de uma base documental atualizada, mas os editais e regulamentos são documentos dinâmicos, sujeitos a retificações. O desafio central emerge ao se tentar atualizar os dados de forma eficiente. Quando um documento é alterado, seus fragmentos derivados (*chunks*) e os respectivos *embeddings* vetoriais tornam-se obsoletos. A dificuldade ainda consiste em desenvolver uma estratégia para identificar e atualizar apenas os *chunks* modificados, sem a necessidade de reprocessar o documento inteiro. A implementação de um mecanismo de versionamento e diferenciação de conteúdo em arquivos PDF mostra-se uma tarefa complexa e permanece como um problema em aberto, demandando uma solução robusta para garantir a consistência dos dados sem comprometer o desempenho.

Outro desafio significativo foi a operacionalização da inserção de documentos na base de conhecimento. Atualmente, todo o fluxo de processamento — desde o upload de um edital em PDF até sua fragmentação (chunking) e a geração dos embeddings vetoriais — é executado por meio de um script manual. Essa abordagem, embora funcional, cria uma dependência técnica e impede que administradores do sistema, como os servidores da universidade, possam gerenciar o ciclo de vida dos documentos de forma autônoma através de um painel administrativo. A ausência de uma interface visual para o upload e a gestão de novas versões dos editais torna o processo menos ágil e escalável, representando um obstáculo para a manutenção de uma base de conhecimento dinâmica e constantemente atualizada.

Por fim, a gestão da infraestrutura e a otimização de custos representaram um desafio constante. A arquitetura da aplicação integra múltiplos serviços de terceiros, como modelos de IA da OpenAI e Google, armazenamento de objetos S3 e serviços de autenticação. Orquestrar esses componentes de forma segura e, ao mesmo tempo, monitorar e controlar os custos operacionais foi crucial. A implementação de ferramentas como o Cloudflare AI Gateway e a escolha de modelos de IA com boa relação custo-benefício foram medidas essenciais para garantir a sustentabilidade financeira do projeto no contexto de uma universidade pública.

7.3 Trabalhos Futuros

1. **Implementação de um Sistema de Sincronização Automática:** Para solucionar a dificuldade de atualização da base de conhecimento, propõe-se o desenvolvimento de um módulo de sincronização que monitore repositórios institucionais em busca de alterações nos documentos. Esse sistema poderia realizar uma análise diferencial (*diff*) entre as versões dos arquivos, identificando, extraindo e reprocessando apenas os trechos modificados, otimizando a atualização dos *chunks* e *embeddings* de forma automatizada.

2. **Expansão para Funcionalidades Multimodais:** A próxima versão do Harpia poderia incorporar capacidades multimodais, permitindo que os usuários enviem imagens de documentos para análise ou interajam por meio de comandos de voz. Tais funcionalidades ampliariam a acessibilidade e a gama de casos de uso da ferramenta.

3. **Integração com Sistemas Externos:** Expandir a integração do Harpia com plataformas externas, como WhatsApp, redes sociais, sites institucionais e landing pages, permitiria ampliar o alcance e a conveniência do atendimento. O chatbot poderia responder dúvidas em múltiplos canais, notificar usuários sobre prazos e eventos relevantes, e fornecer informações consistentes de forma centralizada, evoluindo de uma ferramenta interna para um assistente omnicanal.

4. **Avaliação Quantitativa em Larga Escala:** Conforme mencionado na metodologia, a realização de um estudo quantitativo durante um ciclo completo de processo seletivo permitiria coletar métricas robustas sobre o impacto do chatbot, como a redução no volume de e-mails e chamadas aos setores administrativos, o tempo médio de atendimento e a satisfação geral dos usuários, validando empiricamente a eficácia da solução.

Referências

ADAMOPOULOU, E.; MOUSSIADES, L. Chatbots: History, technology, and applications. *Machine Learning with Applications*, v. 2, p. 100006, 2020. ISSN 2666-8270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666827020300062>.

ADAMOPOULOU, E.; MOUSSIADES, L. An overview of chatbot technology. In: . [S.l.: s.n.], 2020. p. 373–383. ISBN 978-3-030-49185-7.

ALVES, K.; VITORIO, A.; LUNA, J. D.; SOUZA, R. de. Inteligência artificial – aplicações e tendências. *Brazilian Journal of Development*, v. 9, p. 12560–12570, 04 2023.

AUTOKENT. *pdf-parse*. 2018. Acesso em: 04 set. 2025. Disponível em: <https://www.npmjs.com/package/pdf-parse>.

BARBOSA, L. S. Um chatbot especializado para o contexto da universidade federal de ouro preto. 2024.

BOGNER, J.; MERKEL, M. To type or not to type? a systematic comparison of the software quality of javascript and typescript applications on github. In: *Proceedings of the 19th International Conference on Mining Software Repositories*. [S.l.: s.n.], 2022. p. 658–669.

BRITO, F.; OTTONI, A.; OTTONI, L. Aplicações de aprendizado por reforço em manipuladores robóticos: Uma revisão sistemática. In: . [S.l.: s.n.], 2023. p. 1–8.

CHEN, K. *Improving Front-end Performance through Modular Rendering and Adaptive Hydration (MRAH) in React Applications*. 2025. Disponível em: <https://arxiv.org/abs/2504.03884>.

CHINIMILLI, V. R. P.; SADASIVUNI, L. The rise of ai: a comprehensive research review. *IAES International Journal of Artificial Intelligence (IJ-AI)*, v. 13, p. 2226, 06 2024.

CLOUDFLARE. *Cloudflare AI Gateway*. 2025. Disponível em: <https://developers.cloudflare.com/ai-gateway>.

DEEPMIND, G. *Modelos Gemini — Documentação da API*. 2025. <https://ai.google.dev/gemini-api/docs/models?hl=pt-br>. Acessado em: 02 set. 2025.

DIAS, V. L. Desenvolvimento de um sistema para submissão de candidaturas ao processo seletivo do programa de pós-graduação em computação da ufu. Universidade Federal de Uberlândia, 2023.

FALQUETO, J. M. Z.; FARIAS, J. S. A trajetória e a funcionalidade da universidade pública brasileira. Universidade Federal de Santa Catarina, 2013.

GAO, Y.; XIONG, Y.; GAO, X.; JIA, K.; PAN, J.; BI, Y.; DAI, Y.; SUN, J.; WANG, M.; WANG, H. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. Disponível em: <https://arxiv.org/abs/2312.10997>.

GROUP, P. G. D. *pgvector Documentation*. 2025. Disponível em: <https://github.com/pgvector/pgvector>.

GROUP, P. G. D. *PostgreSQL Documentation*. 2025. Disponível em: <https://www.postgresql.org>.

GROUP, P. G. D. *PostgreSQL Text Search (TSVector)*. 2025. Disponível em: <https://www.postgresql.org/docs/current/textsearch.html>.

GUPTA, S.; RANJAN, R.; SINGH, S. N. *A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions*. 2024. Disponível em: <https://arxiv.org/abs/2410.12837>.

HECK, F. R.; DREBES, A. A.; GUERRA, D. C.; SILVEIRA, F. E. d. Ampliando os serviços de entrega de documentação on-line para ingresso em concursos de graduação na universidade federal do rio grande do sul/ufrgs. In: *Workshop de Tecnologia da Informação e Comunicação das IFES (12: 2018: Foz do Iguaçu).[Anais..][recurso eletrônico]. Foz do Iguaçu*. [S.l.: s.n.], 2018.

INC., C. *Clerk Documentation*. 2025. Disponível em: <https://clerk.com>.

LABS, T. *Tailwind CSS Documentation*. 2025. Disponível em: <https://tailwindcss.com>.

LAN, Y.; LI, X.; DU, H.; LU, X.; GAO, M.; QIAN, W.; ZHOU, A. Survey of natural language processing for education: Taxonomy, systematic review, and future trends. *arXiv preprint arXiv:2401.07518*, 2024.

LANGCHAIN. *LangChain*. 2022. Acesso em: 04 set. 2025. Disponível em: <https://python.langchain.com/docs/introduction/>.

LANGCHAIN. *RecursiveCharacterTextSplitter*. 2022. Acesso em: 04 set. 2025. Disponível em: <https://python.langchain.com/api_reference/text_splitters/character/langchain_text_splitters.character.RecursiveCharacterTextSplitter.html>.

LAVIGNE, F. C. *Comportamento informacional humano em instituições universitárias: uma análise da ansiedade informacional do profissional de Secretariado Executivo da Ufba*. [S.l.]: Editora CRV, 2024.

LIMA, P. R. S.; PRESSER, N. H.; LOZANO, A. R. P. Transparência e qualidade informacional: um estudo dos portais das universidades federais brasileiras. *Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação*, v. 30, p. 1–29, maio 2025. Disponível em: <https://periodicos.ufsc.br/index.php/eb/article/view/102691>.

MICROSOFT. *TypeScript: JavaScript With Syntax For Types*. 2025. <https://www.typescriptlang.org/>. Acessado em: 02 set. 2025.

MINAEE, S.; MIKOLOV, T.; NIKZAD, N.; CHENAGHLU, M.; SOCHER, R.; AMATRIAIN, X.; GAO, J. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

OLIVEIRA, E. S. d. Pluto: um sistema de chatbot que utiliza ia e a abordagem rag para responder dúvidas sobre o sigaa. UFAL Campus Arapiraca, 2025.

OPENAI. *OpenAI API Reference*. 2025. Disponível em: <https://platform.openai.com>.

PANTOJA, A.; NASCIMENTO, K.; DUARTE, G.; SILVA, A.; MARTINS, Y. Desenvolvimento de um chatbot inteligente para auxílio de atividades organizacionais durante o período de processos seletivos da ufpa. In: *Anais do X Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software*. Porto Alegre, RS, Brasil: SBC, 2025. p. 143–154. ISSN 2763-874X. Disponível em: <https://sol.sbc.org.br/index.php/washes/article/view/35927>.

PATI, S.; ZAKI, Y. *Evaluating the Efficacy of Next.js: A Comparative Analysis with React.js on Performance, SEO, and Global Network Equity*. 2025. Disponível em: <https://arxiv.org/abs/2502.15707>.

PEREIRA, S. do L. Processamento de linguagem natural. 2019.

PEYKANI, P.; RAMEZANLOU, F.; TANASESCU, C.; GHANIDEL, S. Large language models: A structured taxonomy and review of challenges, limitations, solutions, and future directions. *Applied Sciences*, v. 15, n. 14, 2025. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/15/14/8103>.

PEYTON, K.; UNNIKRISHNAN, S.; MULLIGAN, B. A review of university chatbots for student support: Faqs and beyond. *Discover Education*, Springer, v. 4, n. 1, p. 21, 2025.

ROSA, J. M. d. Gestão de processos na administração pública: estudo sobre a recepção de documentação de calouros na universidade federal do rio grande do sul. 2024.

SHADCN. *Shadcn/UI Documentation*. 2025. Disponível em: <https://ui.shadcn.com>.

SHAO, M.; BASIT, A.; KARRI, R.; SHAFIQUE, M. Survey of different large language model architectures: Trends, benchmarks, and challenges. *IEEE Access*, IEEE, 2024.

SHARMA, C. *Retrieval-Augmented Generation: A Comprehensive Survey of Architectures, Enhancements, and Robustness Frontiers*. 2025. Disponível em: <https://arxiv.org/abs/2506.00054>.

SOURCE, M. O. *React Documentation*. 2025. Disponível em: <https://react.dev>.

Supabase Docs. *Hybrid Search — Combine keyword search with semantic search*. 2025. <https://supabase.com/docs/guides/ai/hybrid-search>. Acesso em: 4 de setembro de 2025.

TEAM, D. *Drizzle ORM Documentation*. 2025. Disponível em: <https://orm.drizzle.team>.

VALENTE, M. T. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Editora Independente, 2020. ISBN 978-65-00-01950-6. Disponível em: <https://engsoftmoderna.info/>.

VERCEL. *Next.js Documentation*. 2025. Disponível em: <https://nextjs.org>.

VERCEL. *Vercel AI Elements*. 2025. Disponível em: <https://vercel.com/ai>.

VERCEL. *Vercel AI SDK Documentation*. 2025. Disponível em: <https://sdk.vercel.ai>.

VIEIRA, N. M. d. R.; SANTOS, R. G. d. A.; SILVA, P. R. L. Reflexões sobre o prazer, o sofrimento e o adoecimento dos técnicos-administrativos nos institutos federais de educação. In: *Anais da 70ª Reunião Anual da SBPC*. Maceió, AL, Brasil: [s.n.], 2018. Disponível em: <https://www.sbpcnet.org.br/livro/70ra/trabalhos/resumos/2809_17cdfd281773b6337deab94124ad729d3.pdf>.

ZIMERMAN, F. E. Explorando chatbots baseados em llm para criação automática de questões práticas de programação de computadores. 2024.