

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Tárick Lorrán Batista Leite

**Sistema de Apoio Administrativo com Agentes
Inteligentes para Resolução de Dúvidas sobre
Estágio**

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Tárick Lorrán Batista Leite

**Sistema de Apoio Administrativo com Agentes Inteligentes
para Resolução de Dúvidas sobre Estágio**

Trabalho de conclusão de curso apresentado à
Faculdade de Computação da Universidade Fe-
deral de Uberlândia, como parte dos requisitos
exigidos para a obtenção título de Bacharel em
Sistemas de Informação.

Orientador: Prof. Dr. Victor Sobreira

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2025

**ATA DE DEFESA - GRADUAÇÃO**

Curso de Graduação em:	Sistemas de Informação, Campus Santa Mônica				
Defesa de:	Trabalho de Conclusão de Curso: GSI039 - Trabalho de Conclusão de Curso 1				
Data:	29/09/2025	Hora de início:	10:00	Hora de encerramento:	11:30
Matrícula do Discente:	12021BSI227				
Nome do Discente:	Tárick Lorrán Batista Leite				
Título do Trabalho:	Sistema com Agentes de IA para Suporte Administrativo Automatizado no Atendimento e Resolução de Dúvidas Frequentes sobre Estágio Supervisionado				
A carga horária curricular foi cumprida integralmente?	(x) Sim () Não				

Reuniu-se na [Sala Virtual do Microsoft Teams](#), da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Curso de Graduação em Sistemas de Informação, composta pelos professores: Dr. Adriano Mendonça Rocha - FACOM/UFU; Dr. Igor da Penha Natal - FACOM/UFU; Dr. Victor Sobreira - FACOM/UFU orientador do candidato.

Iniciando os trabalhos, o presidente da mesa, Dr. Victor Sobreira, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público e concedeu ao estudante a palavra para a exposição do seu trabalho. A duração da apresentação do estudante e o tempo de arguição e resposta foram conforme as normas do curso.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado com Nota [95].

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.



Documento assinado eletronicamente por **Victor Sobreira, Professor(a) do Magistério Superior**, em 29/09/2025, às 15:42, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Adriano Mendonça Rocha, Professor(a) do Magistério Superior**, em 29/09/2025, às 16:47, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Igor da Penha Natal, Professor(a) do Magistério Superior**, em 30/09/2025, às 08:29, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).

Agradecimentos

Primeiramente, agradeço a Deus, pela força, sabedoria e perseverança ao longo de toda a minha jornada.

Agradeço à minha esposa, Tawane, por sempre acreditar em mim, confiar em minhas decisões e me apoiar incondicionalmente.

Agradeço a minha mãe, Maura, que desde o início me incentivou a me dedicar e estudar.

Agradeço à minha cunhada, Ralciane, que neste momento final contribuiu com seus conhecimentos.

Agradeço ao meu orientador, Victor, que me apoiou nesta pesquisa de forma paciente, sempre disposto a me ajudar.

Agradeço à minha família, pelo amor, apoio constante e por estarem sempre comigo nos momentos bons e difíceis.

A todos que, de alguma forma, contribuíram para a realização deste trabalho, meu sincero obrigado.

Lista de ilustrações

Figura 1 – Diagrama de pacotes Fonte: O autor, 2025	25
Figura 2 – Fluxo de Funcionamento do Sistema Fonte: O autor, 2025	27
Figura 3 – Diagrama de Componentes Fonte: O autor, 2025	28
Figura 4 – Rota para criar a coleção Fonte: Postman, 2025	29
Figura 5 – Criando a coleção demonstracao-teste Fonte: Postman, 2025	29
Figura 6 – Função para criar a coleção Fonte: O autor, 2025	30
Figura 7 – Coleção criada no Qdrant. Podemos perceber o tamanho do vetor definido e também a utilização da distância cosseno Fonte: Qdrant, 2025	30
Figura 8 – Rota para upload do documento Fonte: O autor, 2025	31
Figura 9 – Função para processar o arquivo Fonte: O autor, 2025	31
Figura 10 – Postman - Upload de documento Fonte: Postman, 2025	32
Figura 11 – Log - Tokenização por sentença Fonte: O autor, 2025	32
Figura 12 – Log - Tokenização Fonte: O autor, 2025	32
Figura 13 – Log - <i>Chunk</i> gerado Fonte: O autor, 2025	33
Figura 14 – Função para armazenar o arquivo no MongoDB Fonte: O autor, 2025	33
Figura 15 – Log - Representação semânticas (embeddings) Fonte: O autor, 2025	34
Figura 16 – Qdrant: Quantidade de vetores cadastrados Fonte: Qdrant, 2025	34
Figura 17 – Log - Embedding da Pergunta Fonte: O autor, 2025	35
Figura 18 – Postman - Resultado da pergunta Fonte: O autor, 2025	40
Figura 19 – Modelagem do Banco de Dados Fonte: O autor, 2025	41
Figura 20 – Envio da pergunta para o orquestrador Fonte: Postman, 2025	42
Figura 21 – IA Agent - Log do Orquestrador Fonte: O autor, 2025	44
Figura 22 – Agente de IA - Log do Gerente Fonte: O autor, 2025	44
Figura 23 – Agente de IA - Log da Ação do Gerente Fonte: O autor, 2025	44
Figura 24 – Agente de IA - Log do Agente Fonte: O autor, 2025	44
Figura 25 – Agente de IA - Resultado Final da Execução Fonte: O autor, 2025	45
Figura 26 – Configuração do Bot Framework Fonte: Microsoft Bot Framework Emulator, 2025	47
Figura 27 – Resposta da Pergunta no Chat Bot Framework Fonte: Microsoft Bot Framework Emulator, 2025	48
Figura 28 – Resposta da Pergunta 1 Fonte: Microsoft Bot Framework Emulator, 2025	51
Figura 29 – Resposta da Pergunta 2 Fonte: Microsoft Bot Framework Emulator, 2025	52
Figura 30 – Resposta da Pergunta 3 Fonte: Microsoft Bot Framework Emulator, 2025	53
Figura 31 – Resposta da Pergunta 4 Fonte: Microsoft Bot Framework Emulator, 2025	54
Figura 32 – Resposta da Pergunta 5 Fonte: Microsoft Bot Framework Emulator, 2025	55
Figura 33 – Resposta da Pergunta 6 Fonte: Microsoft Bot Framework Emulator, 2025	55
Figura 34 – Resposta da Pergunta 7 Fonte: Microsoft Bot Framework Emulator, 2025	56
Figura 35 – Resposta da Pergunta 8 Fonte: Microsoft Bot Framework Emulator, 2025	57

Figura 36 – Resposta da Pergunta 9 Fonte: Microsoft Bot Framework Emulator, 2025 57

Figura 37 – Resposta da Pergunta 10 Fonte: Microsoft Bot Framework Emulator, 2025 . . . 58

Figura 38 – Avaliação das respostas de cada pergunta 59

Sumário

1	INTRODUÇÃO	11
1.1	Objetivos	11
1.2	Organização do Trabalho	12
2	FUNDAMENTAÇÃO TEÓRICA	13
2.1	Processamento de Linguagem Natural	13
2.2	Large Language Models (LLMs)	14
2.3	Retrieval-Augmented Generation (RAG)	15
2.4	Chatbots	15
2.5	Engenharia de Prompt	16
3	TRABALHOS RELACIONADOS	17
4	METODOLOGIA	19
4.1	Caracterização da Pesquisa	19
4.2	Tecnologias	19
4.2.1	Linguagem Python e Bibliotecas em Destaque	20
4.2.2	Linguagem TypeScript e Bibliotecas em Destaque	20
4.2.3	Microsoft Bot Framework	20
4.2.4	Gemini e API	20
4.2.5	Bancos de Dados	21
4.2.6	Testes e Modelagem do Sistema	21
4.3	Etapas de Desenvolvimento	22
4.4	Avaliação	22
5	DESENVOLVIMENTO DO SISTEMA	24
5.1	Projeto e Modelagem Geral do Sistema	24
5.2	Fluxo de Funcionamento do Sistema	26
5.3	Visão Geral de Componentes	27
5.4	Desenvolvimento das APIs do RAG	29
5.4.1	Criação da coleção	29
5.4.2	Upload de Documentos (Indexação)	30
5.4.3	Recuperação e Geração (Retrieval Generation)	34
5.5	Desenvolvimento do Agente de IA	40
5.5.1	Criação dos Agentes	41
5.5.2	Criação dos Prompts	45
5.6	Desenvolvimento do ChatBot	46
5.6.1	Criação do aplicativo de Chat para Microsoft Teams	46

5.7	Repositório e Controle de Versões	48
6	RESULTADOS	50
6.1	Perguntas, Respostas e Análises	50
6.2	Análise Geral	58
6.3	Limitações e Trabalhos Futuros	61
7	CONCLUSÃO	63
	REFERÊNCIAS	64

Resumo

Este trabalho tem como objetivo desenvolver um sistema baseado em Agentes de Inteligência Artificial para auxiliar a Coordenação de Estágio no atendimento e na resolução de dúvidas dos estudantes relacionadas aos processos de estágio supervisionado. Para isso, foi implementado um sistema *Retrieval-Augmented Generation (RAG)* com a base de dados vetorial *Qdrant*, permitindo o armazenamento e a consulta de documentos como normas, leis e regulamentos. A arquitetura de agentes foi projetada de modo a possibilitar o cadastro de gerentes, agentes e ferramentas, organizando e separando suas funcionalidades específicas. A solução foi estruturada para que, ao receber uma pergunta do usuário, o agente acesse a base de dados, recupere informações relevantes e apresente a resposta por meio de um sistema de *chat* integrado ao *Microsoft Bot Framework*. Algumas das limitações encontradas envolveram: a dependência da qualidade dos documentos indexados e as dificuldades relacionadas à reformulação automática de perguntas. Os resultados indicam que o sistema foi capaz de fornecer respostas adequadas e claras, com desempenho satisfatório, confirmando que há grande potencial de aplicação de sistemas similares ao proposto neste trabalho.

Palavras-chave: Agentes de Inteligência Artificial, *Retrieval-Augmented Generation*, *Qdrant*, Coordenação de Estágio, *Microsoft Bot Framework*.

Abstract

This work aims to develop a system based on Artificial Intelligence Agents to assist the Internship Coordination in handling and resolving students' inquiries related to supervised internship processes. To achieve this, a *RAG* system was implemented with the *Qdrant* vector database, enabling the storage and retrieval of documents such as regulations, laws, and guidelines. The agent architecture was designed to allow the registration of managers, agents, and tools, organizing and separating their specific functionalities. The solution was structured so that, upon receiving a user's question, the agent accesses the database, retrieves relevant information, and provides a response through a *chat* system integrated with the *Microsoft Bot Framework*. Some limitations encountered included the dependence on the quality of the indexed documents and difficulties related to the automatic reformulation of questions. The results indicate that the system was able to provide clear and adequate answers with satisfactory performance, confirming the great potential for applying systems similar to the one proposed in this work.

Keywords: Artificial Intelligence Agents, Retrieval-Augmented Generation, Qdrant, Internship Coordination, Microsoft Bot Framework.

Lista de Abreviaturas e Siglas

API *Application Programming Interface*. [19–21](#), [24](#), [25](#), [28](#), [29](#), [47](#), [48](#)

BIA Bradesco Inteligência Artificial. [11](#)

COCIC Colegiado do Curso de Ciência da Computação. [17](#)

CoT *Chain-of-Thought*. [16](#), [45](#)

FACOM Faculdade de Computação. [11](#), [17](#)

GPT *Generative Pre-trained Transformer*. [14](#)

IA Inteligência Artificial. [11](#), [13](#), [14](#), [17–22](#), [24](#), [26](#), [28](#), [40](#), [46](#), [48–50](#), [55](#), [61](#), [63](#)

LLM *Large Language Model*. [12–16](#), [19–22](#), [40](#), [43](#), [44](#)

PLN Processamento de Linguagem Natural. [12–14](#), [17](#), [20](#), [32](#)

RAG *Retrieval-Augmented Generation*. [8](#), [9](#), [12](#), [13](#), [15](#), [19–22](#), [24–26](#), [28](#), [29](#), [31](#), [41](#), [42](#), [48](#), [49](#), [52](#), [60–62](#)

UFOP Universidade Federal de Ouro Preto. [17](#)

UFRJ Universidade Federal do Rio de Janeiro. [16](#)

UFU Universidade Federal de Uberlândia. [11](#)

UML *Unified Modeling Language*. [19](#), [21](#)

1 Introdução

Em 2020, durante a pandemia do *COVID-19*, as instituições de ensino tiveram que se adaptar rapidamente para manter as atividades acadêmicas no período de isolamento social. A adoção do ensino remoto foi crucial para minimizar os prejuízos causados pela ausência de aulas presenciais (CORDEIRO, 2020). Mesmo em 2025, na Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia (UFU) ainda é perceptível a frequência no uso de ferramentas de comunicação, como o *Microsoft Teams*, no cotidiano dos estudantes.

Um estudo conduzido em 2021 pelo Grupo Sercom, provedor de serviços e plataformas de experiência do cliente, em parceria com o instituto de pesquisas Qualibest, entrevistou 1.081 pessoas de todas as regiões do Brasil para analisar os canais de contato preferidos entre consumidores e empresas. O *chat* nas plataformas de suporte liderou a preferência, sendo escolhido por 51% dos participantes, seguido de perto pelo WhatsApp, utilizado por 49%. O e-mail ocupou a terceira posição, com 47%, enquanto o telefone, cada vez menos utilizado, apareceu em quarto lugar, com 43% da preferência (PACHECO, 2021).

Diariamente, as organizações têm adotado a tecnologia como meio para oferecer à sociedade soluções inovadoras e autônomas, incorporando recursos digitais às suas estratégias corporativas com o objetivo de impulsionar a transformação digital. Nesse contexto, surgiram os assistentes virtuais, também conhecidos como *chatbots*, que introduziram uma nova abordagem para a automatização do atendimento. Essas ferramentas proporcionam um canal alternativo para que os usuários solucionem suas demandas de forma totalmente tecnológica, reduzindo a necessidade de interação humana (SCHUNK, 2020). Um exemplo desses avanços ocorreu com o Banco Bradesco, que implementou a assistente virtual Bradesco Inteligência Artificial (BIA) em 2016, que já soma mais de 2 bilhões de interações (MARTINS, 2023). Com a atualização da BIA para utilizar Inteligência Artificial (IA) Generativa em 2023, os primeiros resultados já podem ser observados: o Bradesco alcançou índices significativos, com uma taxa de resolutividade de 82% no primeiro nível de atendimento e uma retenção de 89%, impactando diretamente o autoatendimento nos canais digitais (MICROSOFT, 2024).

1.1 Objetivos

Este trabalho tem como objetivo desenvolver um Agente de IA integrado a um *chatbot* que poderá ser utilizado no *Microsoft Teams*, voltado para apoiar a Coordenação de Estágios. A solução busca enfrentar problemas, como a alta demanda por informações, a repetição de dúvidas frequentes e o retrabalho em tarefas que poderiam ser automatizadas, fatores que geram sobrecarga e consomem tempo que poderia ser destinado a outras atividades. O sistema pretende oferecer respostas rápidas e eficientes, contribuindo para otimização do atendimento aos estudantes.

Além disso, a aplicação será validada por meio de um estudo de caso realizado junto à Coordenação, permitindo avaliar sua efetividade na prática.

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

- Realizar o levantamento de requisitos junto à Coordenação de Estágio, identificando as principais necessidades e demandas dos estudantes e da equipe administrativa;
- Implementar um sistema de *RAG* para armazenamento e recuperação de informações relevantes, como leis, normas e processos administrativos relacionados ao estágio;
- Projetar uma arquitetura de agentes que possibilite a gestão de ferramentas e fluxos de atendimento;
- Disponibilizar uma interface de *chatbot* para interação, proporcionando respostas claras e rápidas aos estudantes;
- Avaliar o desempenho da solução, considerando a precisão das respostas, e o tempo de atendimento.

1.2 Organização do Trabalho

Este trabalho está estruturado da seguinte forma:

- Seção 2 - **Fundamentação Teórica:** Apresenta os conceitos que sustentam o desenvolvimento da solução, incluindo *Processamento de Linguagem Natural (PLN)*, *Large Language Models (LLMs)*, *RAG* e *Chatbots*.
- Seção 3 - **Trabalhos Relacionados:** Apresenta a revisão bibliográfica de pesquisas e soluções existentes, destacando métodos, técnicas e abordagens relevantes para o desenvolvimento da solução proposta.
- Seção 4 - **Metodologia:** Descreve a caracterização da pesquisa, a abordagem utilizada, as etapas de desenvolvimento do sistema e as métricas de avaliação aplicadas para analisar as respostas do agente.
- Seção 5 - **Desenvolvimento do Sistema:** Expõe o processo de construção do sistema, detalhando a modelagem, a arquitetura, os fluxos de funcionamento, decisões de design e a integração das tecnologias utilizadas.
- Seção 6 - **Resultados:** Apresenta a análise dos resultados obtidos com o sistema, verificando sua capacidade de atender às solicitações dos usuários, identificando pontos fortes, limitações e oportunidades de melhorias.
- Seção 7 - **Conclusão:** Apresenta as considerações finais do trabalho, destacando as contribuições, limitações e propostas para trabalhos futuros.

2 Fundamentação Teórica

Esta Seção apresenta os fundamentos que sustentam o desenvolvimento da solução proposta. São discutidos os principais conceitos teóricos que embasam a pesquisa, incluindo [PLN](#), [LLMs](#), [RAG](#), *Chatbots* e Engenharia de *Prompt*. Esses tópicos fornecem a base necessária para compreender de que forma a [IA](#) pode ser aplicada no contexto do atendimento administrativo.

O [PLN](#) representa a área da computação responsável por permitir que máquinas interpretem e produzam linguagem humana. Os [LLMs](#) ampliam essas capacidades, trazendo maior sofisticação às interações e possibilitando respostas mais contextualizadas. A abordagem [RAG](#) complementa esses modelos ao incorporar informações externas, garantindo resultados mais atualizados e confiáveis.

Nesse contexto, a Engenharia de *Prompt* surge como um conjunto de técnicas voltadas a otimizar a comunicação entre o usuário e o modelo de linguagem, permitindo direcionar o comportamento da [IA](#) por meio de instruções elaboradas. Essa prática tem papel fundamental na obtenção de respostas mais precisas e alinhadas aos objetivos da aplicação.

Por fim, os *Chatbots* demonstram como essas tecnologias podem ser aplicadas em interfaces de comunicação, servindo como ponto de contato direto entre usuários e sistemas inteligentes.

2.1 Processamento de Linguagem Natural

Com o avanço da Computação, as mudanças no dia a dia são cada vez mais perceptíveis, desde a criação do *PIX* para transferências instantâneas até o desenvolvimento de novos modelos de Inteligência Artificial. Nesse contexto, o [PLN](#) se destaca ao possibilitar uma comunicação mais fluida entre humanos e máquinas. É uma área da computação que lida com a interpretação e geração da comunicação humana em suas diversas formas, incluindo sons, palavras, frases e textos completos. Ele abrange aspectos como estrutura, significado, contexto e uso da linguagem, permitindo que os computadores interajam de maneira mais natural com os seres humanos ([GONZALEZ; LIMA, 2003](#)).

Os estudos e aplicação do [PLN](#) tiveram início por volta da década de 1950, com o objetivo de utilizar a linguagem natural como um meio de comunicação entre humanos e computadores, sendo uns dos pioneiros nessa área o Alan Turing, que desenvolveu o famoso *Teste de Turing*, um experimento projetado para avaliar a capacidade de uma máquina em simular o comportamento humano por meio da inteligência artificial ([PACHECO, 2021](#)).

Em 1966, Joseph Weizenbaum desenvolveu a Eliza, que simulava um psicólogo. Ela interagia com os usuários realizando perguntas e consequentemente dava continuidade na consulta. As perguntas realizadas faziam com que os pacientes aprofundassem em seus problemas em

detalhes. Consequentemente as pessoas não sabiam que estavam interagindo com um robô, mas sim com um humano de verdade o que marcou os primeiros passos de pesquisa sobre IA (MANFIO; MORENO, 2018).

Atualmente, um grande nome nos avanços tecnológicos que estamos presenciando é da empresa *OpenAI*, que lançou em 2022 a sua própria ferramenta de IA. A empresa desenvolveu o *ChatGPT*, um modelo baseado na arquitetura *Generative Pre-trained Transformer (GPT)*. Este modelo foi treinado com grandes volumes de dados textuais e aplicado a um *chatbot* e dessa forma consegue responder a dúvidas e solicitações dos usuários, podendo escrever desde textos e até mesmo programas computacionais (CUNHA, 2023; SANTOS, 2023).

O sucesso alcançado pela *OpenAI* estimulou o surgimento de novos modelos de IA desenvolvidos por outras empresas, como o *Gemini*, da *Google*, e o modelo recente da *startup* chinesa *DeepSeek*. Um dos principais diferenciais do *Gemini* é sua capacidade multimodal, que permite o processamento de diferentes tipos de dados, como texto, imagens e código (SÁ, 2024).

Já o *DeepSeek*, apesar de novo no cenário da IA, apresenta recursos especializados direcionados a um público específico, destacando-se como uma opção competitiva, principalmente em análise de dados e recuperação de informações (ARIDAN et al., 2025).

2.2 Large Language Models (LLMs)

Com os avanços do PLN, surgiram modelos cada vez mais sofisticados, capazes de compreender e gerar textos de forma semelhante à comunicação humana. Entre essas inovações, destacam-se os *Large Language Models (LLMs)*, ou Modelos de Linguagem de Grande Escala, que têm sido amplamente explorados no campo da inteligência artificial devido às suas capacidades avançadas de interpretação e geração de linguagem natural. Esses modelos são treinados em extensos conjuntos de dados e estruturados com bilhões de parâmetros, permitindo uma performance altamente eficiente na produção de textos (VOGEL; RAMOS; FRANZONI, 2025).

Enquanto o PLN faz parte do aprendizado de máquina e capacita os computadores a compreender, interpretar e reproduzir a linguagem humana, os *LLMs* representam uma evolução desses modelos. Eles são geralmente estruturados com redes neurais profundas, contendo um grande número de parâmetros e treinados com enormes quantidades de texto, o que os torna altamente eficientes para diversas tarefas relacionadas à linguagem (RAMOS, 2023).

As aplicações computacionais baseadas em *LLMs* desempenham um papel fundamental em diversas etapas do processo acadêmico, desde a formulação de pesquisas até a redação e publicação de artigos científicos. Essas tecnologias auxiliam na busca por literatura relevante, permitem interações baseadas em perguntas e respostas sobre artigos e contribuem para a análise e revisão de conteúdo (RAMOS, 2023). Entre os exemplos de *LLMs* mais utilizados atualmente, destacam-se os desenvolvidos pela *OpenAI*, como o *ChatGPT*, e pelo *Google*, como o *Gemini*.

2.3 Retrieval-Augmented Generation (RAG)

Os modelos de *LLMs* possuem capacidades notáveis de processamento, mas ainda enfrentam limitações, o que leva à ocorrência de alucinações devido ao uso de informações desatualizadas; afinal, os *LLMs* são treinadas com informações do cotidiano. As alucinações em modelos de linguagem acontecem quando eles produzem respostas que parecem corretas, mas são falsas e pode ocorrer até mesmo com perguntas simples (OPENAI, 2025).

Nesse cenário, o *RAG* surge como uma alternativa, ao integrar dados provenientes de fontes externas. Essa abordagem aumenta a precisão e a confiabilidade das respostas, especialmente em tarefas que demandam alto nível de conhecimento, além de permitir atualizações contínuas e a incorporação de informações especializadas (GAO et al., 2024).

Em geral, o *RAG* funciona da seguinte forma:

- **Indexação (*Indexing*):** Limpeza e extração de dados brutos de diferentes formatos como *PDF*, *HTML*, *Word* e *Markdown*, que são convertidos para texto simples e padronizado, e em seguida o conteúdo é segmentado em blocos menores. Esses blocos são codificados em representação vetoriais por meio de um modelo de *embedding* e armazenados em um banco de dados vetorial, possibilitando a busca eficiente por similaridade na etapa de recuperação (GAO et al., 2024).
- **Recuperação (*Retrieval*):** A consulta do usuário é codificada no mesmo formato vetorial utilizado na indexação, e então calcula as pontuações de similaridade com os vetores dos documentos indexados. Os blocos mais relevantes (top k) são recuperados e utilizados para ampliar o contexto do *prompt* (GAO et al., 2024).
- **Geração (*Generation*):** A consulta e os documentos selecionados são combinados para formar um *prompt* coerente, ao qual o modelo de linguagem gera uma resposta. Dependendo da tarefa, a resposta pode se basear no conhecimento interno do modelo ou restringir-se ao conteúdo recuperado (GAO et al., 2024).

2.4 Chatbots

O termo *chatbot* é formado pela combinação das palavras em inglês *chat* (aquele que conversa) e *bot* (abreviação de *robot*), referindo-se a um robô em forma de *software* que interage com pessoas por meio de diálogos. Assim, pode ser definido como um programa robótico capaz de se comunicar com seres humanos, simulando o comportamento de uma pessoa durante uma conversa, respondendo perguntas e mantendo uma interação fluida. Isso cria a impressão de que o usuário está dialogando com outro ser humano, e não com um *software* (COSTA, 2023).

Os *chatbots* auxiliam na automatização de tarefas e são amplamente utilizados no atendimento ao cliente por empresas de pequeno, médio e grande porte em diversas áreas. Essas

ferramentas são capazes de reduzir custos operacionais e o tempo de resposta, tornando o atendimento mais ágil e eficiente (SOUZA; SILVA, 2021).

Esses programas de computador podem se comunicar de duas formas principais. A primeira ocorre por meio de elementos da *interface*, onde o usuário é restringido a um conjunto de ações predefinidas. A segunda forma utiliza o processamento de linguagem natural, permitindo maior flexibilidade e possibilitando entradas variadas sem limitações específicas (ARAUJO, 2020).

No contexto educacional, os *chatbots* têm sido implementados para auxiliar estudantes em diversas demandas, desde informações administrativas até suporte acadêmico. Um exemplo relevante é a solução desenvolvida pela Universidade Federal do Rio de Janeiro (UFRJ), que aprimorou o atendimento aos estudantes ao integrar diferentes serviços e facilitar o acesso a informações acadêmicas relevantes (ARAUJO, 2020).

2.5 Engenharia de Prompt

A engenharia de *prompts* é uma técnica essencial para ampliar as capacidades das *LLMs*. Em vez de realizar ajustes diretos nos parâmetros internos do modelo, essa abordagem utiliza instruções específicas para orientar o comportamento da IA em diferentes tarefas, aplicando novos contextos a partir da formulação adequada do *prompt* (SAHOO et al., 2025). Dessa forma, a maneira como o comando é estruturado passa a influenciar diretamente a qualidade, a coerência e a profundidade das respostas geradas.

Dentro desse contexto, diversas estratégias de engenharia de *prompts* foram desenvolvidas com o objetivo de aprimorar a capacidade de raciocínio e tomada de decisão das *LLMs*. Entre as principais estratégias destaca-se o *Chain-of-Thought (CoT)*, voltada a reduzir as limitações dos modelos de linguagem em tarefas que exigem raciocínio complexo. Essa técnica orienta o modelo a estruturar seu raciocínio de forma progressiva, apresentando o processo de pensamento em etapas até chegar à resposta final. Diferentemente de um *prompt* tradicional, que consiste em dar ao modelo uma instrução ou descrição direta da tarefa, o *CoT* incentiva o modelo a explicitar a sequência lógica de raciocínio, o que resulta em respostas mais organizadas e consistentes (SAHOO et al., 2025).

A partir dessa capacidade de raciocínio estruturado, surge o *ReAct Prompting*, que amplia a abordagem anterior ao integrar raciocínio e ação. Essa técnica permite que modelos de linguagem gerem, de maneira intercalada, tanto o raciocínio (*reasoning traces*) quanto as ações específicas de uma tarefa, promovendo uma sinergia entre pensar e agir. Com isso, o modelo é capaz de induzir, acompanhar e atualizar planos de ação de forma dinâmica, além de lidar com exceções durante a execução (SAHOO et al., 2025).

3 Trabalhos Relacionados

Nesta Seção é apresentada uma revisão bibliográfica que aborda soluções já desenvolvidas, permitindo identificar ferramentas e resultados relevantes que servem de base para este trabalho. Ao todo, são discutidos três trabalhos distintos.

O trabalho de [Rocha \(2023\)](#) analisado, propõe uma solução *Web* responsiva voltada à automatização das etapas do processo de estágio da [FACOM](#), com o objetivo de facilitar o fluxo de atividades para todas as partes envolvidas. A abordagem adotada utiliza máquinas de estados e uma arquitetura em camadas (*frontend*, *backend* e banco de dados), com transições baseadas em entradas dos usuários e eventos temporais. Como resultado, foi desenvolvido um protótipo funcional que se destaca pela manutenibilidade, flexibilidade e facilidade de evolução, além de oferecer uma visão integrada dos fluxos do sistema. A minha proposta complementa esse objetivo ao incorporar uma camada inteligente capaz de esclarecer dúvidas, analisar documentos e oferecer um suporte mais dinâmico, interativo e alinhado às demandas dos estudantes e da coordenação.

Referente ao trabalho de [PACHECO \(2021\)](#), desenvolve uma ferramenta de *chatbot* com o uso de [PLN](#) para as plataformas *Telegram* e *WhatsApp*, visando otimizar o atendimento ao cliente em uma empresa de informática. A metodologia adotada envolveu a criação de diferentes versões de *chatbots* utilizando tecnologias como *Python* com *Telethon*, *DialogFlow* e *ChatRace*, possibilitando a comparação entre as abordagens quanto à facilidade de desenvolvimento, integração com serviços externos, custo e qualidade da experiência conversacional. Os resultados indicaram que todas as soluções apresentaram bom desempenho, com destaque para o uso do *ChatRace*, considerado o mais viável para a empresa analisada devido à sua simplicidade, recursos integrados e qualidade no atendimento. O trabalho envolve diferentes ferramentas, algumas das quais serviram de base para este trabalho, como o *Microsoft Bot Framework*, adotado aqui como *interface* de comunicação entre o usuário e o Agente de [IA](#) por meio do *chat*.

[MONTEIRO \(2021\)](#) apresenta o *chatbot* Helena, criado com o objetivo de auxiliar o [Colegiado do Curso de Ciência da Computação \(COCIC\)](#) da [Universidade Federal de Ouro Preto \(UFOP\)](#) no atendimento aos estudantes, especialmente em relação às dúvidas mais recorrentes sobre trâmites acadêmicos, como trancamento de disciplinas e colação de grau. A proposta inclui a elaboração de uma arquitetura funcional para a disponibilização do sistema via internet, validação da ferramenta por meio de questionários simulados e a coleta anônima de dados para futuras análises. Os resultados foram bastante positivos, demonstrando que o *chatbot* oferece uma boa experiência de uso e precisão nas respostas, além de não demandar custos com infraestrutura, o que reforça a viabilidade de sua implementação. O trabalho analisado apresenta semelhanças com o sistema proposto, diferenciando pelo foco específico na área de estágio e pela incorporação de [IA](#) Generativa. Esse recurso permite ao agente compreender, interagir e executar tarefas administrativas de forma mais dinâmica e eficiente.

Por fim, os três trabalhos analisados se relacionam diretamente com a proposta nesta monografia, que consistiu no desenvolvimento de um Agente de IA utilizando IA Generativa e integrando-o a uma ferramenta de *chatbot*. O objetivo é apoiar a coordenação de estágio no atendimento aos estudantes, tendo como função principal o esclarecimento de dúvidas. Futuramente, a solução poderá ser expandida para automatizar processos, como a análise de documentos e outras atividades administrativas, contribuindo para a otimização do trabalho da coordenação e para um atendimento mais ágil e eficiente aos estudantes.

4 Metodologia

Esta Seção apresenta a caracterização da pesquisa e a metodologia utilizada. Além disso, é apresentado as métricas de avaliação com as perguntas levantadas e as respostas esperadas e obtidas pelo agente desenvolvido.

4.1 Caracterização da Pesquisa

Esta pesquisa é caracterizada como aplicada, pois busca desenvolver uma solução prática voltada à melhoria do atendimento da Coordenação de Estágios. Quanto aos procedimentos, combina uma investigação bibliográfica, utilizada para fundamentar teoricamente o estudo, com uma abordagem experimental, que envolve a construção e a validação de um protótipo de *software*. Em relação à forma de análise, adota-se uma perspectiva quali-quantitativa, unindo a avaliação qualitativa da aplicabilidade da solução com a avaliação quantitativa de aspectos como desempenho e tempo de resposta.

O estudo consiste no desenvolvimento de um sistema de informações em formato de *chatbot*, aplicando tecnologias como *RAG* e Agentes de *IA* para a plataforma do *Microsoft Teams*, com o objetivo de fornecer assistência aos estudantes, esclarecendo dúvidas relacionadas ao Estágio Supervisionado.

O desenvolvimento foi realizado utilizando a linguagem de programação *Python* e *TypeScript*, com integração de ferramentas como *Qdrant* para armazenamento dos dados vetorizados dos documentos de Estágio Supervisionado, *Sentence Transformers* para a criação dos *embeddings*, a *Application Programming Interface (API)* gratuita da *LLM Gemini* para o processamento das solicitações dos usuários e por fim, a utilização da base de dados *NoSQL MongoDB* para armazenamento dos documentos indexados, histórico de conversas e configurações dos agentes. Além disso, utilizamos como *interface* para testes o emulador *Microsoft Bot Framework* que possibilitou a interação via *chat* entre o usuário com o agente.

4.2 Tecnologias

Esta Seção apresenta as principais tecnologias utilizadas no desenvolvimento do Agente de *IA*. Cada uma delas foi escolhida por sua familiaridade na utilização como também por sua relevância e aderência aos conceitos teóricos discutidos na Seção 2.

Para melhor organização, as seções seguintes descrevem as tecnologias empregadas e sua relação com os fundamentos abordados anteriormente: a linguagem *Python* e suas bibliotecas, a linguagem *TypeScript* e bibliotecas associadas, o modelo *Gemini* e sua *API*, os bancos de dados utilizados, a ferramenta *Postman* para testar as *APIs* e por fim, os *Unified Modeling Languages*

(*UMLs*) empregados no projeto.

4.2.1 Linguagem Python e Bibliotecas em Destaque

A linguagem *Python*¹ foi escolhida em razão de sua simplicidade, vasta comunidade e ampla disponibilidade de bibliotecas. No contexto deste trabalho, *Python* é o elo principal entre os fundamentos teóricos discutidos na Seção 2 e a implementação prática do sistema.

Entre as bibliotecas utilizadas, destacam-se *Sentence Transformers* e a *PLN NLTK* que foram empregadas na criação de *embeddings*², etapa essencial para a implementação do *RAG* (Seção 2.3). Já o *FastAPI* foi utilizado para a construção das *APIs* que conectam os módulos do sistema, garantindo eficiência na comunicação entre o Agente de *IA* e o *chatbot*.

4.2.2 Linguagem TypeScript e Bibliotecas em Destaque

Para o desenvolvimento do *chatbot* (Seção 2.4), foi utilizado a linguagem *TypeScript*³ seguindo os padrões recomendados para aplicativos *Microsoft Teams* (IGNITE, 2025). A principal biblioteca utilizada foi a *botbuilder* que serve para a construção de *bots* oferecendo a infraestrutura necessária para lidar com mensagens, manter contexto da conversa e integrar ao *Microsoft Teams*.

Outra biblioteca relevante é o *Express*⁴, utilizada na construção das *APIs* que conectam o *chatbot* ao *Teams* e disponibilizam um *webhook* para o retorno das respostas geradas pelo agente.

4.2.3 Microsoft Bot Framework

Para a realização dos testes, foi utilizado o *Microsoft Bot Framework* que possibilita a *interface* do sistema com os usuários por meio do *chatbot*. Ele está relacionado à Seção 2.4, que aborda os *chatbots* como ferramentas que viabilizam a interação entre humanos e o Agente de *IA*, funcionando como uma camada de comunicação para transmitir as respostas geradas pelo agente de forma acessível aos estudantes.

4.2.4 Gemini e API

Para o modelo de *LLM*, foi utilizado o *Gemini*⁵, discutido na Seção 2.2. Ele atua como o núcleo de raciocínio do agente sendo responsável por interpretar perguntas e gerar respostas em linguagem natural.

¹ <<https://www.python.org/>>

² Embeddings são representações vetoriais geradas por modelos de aprendizado de máquina, cujo objetivo é capturar características semânticas ou significativas de um objeto (como palavras, sentenças ou documentos) (CLOUDEFLARE, 2025), permitindo que informações complexas sejam comparadas em um espaço matemático.

³ <<https://www.typescriptlang.org/>>

⁴ <<https://expressjs.com/>>

⁵ <<https://gemini.google/about/?hl=pt-BR>>

Para que a *LLM* produza respostas contextualmente relevantes, ou seja, não utilize o conhecimento pré-treinado, o agente utiliza a pergunta do usuário para realizar a recuperação das informações no *RAG* e, em seguida, fornece este contexto recuperado ao modelo para gerar a resposta adequada.

O *Gemini*, além de ser um modelo avançado, a *Google* disponibiliza a *API* gratuitamente para desenvolvedores. O modelo utilizado para este trabalho foi *2.5-flash-preview-05-20*, que apresentou melhores desempenho durante os testes.

4.2.5 Bancos de Dados

Durante o desenvolvimento do trabalho, foram utilizados dois bancos de dados: *Qdrant*⁶ e *MongoDB*⁷.

O *Qdrant* é um banco de dados vetorial voltado para o armazenamento e recuperação das informações por similaridade. Ele foi utilizado na etapa de *RAG*, discutido na Seção 2.3, tanto para armazenar os textos extraídos dos documentos, quanto para registrar as conversas dos usuários, funcionando como uma memória de longo prazo. Este processo possibilita que o agente tenha informações atualizadas e mantenha o histórico de interações anteriores.

Já *MongoDB* é um banco de dados Não Relacional utilizado para armazenar os documentos anexados durante a realização do processo de *RAG*, permitindo a disponibilização das fontes para *download*. Além disso, o banco foi utilizado para armazenar as configurações dos agentes como, por exemplo, o *Managers*, *Agents* e *Tools*, bem como os *logs* de execução e histórico de conversas de curto prazo (Seção 2.4).

4.2.6 Testes e Modelagem do Sistema

O Postman⁸ é uma ferramenta voltada para o desenvolvimento e teste de *APIs*. Neste trabalho, foi utilizado para validar a comunicação entre os módulos do sistema, desde a indexação e recuperação de informações no *RAG* até os testes de interação do Agente de *IA* com o *chatbot*. Dessa forma, possibilitou simular cenários reais de uso, identificar inconsistências e garantir maior confiabilidade no fluxo de atendimento.

Além disso, a *UML* foi empregada neste trabalho para representar de forma visual a arquitetura e o funcionamento do sistema. Foram elaborados diagramas de pacotes, componentes e de sequência, que auxiliaram na compreensão das interações, na definição das responsabilidades dos agentes e na organização da solução proposta. Esses modelos contribuíram tanto para o planejamento quanto para a comunicação técnica do projeto.

⁶ <<https://qdrant.tech/>>

⁷ <<https://www.mongodb.com/>>

⁸ <<https://www.postman.com/>>

4.3 Etapas de Desenvolvimento

As etapas de desenvolvimento incluíram:

- Desenvolvimento do sistema de *RAG* utilizando a linguagem *Python*.
- Desenvolvimento do Agente de *IA* integrando-o ao *RAG* utilizando a linguagem *Python*.
- Base de dados como *Qdrant* para armazenamento dos dados vetorizados e o *MongoDB* para armazenamento de documentos e também das configurações dos agentes.
- Desenvolvimento do aplicativo para *Microsoft Teams* utilizando a linguagem *TypeScript* e integrando-o ao Agente de *IA*. Para testes, foi utilizado o emulador *Microsoft Bot Framework* como *interface* do *chatbot*.

4.4 Avaliação

Para avaliar as respostas do agente, os testes foram realizados em um computador com 16 GB de *RAM*, processador de 2,30 GHz e placa de vídeo dedicada *NVIDIA GeForce MX230* com 2 GB de memória.

Trinta e uma perguntas dos estudantes, devidamente anonimizadas, com as respectivas respostas dadas pela Coordenação de Estágios do Curso de Sistemas de Informação do Campus Monte Carmelo foram levantadas e pré-selecionadas para os testes com o sistema proposto. Além disso, outras dez perguntas e respostas elaboradas pela *LLM Gemini* também foram pré-selecionadas. Deste conjunto, dez perguntas foram efetivamente selecionadas e utilizadas nos testes, sendo cinco de cada grupo.

Como método de avaliação, adotamos critérios como Organização, Enriquecimento, Concisão, Alucinação, Completude e Precisão. Para mensurar cada critério, utilizamos a **escala de Likert**, variando as indicações entre: discordo plenamente, discordo, indiferente/neutro, concordo e concordo plenamente.

A seguir, definimos cada critério:

C1 - Precisão: A resposta corresponde a pergunta.

C2 - Completude: A resposta é completa, cobrindo todas as informações essenciais questionadas.

C3 - Alucinação: A resposta apresenta informações incorretas ou inventadas.

C4 - Concisão: A resposta é objetiva e concisa, contendo apenas as informações essenciais e sem dados desnecessários.

C5 - Enriquecimento: A resposta inclui informações complementares importantes, tais como documentos e links para acesso.

C6 - Organização: A resposta é bem estruturada, com ordenação coerente das ideias expostas.

5 Desenvolvimento do Sistema

Esta Seção apresenta o processo de construção do sistema proposto, desde as etapas iniciais de modelagem até a implementação final do Agente de IA. São descritos os diagramas utilizados para representar a arquitetura e os fluxos de funcionamento, bem como as decisões de *design* e a integração das tecnologias discutidas anteriormente. O objetivo é oferecer uma visão clara e organizada de como a solução foi estruturada, evidenciando os aspectos práticos que transformaram a fundamentação teórica em um protótipo funcional.

5.1 Projeto e Modelagem Geral do Sistema

Esta Seção apresenta o projeto e a modelagem geral do sistema desenvolvido. O objetivo é descrever a estrutura planejada da aplicação, destacando seus principais módulos e a forma como eles se relacionam para compor a solução proposta. Durante a fase de modelagem, foram definidos os componentes responsáveis pela *interface* com o usuário, processamento das solicitações e gerenciamento das informações.

O sistema foi concebido com base em uma arquitetura modular, na qual os diferentes componentes se comunicam por meio de *APIs*. Essa abordagem facilita a manutenção, a escalabilidade e a integração entre os módulos. Entre os principais componentes definidos estão o *chatbot*, o Agente de IA e o sistema *RAG*, que juntos formam a base da solução. O *chatbot* representa o ponto de contato com o usuário; o Agente de IA atua como orquestrador das interações; e o *RAG* funciona como núcleo de armazenamento e recuperação das informações necessárias para a geração das respostas.

A partir dessas definições, foi elaborado o diagrama de pacotes apresentado na Figura 1. O pacote é um elemento da modelagem orientada a objetos utilizado para agrupar componentes relacionados de uma especificação (MARTINS; DINIZ; SILVA, 2017), com o intuito de apresentar os pacotes organizados conforme as principais classes do sistema, facilitando a compreensão da sua estrutura e organização interna. Uma breve descrição dos principais pacotes desenvolvidos é apresentada a seguir:

- **Pacote Bot Teams:** Atua como o *frontend* e ponto de entrada para o usuário. É responsável por gerenciar a *interface* de *chat* e capturar as perguntas. Consome a *API* do Agente de IA para enviar as requisições do usuário e exibir as respostas geradas.
- **Pacote Agente de IA:** Funciona como o orquestrador central da aplicação. É responsável por gerenciar o estado da conversação, o perfil dos usuários e a lógica de acionamento de ferramentas. Para responder perguntas que exigem conhecimento específico, ele invoca os serviços do pacote *RAG* para obter o contexto necessário. Utiliza um Banco de Dados

NoSQL (*MongoDB*) para persistir o histórico de conversas e os dados de gerenciamento dos agentes.

- **Pacote Segurança:** Implementado como um *middleware* de *API*, este pacote é responsável por proteger os *endpoints* do *RAG*. Ele realiza a validação de *tokens* de acesso, garantindo que apenas o agente ou outros serviços autorizados possam consumir os dados da base de conhecimento.
- **Pacote *RAG*:** Este pacote constitui o núcleo da base de conhecimento. Ele expõe *APIs REST* com funcionalidades para o ciclo de vida dos documentos. Sua arquitetura de armazenamento foi projetada da seguinte forma:
 - Utiliza um Banco de Dados Vetorial (*Qdrant*) para armazenar os *chunks* de texto e seus *embeddings*, permitindo buscas por similaridade semântica;
 - Utiliza um Banco de Dados *NoSQL* (*MongoDB*) para armazenar os metadados dos documentos (nome, *hash*, relacionamentos) e os arquivos binários originais para *download*.

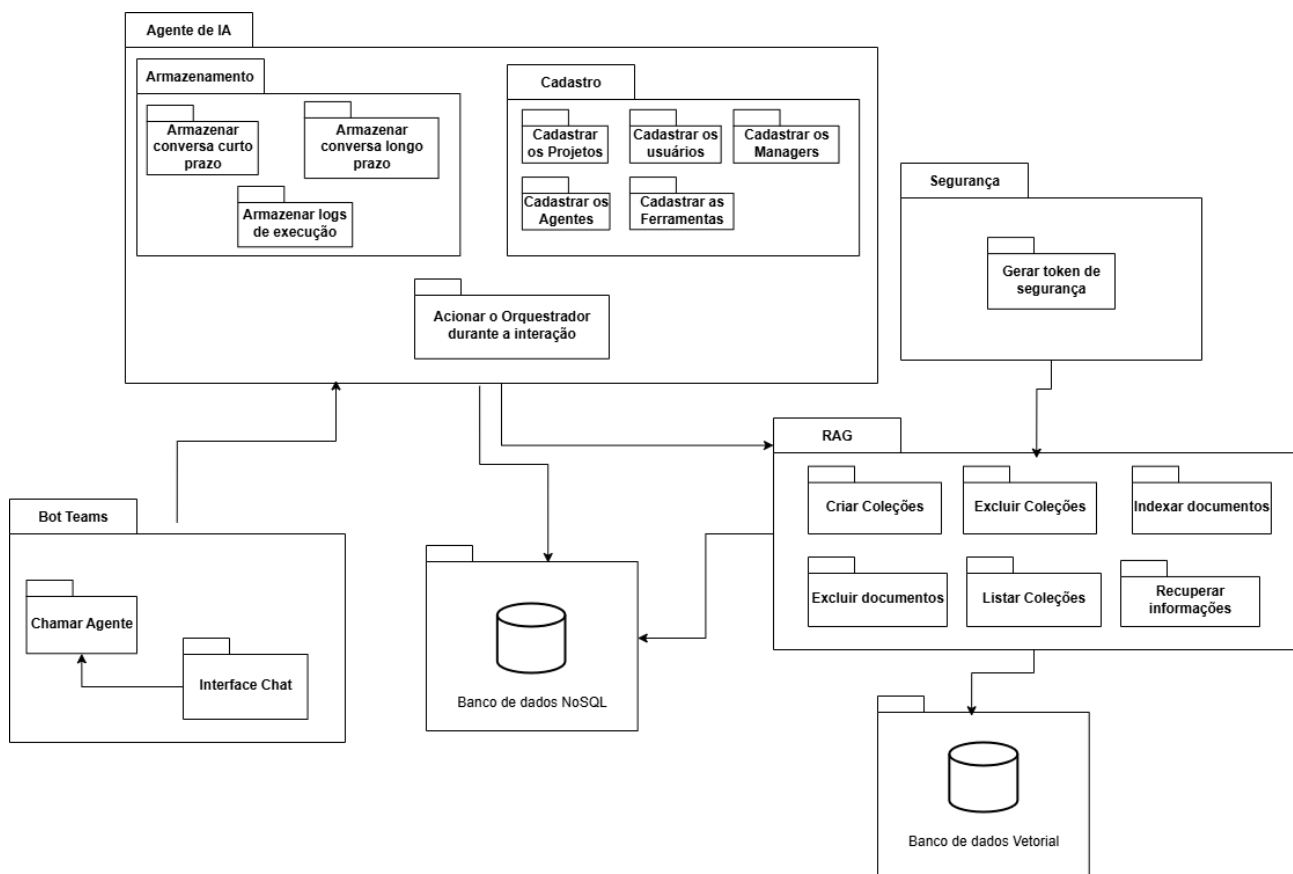


Figura 1 – Diagrama de pacotes

Fonte: O autor, 2025

5.2 Fluxo de Funcionamento do Sistema

O funcionamento do sistema foi modelado por meio de um fluxograma que representa, de forma sequencial, o caminho percorrido por uma solicitação realizada pelo usuário, como mostrado na Figura 2. Inicialmente, o usuário envia uma mensagem por meio do aplicativo de *chatbot*. Essa mensagem é então encaminhada para o agente, responsável por interpretar a requisição e coordenar as etapas seguintes do processamento.

O Agente de **IA** atua como um intermediador inteligente, sendo capaz de analisar o conteúdo da pergunta e encaminhá-la ao sistema **RAG**, que é o núcleo responsável pela recuperação de informações relevantes e pela geração da resposta. O **RAG** funciona realizando uma busca vetorial na base de conhecimento previamente cadastrada, identificando os trechos mais relevantes para a pergunta feita. Com base nesses dados, uma resposta é construída e retornada ao agente.

Por fim, o agente entrega a resposta final ao *bot*, que a repassa ao usuário dentro do ambiente do *Microsoft Teams*, completando o ciclo de atendimento de forma automatizada e eficiente.

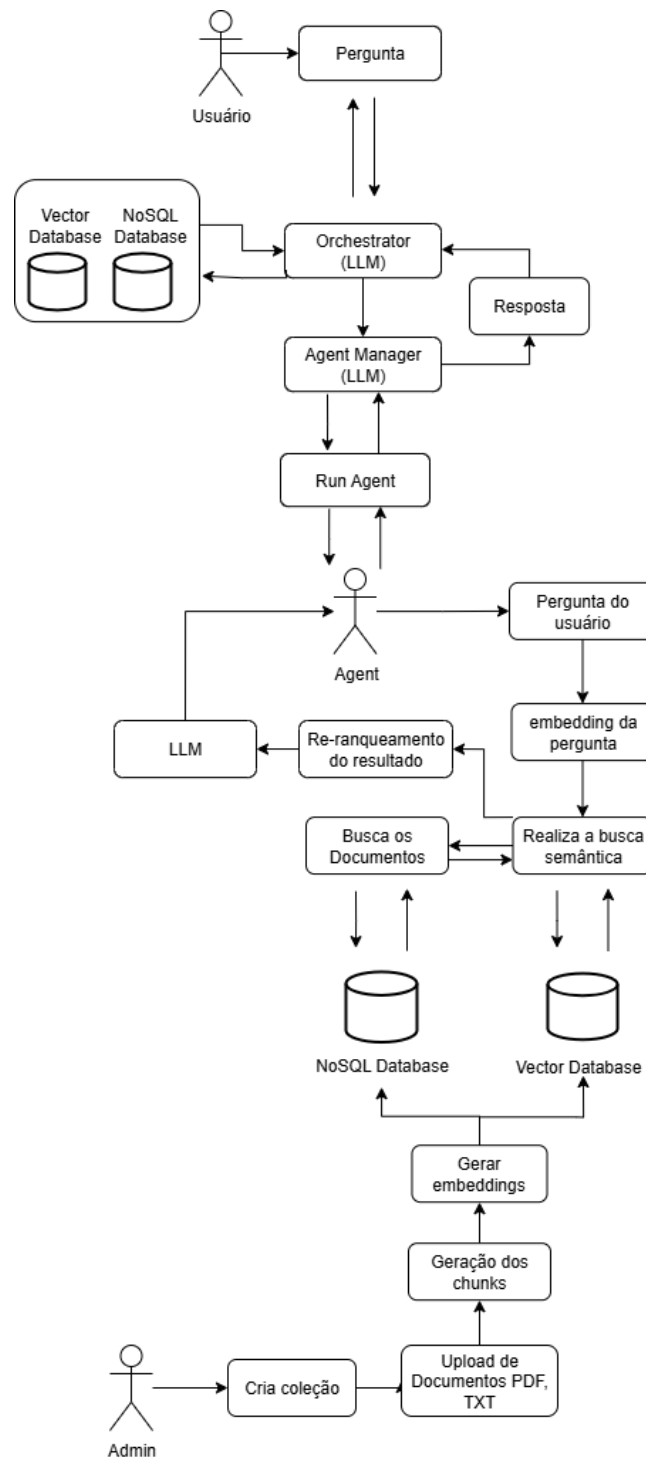


Figura 2 – Fluxo de Funcionamento do Sistema
Fonte: O autor, 2025

5.3 Visão Geral de Componentes

O objetivo do diagrama de componentes é representar os elementos que compõem o *software* e a forma como se relacionam entre si, como mostrado na Figura 3. Ele ilustra os artefatos que constituem esses componentes, como arquivos de código-fonte, bibliotecas, e estruturas de banco de dados (MARTINS; DINIZ; SILVA, 2017).

1. **ReactJs e CSS:** É uma sugestão para o desenvolvimento do *frontend* para administrar o sistema, sendo que o primeiro é um *framework Javascript* popular para a construção de *interfaces* de usuários interativas e responsivas. O segundo servirá para estilizar a página *web*. Juntos eles formaram a camada da aplicação.
2. **Rest API:** É a representação do fornecimento das *APIs* para acessar um serviço na *web*.
3. **NodeJs:** É uma sugestão para o desenvolvimento de um ambiente de execução *Javascript server-side*. Servirá para a criação dos serviços que será chamado pelo *frontend* como, por exemplo, realizar *login*, cadastrar usuário, cadastrar ferramentas e cadastrar *feedbacks*.
4. **MongoDB:** Sistema de gerenciamento de banco de dados não relacional escolhido para armazenar os dados dos agentes e também o binário dos documentos.
5. **Python:** É uma linguagem de programação de alto nível e será utilizado para o desenvolvimento do *backend* do Agente de *IA* e também o *backend* do *RAG*. Nele será utilizado bibliotecas como *Fast API*, *Gemini*, entre outros. Além de ter comunicações com as bases *MongoDB* e *Qdrant*.
6. **Gemini:** É a *IA* Generativa que usaremos para interagir com o usuário.
7. **Qdrant:** É um mecanismo de busca por similaridade vetorial e banco de dados. Será essa base de dados que retornará os resultados das buscas, de acordo com a pergunta do usuário.

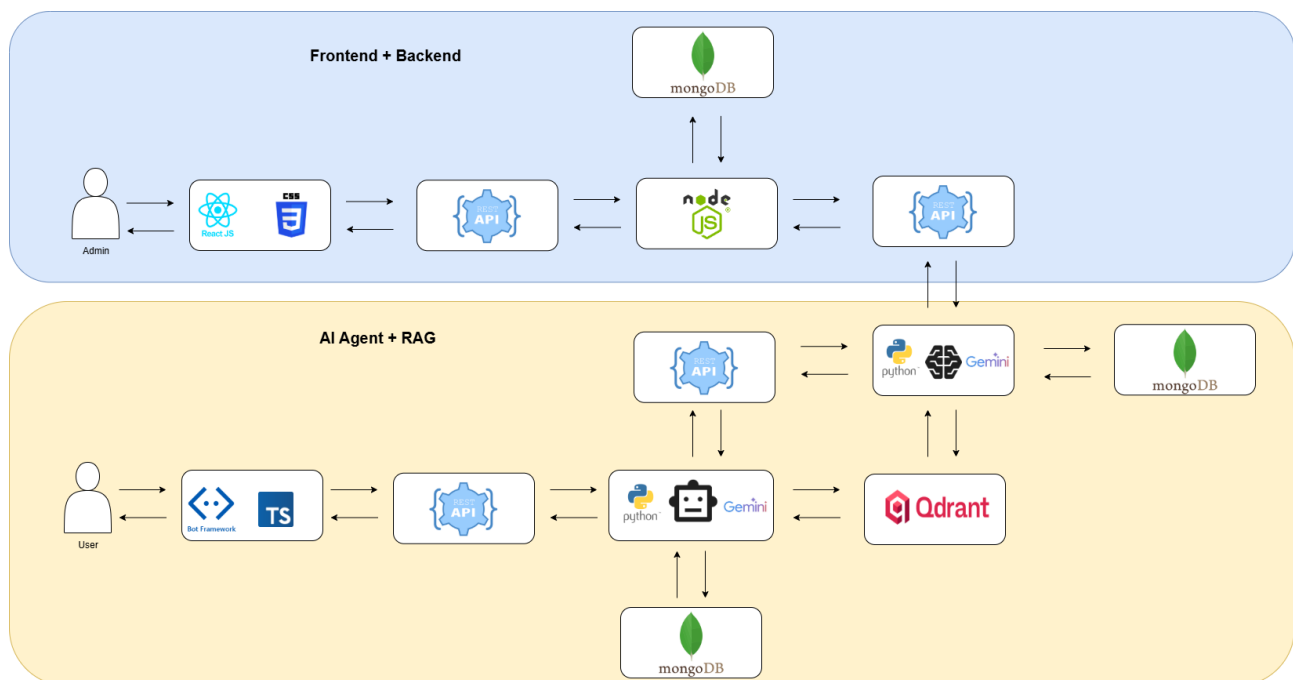


Figura 3 – Diagrama de Componentes

Fonte: O autor, 2025

5.4 Desenvolvimento das APIs do RAG

Esta Seção apresenta a construção das APIs responsáveis pela implementação do RAG. São descritos os serviços criados para indexação, recuperação e disponibilização de informações, garantindo que os dados sejam processados e consultados durante a interação com o agente.

5.4.1 Criação da coleção

Para viabilizar o *upload* de documentos, é necessário que exista previamente uma coleção criada no banco de dados vetorial *Qdrant*. Para isso, foi desenvolvida uma API específica para a criação dessas coleções, conforme a Figura 4.

```
router = APIRouter(  
    prefix="/collection",  
    tags=["Coleções"],  
    dependencies=[Depends(bearer_token_validation)]  
)  
  
@router.post("/create")  
def create_collection(collection_name: str):  
    CollectionValidation.collection_name_not_empty(collection_name)  
    CollectionValidation.collection_does_not_exist(collection_name)  
    return create_collection_controller(collection_name, vector_size=384)
```

Figura 4 – Rota para criar a coleção
Fonte: Postman, 2025

Para fins de demonstração, uma coleção com o nome “demonstracao-teste” foi criada (Figura 5). Essa coleção serviu como base para a demonstração de indexação, recuperação e geração da informação.

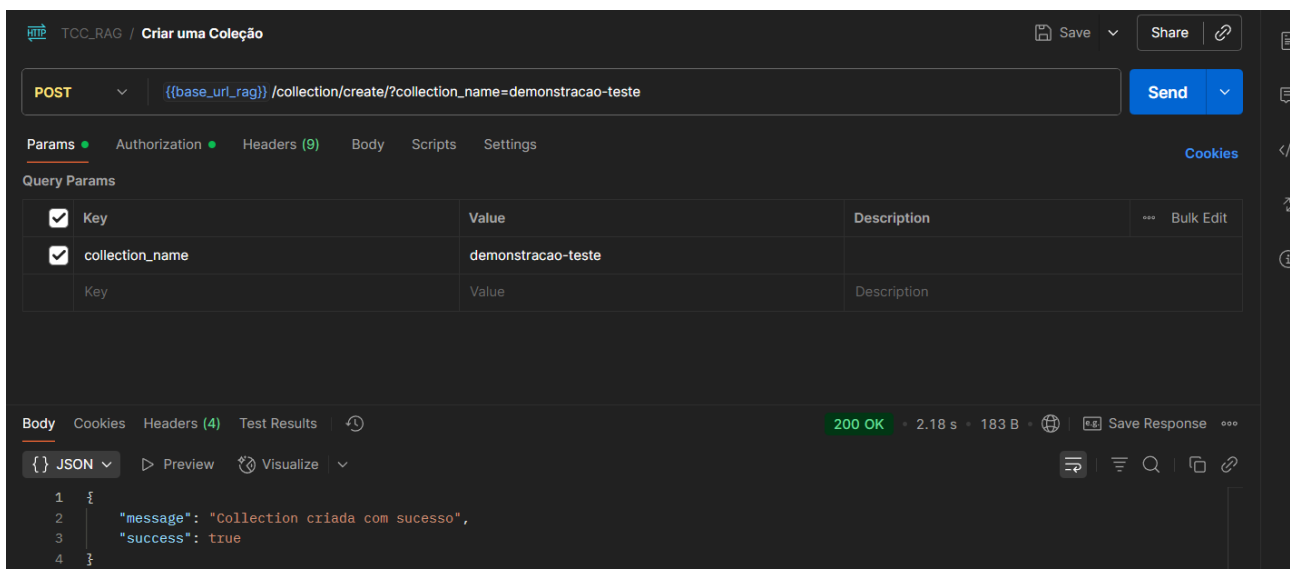


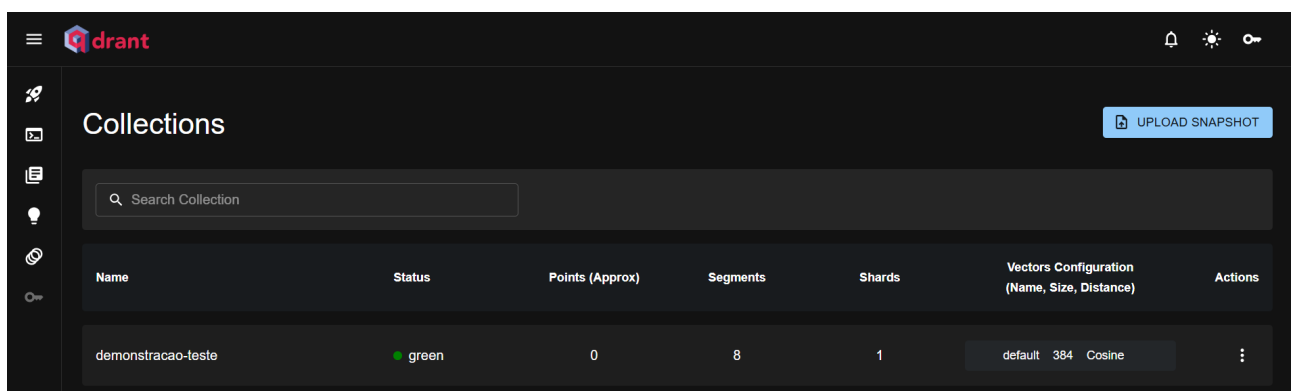
Figura 5 – Criando a coleção demonstracao-teste
Fonte: Postman, 2025

Antes de efetivar a criação, o sistema valida se o nome da coleção foi informado e se ela ainda não está registrada. Após essas verificações, é feita a chamada da função responsável por criar a coleção no *Qdrant*, conforme a Figura 6.

```
def create_collection(self, collection_name: str, vector_size: int) → bool:
    try:
        self.client.create_collection(
            collection_name=collection_name,
            vectors_config=VectorParams(size=vector_size, distance=Distance.COSINE)
        )
        return True
    except Exception as e:
        print(f"[ERRO] Falha ao criar coleção: {e}")
        return False
```

Figura 6 – Função para criar a coleção
Fonte: O autor, 2025

Considerando o modelo de *embedding Sentence Transformer* que está sendo utilizado, foi definido o tamanho do vetor como 384 dimensões (o texto será representado por 384 valores numéricos), com a métrica de similaridade configurada para cosseno (*cosine similarity*).



The screenshot shows the Qdrant web interface. At the top, there's a navigation bar with the Qdrant logo and a search bar. Below that, the 'Collections' section is visible. A table lists the collections, with one entry: 'demonstracao-teste'. The table columns are: Name, Status, Points (Approx), Segments, Shards, Vectors Configuration (Name, Size, Distance), and Actions. The 'demonstracao-teste' row shows a status of 'green', 0 points, 8 segments, 1 shard, and a vector configuration of 'default 384 Cosine'.

Name	Status	Points (Approx)	Segments	Shards	Vectors Configuration (Name, Size, Distance)	Actions
demonstracao-teste	green	0	8	1	default 384 Cosine	

Figura 7 – Coleção criada no Qdrant. Podemos perceber o tamanho do vetor definido e também a utilização da distância cosseno
Fonte: Qdrant, 2025

5.4.2 Upload de Documentos (Indexação)

Com a coleção criada, foi possível realizar o *upload* do primeiro documento. Para isso, o usuário informa para qual coleção os dados vetorizados serão enviados, juntamente com o próprio arquivo.

Caso o procedimento seja uma atualização de documento, o usuário pode fornecer o identificador do documento existente (*document_id_to_update*) para substituição. Se o objetivo for adicionar um aditivo a um documento já existente, basta informar o identificador do documento de referência (*parente_document_id*). Caso um documento com o mesmo conteúdo já exista na coleção, o sistema retorna uma mensagem de erro. Na Figura 8, é possível observar a rota utilizada para o upload do documento.

```

router = APIRouter(
    prefix="/document",
    tags=["Documentos"],
    dependencies=[Depends(bearer_token_validation)]
)

@router.post("/upload")
async def upload_document(
    file: UploadFile = File(...),
    collection_name: str = Query(...),
    document_id_to_update: str = Query(None, description="ID do documento a ser ATUALIZADO (versionamento)."),
    parent_document_id: str = Query(None, description="ID do documento principal ao qual este novo se relaciona (ex: Contrato Pai).")
):
    contents = await file.read()
    hash_document = hash_file(contents)

    CollectionValidation.collection_name_not_empty(collection_name)
    CollectionValidation.collection_exists(collection_name)
    DdocumentValidation.document_extension(file.filename)
    DdocumentValidation.document_exists(hash_document, collection_name)

    if document_id_to_update:
        DdocumentValidation.document_id_exists(document_id_to_update)
    else:
        pass

    if parent_document_id:
        DdocumentValidation.document_id_exists(parent_document_id)

    # Chama o controller, passando o conteúdo do arquivo
    response = upload_document_controller(
        hash_document=hash_document,
        file_content=contents,
        filename=file.filename,
        collection_name=collection_name,
        document_id_to_update=document_id_to_update,
        parent_document_id=parent_document_id
    )

    return response

```

Figura 8 – Rota para upload do documento

Fonte: O autor, 2025

Durante o processo de indexação, o sistema validou se o nome da coleção foi informado e se existe. Também foi verificado a extensão do arquivo anexado, aceitando, no momento, apenas *.txt*, *.pdf* e *.docx*. Após todas as validações, foi iniciado o fluxo de *upload*.

```

try:
    # Processa o novo arquivo para o Qdrant
    chunks = chunk_service.chunk_document(str(temp_filepath), doc_id=hash_document, filename=filename)
    if isinstance(chunks, dict):
        return chunks

    vectors = embedder_service.embed_chunks(chunks)
    qdrant_service.index_chunks(chunks, collection_name=collection_name, vectors=vectors)

```

Figura 9 – Função para processar o arquivo

Fonte: O autor, 2025

Para continuidade dos testes, foi importado um arquivo contendo um trecho do Projeto Pedagógico de Sistemas de Informação do ano de 2022, com o objetivo de observar como o *RAG* está sendo gerado, utilizando o seguinte trecho de exemplo:

“A computação está entre as áreas do conhecimento em grande e rápida expansão nos últimos anos. Com o aumento do volume de dados, as organizações necessitam do uso adequado de instrumentos para a coleta, armazenamento, processamento e transmissão. A disponibilidade da informação certa e no momento certo torna-se

requisito fundamental para a tomada de decisão e, conseqüentemente, manutenção da qualidade, produtividade e competitividade organizacional.”

Primeiramente, todo o conteúdo do documento é extraído página por página. Em seguida, para cada página, é realizado a detecção do idioma e a tokenização do texto conforme o idioma identificado, utilizando a biblioteca de *PLN NLTK*, o que garante uma segmentação mais precisa das sentenças.

Na Figura 10 mostra o usuário realizando o *upload* do documento para a coleção criada anteriormente.

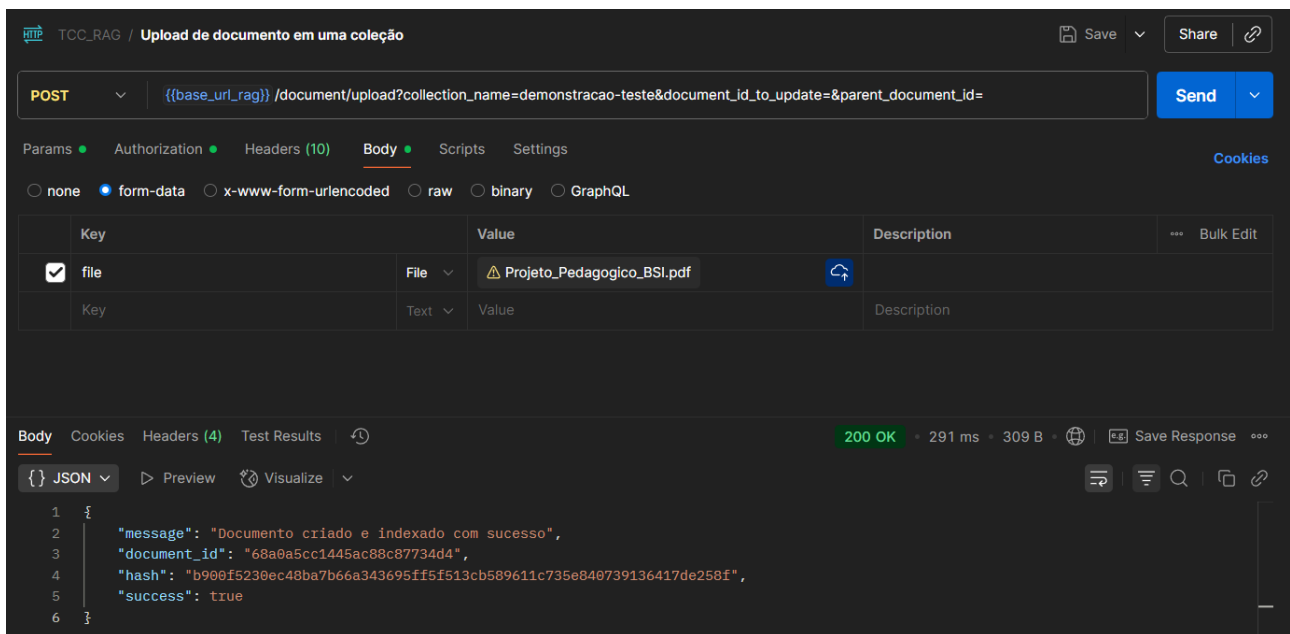


Figura 10 – Postman - Upload de documento
Fonte: Postman, 2025

Após o *upload*, a Figura 11 mostra o processo da biblioteca *NLTK* realizando a segmentação por sentenças, resultando em uma lista contendo três sentenças.

```
Tokenização utilizando NLTK para o idioma: portuguese
['A computação está entre as áreas do conhecimento em grande e rápida expansão nos últimos anos.', 'Com o aumento do volume de dados, as organizações necessitam do uso adequado de instrumentos para a coleta, armazenamento, processamento e transmissão.', 'A disponibilidade da informação certa e no momento certo torna-se requisito fundamental para a tomada de decisão e, conseqüentemente, manutenção da qualidade, produtividade e competitividade organizacional.']
```

Figura 11 – Log - Tokenização por sentença
Fonte: O autor, 2025

Como o processo de *PLN* é realizado por página, ainda será realizado o processo de *tokenização*, em que pegamos cada sentença e quebramos por palavras, conforme a figura abaixo:

```
TOKENS:
1 - ['A', 'computação', 'está', 'entre', 'as', 'áreas', 'do', 'conhecimento', 'em', 'grande', 'e', 'rápida', 'expansão', 'nos', 'últimos', 'anos.']
2 - ['Com', 'o', 'aumento', 'do', 'volume', 'de', 'dados', 'as', 'organizações', 'necessitam', 'do', 'uso', 'adequado', 'de', 'instrumentos', 'para', 'a', 'coleta', 'armazenamento', 'processamento', 'e', 'transmissão.']
3 - ['A', 'disponibilidade', 'da', 'informação', 'certa', 'e', 'no', 'momento', 'certo', 'torna-se', 'requisito', 'fundamental', 'para', 'a', 'tomada', 'de', 'decisão', 'e', 'conseqüentemente', 'manutenção', 'da', 'qualidade', 'produtividade', 'e', 'competitividade', 'organizacional.']
```

Figura 12 – Log - Tokenização
Fonte: O autor, 2025

A realização do processo de *tokenização* é importante para que seja possível criar o *chunk* do documento. O tamanho do *chunk* exerce papel fundamental na eficácia da recuperação em diferentes conjuntos de dados, pois influencia diretamente a capacidade de capturar trechos relevantes. Estudos indicam que *chunks* menores (64 a 128 tokens) tendem a apresentar melhor desempenho em bases que exigem respostas curtas e factuais, enquanto *chunks* maiores (512 a 1024 tokens) são mais adequados para cenários que demandam respostas descritivas ou técnicas (BHAT et al., 2025).

Com base nesse equilíbrio, foi adotado o tamanho de 300 *tokens* para os *chunks*, buscando combinar boa cobertura de contexto com eficiência computacional. Quando uma sentença ultrapassa esse limite, é criado um novo *chunk*, aplicando um *overlap* de 60 *tokens* (equivalente a aproximadamente 20% do tamanho do *chunk*) (MASOOD, 2025) para preservar a continuidade semântica e evitar perdas de informação relevantes na transição entre blocos.

```
CHUNKS:
[{'text': 'A computação está entre as áreas do conhecimento em grande e rápida expansão nos últimos anos. Com o aumento do volume de dados, as organizações necessitam do uso adequado de instrumentos para a coleta, armazenamento, processamento e transmissão. A disponibilidade da informação certa e no momento certo torna-se requisito fundamental para a tomada de decisão e, consequentemente, manutenção da qualidade, produtividade e competitividade organizacional.', 'doc_id': 'b900f5230ec48ba7b66a343695ff5f513cb589611c735e840739136417de258f', 'filename': 'Projeto_Pedagogico_BSI.pdf', 'chunk_id': 0, 'page': 1}]
```

Figura 13 – Log - *Chunk* gerado

Fonte: O autor, 2025

Com a lista de *chunks* gerada, é realizada a vetorização por meio do modelo *sentence-transformers/all-MiniLM-L6-v2*, resultando na sua própria *tokenização* e geração dos *embeddings* correspondentes (Figura 15). Por fim, os vetores são indexados na coleção informada pelo usuário no início do processo e o documento é armazenado no banco de dados *MongoDB* (Figura 14).

```
if document_id_to_update:
    # 1. Obter metadados da versão antiga
    old_metadata = metadata_service.get_document_by_id(document_id_to_update)
    if not old_metadata:
        return {"message": f"Documento com ID {document_id_to_update} não encontrado para atualização.", "success": False}
    old_hash = old_metadata['active_version_hash']
    old_gridfs_file_id = old_metadata.get('gridfs_file_id')

    qdrant_service.delete_by_doc_id(old_hash, collection_name)

    new_gridfs_file_id = metadata_service.save_file(file_content, filename, hash_document)

    metadata_service.update_document_version(document_id_to_update, hash_document, filename, new_gridfs_file_id)

    if old_gridfs_file_id:
        metadata_service.delete_file_from_gridfs(old_gridfs_file_id)

    return {"message": "Documento atualizado com sucesso", "document_id": document_id_to_update, "new_version_hash": hash_document, "success": True}

else:
    new_doc_id = metadata_service.create_document_record(
        filename, collection_name, hash_document, file_content, parent_document_id
    )
    return {"message": "Documento criado e indexado com sucesso", "document_id": new_doc_id, "hash": hash_document, "success": True}

finally:
    os.remove(temp_filepath)
```

Figura 14 – Função para armazenar o arquivo no MongoDB

Fonte: O autor, 2025

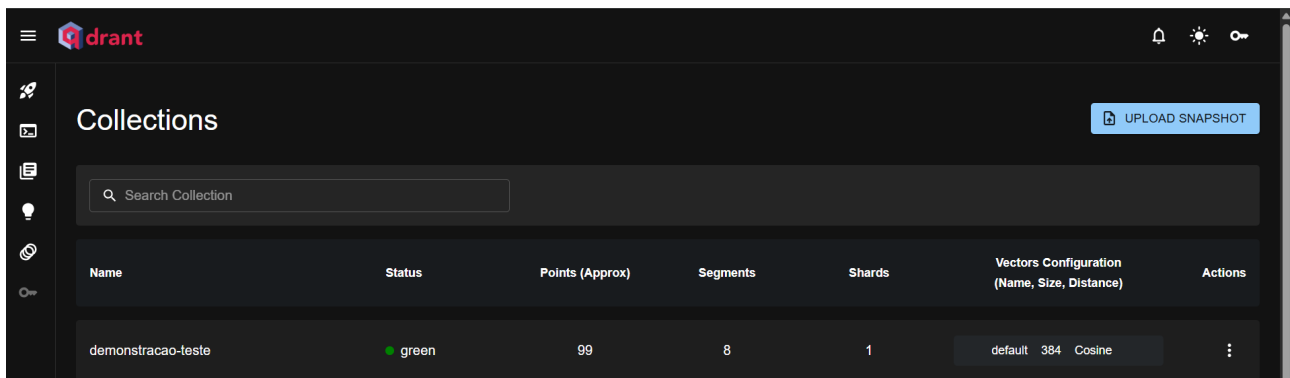
VETORES:
[[0.06259413063526154, -0.02255699783563614, -0.12985187768936157, -0.09321286529302597, -0.10923216491937637, -0.040094390511512756, 0.024078497663140297, 0.1046989411156464, 0.03720048442482948, 0.05571248382329941, 0.055073630064725876, -0.03629468381404877, -0.006025488488376141, 0.019294509664177895, -0.013935250230133533, -0.028242114931344986, 0.027609191834926605, 0.008661938831210136, -0.04352985695004463, -0.01729174703359604, 0.12469494342803955, -0.018220318481326103, -0.03516224026679993, -0.004494767636060715, -0.04335874319076538, 0.04289370775222778, -0.0514222407520485, -0.04326942563056946, -0.017505161464214325, -0.05917402729392052, -0.02333875186741352, 0.07572736591100693, 0.17393498122692108, 0.00017945432045962662, -0.06548146903514862, 0.011756189167499542, 0.0697307139635086, -0.059634577482938766, -0.023682503029704094, 0.00593491829931736, -0.0345083512365818, 0.00021220372582320124, 0.0001428300456609577, -0.08345251530408859, -0.01983562298119068, 0.0025067811366170645, -0.005827861838042736, -0.021506603807210922, -0.037215206772089005, -0.019748924300074577, -0.04200627654790878, -0.04186345264315605, -0.011760947294533253, 0.0395963229238987, -0.03616088628768921, -0.020103085786104202, 0.06738733500242233, 0.008400160819292068, -0.03444131463766098, 0.036934785544872284, -0.026907285675406456, 0.04335648566484451, -0.013577884063124657, -0.0029347098898142576, 0.07317826896905899, 0.0052827936597168446, -0.03755088150501251, 0.012374039739370346, -0.05579433962702751, -0.04155215248465538, 0.05461443215608597, -0.06314542144536972, -0.005987612530589104, 0.028268927708268166, -0.0114480636060238, -0.007795555051416159, -0.10480473935604095, 0.0474456362426281, 0.0055568525567650795, -0.08930031210184097, 0.06068471074104309, 0.007754285354167223, -0.00657104313373566, 0.0190075833584206, -0.03801926225428813, 0.057226436042785645, -0.018246548250317574, 0.021093768998980522, -0.01785011135041714, -0.020231515169143577, -0.000445782789020101, 0.019495198503136635, -0.04992763698101044, -0.0427714355859825, 0.003865940496325403, -0.01950126513839768, 0.01893804222345352, -0.0056362206935893, 0.07416447970932, -0.05004358705285225, 0.03165307768500328, 0.006176293827593327, 0.05688260495662689, -0.027802807498574257, -0.06913589686155319, -0.0471329428255558, -0.042008087038993835, 0.009098886512219906, 0.0451444648205234, 0.09506144374608994, -0.0625780045986176, 0.03180892917547226, -0.1120288074016571, -0.01777857355773449, -0.02222779020667076, -0.01811937987804113, -0.011235158890485764, -0.01621309295296669, -0.002009162988066673, 0.04395870119333267, 0.013632281683385372, -0.0521901436150074, 0.0033194972202181816, -0.03900054469704628, 0.07505474984645844, -0.04167262092232704, 0.03688906505703926, 8.24134828194484e-33, -0.05130387961864471, -0.02832732486591339, -0.022866658866405487, 0.0591200, 0.01416161470151848, -0.10896053910255432, -0.024303190410137177, -0.006035580765455961, 0.01827954687178135, -0.003112210426479578, 0.03691988065838814, -0.013489908538758755, -0.010627388954162598, -0.05741069093346596, -0.005815537180751562, -0.0028988525737076998, 0.023876387625932693, 0.03203136032886505, 0.04162643104791641, -0.010091820731759071, 0.12184412032365799, 0.043258003890514374, -0.018155215308070183, -0.006390843074768782, 0.041131872683763504, -0.05511961877346039, 0.06015007942914963, 0.06546784937381744, -0.05654721334576687, 0.012372899800539017, -0.02766214683651924, -0.028996413573622704, -0.05236421898007393, 0.035976916551589966, -0.00427027326072228, 0.004679915960878134, -0.022429760

Figura 15 – Log - Representação semânticas (embeddings)

Fonte: O autor, 2025

5.4.3 Recuperação e Geração (*Retrieval Generation*)

Com a criação da coleção e a indexação do primeiro documento, o sistema está preparado para realizar a etapa de recuperação. Para demonstrar o processo, além do documento de teste, foi indexado também o documento “Regimento Geral da UFU”, ampliando a base de conhecimento e permitindo avaliar o funcionamento da recuperação da informação. A Figura 16 ilustra o estado do banco de dados vetorial após a indexação, onde a coluna **Points (Approx)** indica a quantidade total de vetores (*chunks*) armazenados.



Name	Status	Points (Approx)	Segments	Shards	Vectors Configuration (Name, Size, Distance)	Actions
demonstracao-teste	green	99	8	1	default 384 Cosine	

Figura 16 – Qdrant: Quantidade de vetores cadastrados

Fonte: Qdrant, 2025

Para demonstração, foi formulado uma pergunta para observar a recuperação da informação dos documentos que estão na coleção:

“Como a rápida expansão da computação e o aumento do volume de dados impactam a necessidade de ferramentas adequadas para a coleta, processamento e transmissão de informações nas organizações?”

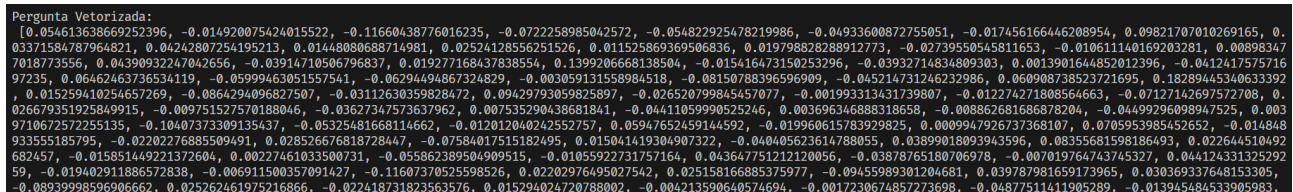
O processo de recuperação é iniciado quando o usuário submete uma consulta. O trecho de código a seguir apresenta a função responsável por receber a requisição, no qual o parâmetro obrigatório é a variável *query*. Os parâmetros *collections* e *limit_context* são opcionais e permite, respectivamente, direcionar a busca para coleções específicas e aplicar uma estratégia de otimização de contexto.

```

1      router = APIRouter(
2          prefix="/ask",
3          tags=["Perguntas"],
4          dependencies=[Depends(bearer_token_validation)]
5      )
6
7      @router.get("/")
8      async def ask(query: str,
9          collections: list[str] | None = Query(None),
10         limit_context: bool = Query(False)):
11         RetrieverValidation.query(query)
12         retrieved_chunks = retriever(query, collections, limit_context)
13         return retrieved_chunks

```

Quando o usuário realiza a pergunta, a primeira etapa consiste na geração do *embedding*, isto é, a transformação da pergunta em um vetor numérico utilizando o mesmo modelo empregado na indexação (Figura 17).



Pergunta Vetorizada:
[0.054613638669252396, -0.014920075424015522, -0.11660438776016235, -0.0722258985042572, -0.054822925478219986, -0.04933600872755051, -0.017456166446208954, 0.09821707010269165, 0.03371584787964821, 0.04242807254195213, 0.01448080688714981, 0.02524128556251526, 0.011525869369506836, 0.019798828288912773, -0.02739550545811653, -0.010611140169203281, 0.008983477018773556, 0.04390932247042656, -0.03914710506796837, 0.019277168437838554, 0.1399206668138504, -0.015416473150253296, -0.03932714834809303, 0.0013901644852012396, -0.041241757571697235, 0.06462463736534119, -0.05999463051557541, -0.06294494867324829, -0.003059131558984518, -0.08150788396596909, -0.045214731246232986, 0.060908738523721695, 0.18289445340633392, 0.015259410254657269, -0.0864294096827507, -0.03112630359828472, 0.09429793059825897, -0.026520799845457077, -0.001993313431739807, -0.012274271808564663, -0.07127142697572708, 0.026679351925849915, -0.009751527570188046, -0.03627347573637962, 0.007535290438681841, -0.04411059990525246, 0.003696346888318658, -0.008862681686878204, -0.04499296098947525, 0.0039710672572255135, -0.10407373309135437, -0.05325481668114662, -0.012012040242552757, 0.05947652459144592, -0.019960615783929825, 0.0009947926737368107, -0.0705953985452652, -0.014848933555185795, -0.02202276885509491, 0.028526676818728447, -0.07584017515182495, 0.015041419304907322, -0.040405623614788055, 0.03899018093943596, 0.08355681598186493, 0.022644510492682457, -0.015851449221372604, 0.00227461033500731, -0.055862389504909515, -0.01055922731757164, 0.043647751212120056, -0.03878765180706978, -0.007019764743745327, 0.04412433132529259, -0.019402911886572838, -0.006911500357091427, -0.11607370525598526, 0.02202976495027542, 0.025158166885375977, -0.09455989301204681, 0.039787981659173965, 0.030369337648153305, -0.0893998596906662, 0.025262461975216866, -0.022418731823563576, 0.015294024720788002, -0.004213590640574694, -0.0017230674857273698, -0.04877511411905289, -0.013945484533905983,

Figura 17 – Log - Embedding da Pergunta

Fonte: O autor, 2025

Com a pergunta vetorizada, o sistema verifica se o usuário especificou a coleção no qual a consulta deve ser realizada. Caso não tenha a indicação, a busca é feita em uma lista de coleções previamente definidas pelo próprio usuário, sendo que cada coleção possui uma breve descrição associada. Abaixo, mostramos um exemplo de trecho do código dessas descrições:

```

1      DESCRICOES = {
2          "normas-estagio": [
3              "Informações sobre regras e prazos do estágio curricular",
4              "Orientações para alunos sobre o processo de estágio
5              ↳ supervisionado",
6              "Regulamento do estágio, carga horária, prazos de entrega e
7              ↳ critérios de aprovação",
8              "esclarecimentos de dúvidas relacionadas ao estágio",
9              "A Lei do Estágio (Lei 11.788 de 25 setembro de 2008) estabelece
10             ↳ definições e regras sobre qualquer tipo de estágio",
11             "Normas, leis e prazos"

```

```

9         ],
10        "micologia": [
11            "Apostila de micologia com conteúdo sobre os filos de fungos",
12            "Fungos, características, estruturas, reprodução e importância
↪ ecológica",
13            "Informações sobre fungos de importância médica e ambiental",
14            "Classificação e morfologia dos fungos em diferentes filos"
15        ],
16        "ihc": [
17            "Apostila de IHC (Interação Humano-Computador)",
18            "Conceitos básicos de IHC, usabilidade e design de interfaces",
19            "Princípios de interação humano-computador e avaliação de
↪ usabilidade",
20            "Técnicas de design centrado no usuário e prototipagem"
21        ]
22    }

```

A partir dessas descrições, é realizado o *embedding* de cada coleção utilizando a mesma biblioteca para indexação da pergunta e documentos. Em seguida, calcula-se a similaridade do cosseno entre a pergunta e as descrições, retornando a lista de coleções com o valor de similaridade superior a 0.55. Esse limiar foi definido empiricamente, com base em testes realizados durante o desenvolvimento.

```

1    import numpy as np
2
3    @staticmethod
4    def cosine_similarity(a, b):
5        a = np.array(a)
6        b = np.array(b)
7        return np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b))

```

Para fins de demonstração, foi especificada a coleção “*demonstracao-teste*” e aplicado o limite de contexto.

Como mencionado anteriormente, a limitação do contexto é fundamental para aumentar a velocidade de processamento dos *chunks* e, conseqüentemente proporcionar respostas mais rápidas. Quando o contexto não é restringido, ao identificar um *chunk* de um documento que possui similaridade com a pergunta, a busca precisa considerar todos os *chunks* do documento para o *re-ranqueamento*, o que demanda tempo e processamento. Ao aplicar a limitação de contexto, implementamos uma lógica que considera apenas as cinco páginas anteriores e posteriores ao *chunk* identificado. Dessa forma, o escopo para análise é reduzido, diminuindo o tempo de processamento necessário para o *re-ranqueamento*.

Com a coleção definida, o próximo passo consiste na execução da busca semântica na base de conhecimento. Para isso, utilizamos os parâmetros a pergunta do usuário vetorizada, a quantidade máxima de *chunks* a serem retornados, o limiar de similaridade e a lista de coleções.

A busca vetorial retorna um conjunto de *chunks* que apresentam maior similaridade com a consulta do usuário. Entretanto, esses trechos isolados podem conter apenas parte da informação necessária para a construção de uma resposta completa.

Para mitigar essa limitação, é realizado um processo de expansão de contexto. Nessa etapa, os documentos de origem são identificados a partir dos *chunks* recuperados na busca inicial. Em seguida, caso estejamos utilizando uma lista de coleções, as páginas correspondentes são verificadas e os documentos são buscados no banco de dados *MongoDB*. Esse procedimento permite identificar a coleção a que cada documento pertence, uma vez que o *Qdrant* não retorna essa informação e os *chunks* não são organizados por padrão. Caso estivéssimos trabalhando com apenas um item da lista de coleções, não haveria a consulta no banco de dados.

Com o dicionário estruturado, contendo o *hash* do documento, as páginas e o nome da coleção, podemos utilizar o parâmetro *limit_context*, informado no início da requisição, para limitar a janela que será realizada a expansão de contexto. Nessa etapa, efetuamos uma nova busca no *Qdrant* para recuperar os *chunks* adicionais, considerando a variável *CONTEXT_WINDOW_SIZE*, definida no momento com o valor 5. Dessa forma, são obtidos os *chunks* que correspondem às cinco páginas anteriores e posteriores ao *chunk* identificado na busca inicial.

```

1      initial_chunks = qdrant_service.search_question(vector_question,
2      ↪ MAXIMUM_CHUNK_TOP, relevant_collections, THRESHOLD)
3
4      if not initial_chunks:
5          return {"message": "Não foram encontrados documentos para a pergunta",
6          ↪ "success": False}
7      expanded_context_chunks = {}
8      if limit_context:
9          print(f"INFO: Usando estratégia de Janela de Contexto (+/-
10         ↪ {CONTEXT_WINDOW_SIZE} páginas).")
11
12         # Coleta as páginas e coleções dos chunks encontrados
13         relevant_pages_by_doc = defaultdict(lambda: {'pages': set(),
14         ↪ 'collection': ''})
15
16         if len(relevant_collections) == 1:
17             collection_name = relevant_collections[0]
18             for doc_id, chunks in initial_chunks.items():
19                 relevant_pages_by_doc[doc_id]['collection'] = collection_name

```



```

16         for chunk in chunks:
17             relevant_pages_by_doc[doc_id]['pages'].add(chunk['page'])
18
19     else:
20         for doc_id, chunks in initial_chunks.items():
21             for chunk in chunks:
22                 relevant_pages_by_doc[doc_id]['pages'].add(chunk['page'])
23
24     # Encontra a coleção de cada documento (necessário para a busca no
25     ↪ Qdrant)
26     initial_doc_hashes = list(initial_chunks.keys())
27     for collection_name in relevant_collections:
28         records =
29         ↪ metadata_service.get_documents_by_hashes(collection_name,
30         ↪ initial_doc_hashes)
31         for record in records:
32             ↪ relevant_pages_by_doc[record['active_version_hash']]['collection']
33             ↪ = collection_name
34
35     # Busca os chunks dentro da janela de contexto para cada documento
36     all_window_chunks = []
37     for doc_hash, data in relevant_pages_by_doc.items():
38         if not data['pages'] or not data['collection']:
39             continue
40
41         min_page = max(1, min(data['pages']) - CONTEXT_WINDOW_SIZE)
42         max_page = max(data['pages']) + CONTEXT_WINDOW_SIZE
43
44         chunks_from_window = qdrant_service.get_chunks_by_page_window(
45             collection_name=data['collection'],
46             doc_hash=doc_hash,
47             min_page=min_page,
48             max_page=max_page
49         )
50         all_window_chunks.extend(chunks_from_window)
51
52     # Agrupa o resultado final no formato esperado pelo reranker
53     temp_grouped = defaultdict(list)
54     for chunk in all_window_chunks:
55         temp_grouped[chunk['document_id']].append(chunk)
56     expanded_context_chunks = dict(temp_grouped)

```

Este conjunto expandido é então submetido a um processo de *re-ranqueamento*, cuja finalidade é reavaliar a relevância de cada *chunk* de forma mais precisa em relação à consulta original, utilizando para isso o modelo de *Cross-Encoder* para calcular um novo *score* de relevância para cada um.

Após a ordenação dos *chunks* com base nesse novo *score*, podemos aplicar regras de filtragem para construir o contexto final: Primeiramente, são descartados os *chunks* cujo *score* estejam abaixo de um limiar pré-definido (*THRESHOLD_RERANKER*) e garante um número máximo de *chunks* sejam selecionados (*RERANKER_MAXIMUM_CHUNK_TOP*).

```

1      def rerank(self, question: str, chunks_by_doc: dict) -> dict:
2          """
3          Executa o re-ranqueamento global em todos os chunks recuperados.
4          """
5          # Juntar todos os chunks de todos os documentos em uma única lista
6          all_chunks = []
7          for doc_id, chunks in chunks_by_doc.items():
8              all_chunks.extend(chunks)
9
10         if not all_chunks:
11             return {}
12
13         # Criar os pares [pergunta, texto] para o modelo
14         pairs = [[question, chunk["text"]] for chunk in all_chunks]
15
16         # Executar a predição uma vez para todos os pares
17         print("INFO: Iniciando o processo de re-ranqueamento...")
18         scores = self.model.predict(pairs, show_progress_bar=True)
19         print("INFO: Re-ranqueamento concluído.")
20
21         # Atribuir os scores e ordenar a lista global de chunks
22         for chunk, score in zip(all_chunks, scores):
23             chunk["rerank_score"] = float(score)
24
25         sorted_chunks = sorted(all_chunks, key=lambda x: x["rerank_score"],
26                                ↪ reverse=True)
27
28         # Filtrar a lista global pelo limiar de relevância
29         relevant_chunks = [chunk for chunk in sorted_chunks if
30                            ↪ chunk["rerank_score"] >= self.threshold]
31
32         # Limita o número de chunks ao máximo permitido
33         final_chunks = relevant_chunks[:self.max_chunks]
```



```

32
33     # Reagrupar os chunks finais por seu 'document_id' original
34     # Isso mantém o formato de saída esperado pelo serviço do LLM
35     reranked_result = defaultdict(list)
36     for chunk in final_chunks:
37         doc_id = chunk["document_id"]
38         reranked_result[doc_id].append(chunk)
39
40     return dict(reranked_result)

```

Por fim, o conjunto de *chunks* selecionados e *re-ranqueados* são fornecidos como contexto para a *LLM* que, com base nessa informação e na pergunta do usuário, gera a resposta final.

```

1     llm = AnswerLLM()
2
3     answer = llm.answer_llm(question, BACKEND_BASE_URL, reranked_result)

```

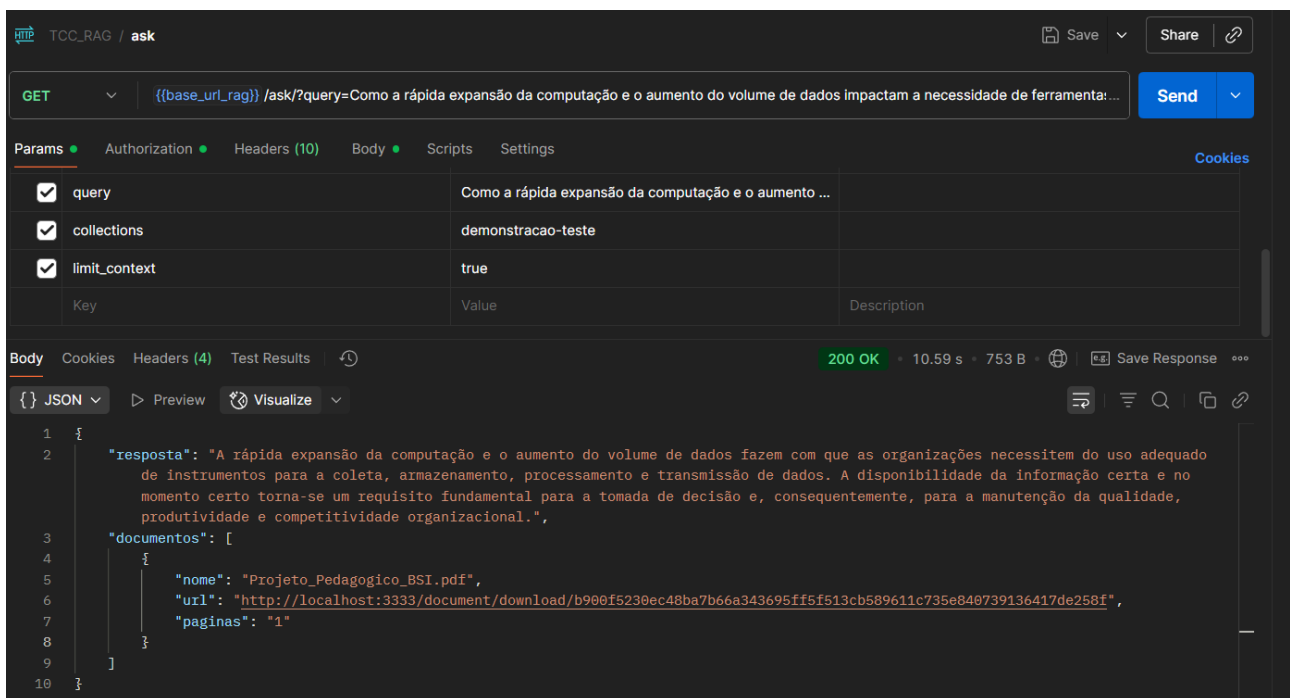


Figura 18 – Postman - Resultado da pergunta

Fonte: O autor, 2025

5.5 Desenvolvimento do Agente de IA

Nesta Seção são apresentadas as etapas de desenvolvimento do Agente de *IA*, incluindo a modelagem do banco de dados e a definição das responsabilidades das hierarquias criadas para

o *manager*, *agent* e *tool*. O objetivo é evidenciar como o agente foi estruturado para coordenar o uso das ferramentas, interagir com o *RAG* e gerar respostas adequadas aos usuários.

5.5.1 Criação dos Agentes

O primeiro passo para a criação dos agentes consistiu na definição do modelo de banco de dados, representado na Figura 19.

O modelo é composto pelas tabelas *Project*, *User*, *Manager*, *Agent* e *Tools*. A tabela *Project* tem como objetivo organizar e cadastrar os projetos, que podem estar vinculados a diferentes usuários. Dessa forma, um usuário pode possuir um ou mais projetos.

Cada projeto pode estar associado a um ou mais gerentes (*managers*). Enquanto um gerente está vinculado a apenas um projeto, um mesmo projeto pode contar com múltiplos gerentes.

Os gerentes, por sua vez, estão relacionados aos agentes. Um gerente pode administrar diversos agentes, embora cada agente pertença exclusivamente a um único gerente.

Por fim, cada agente pode dispor de múltiplas ferramentas (*tools*), sendo que cada ferramenta está vinculada a apenas um agente.

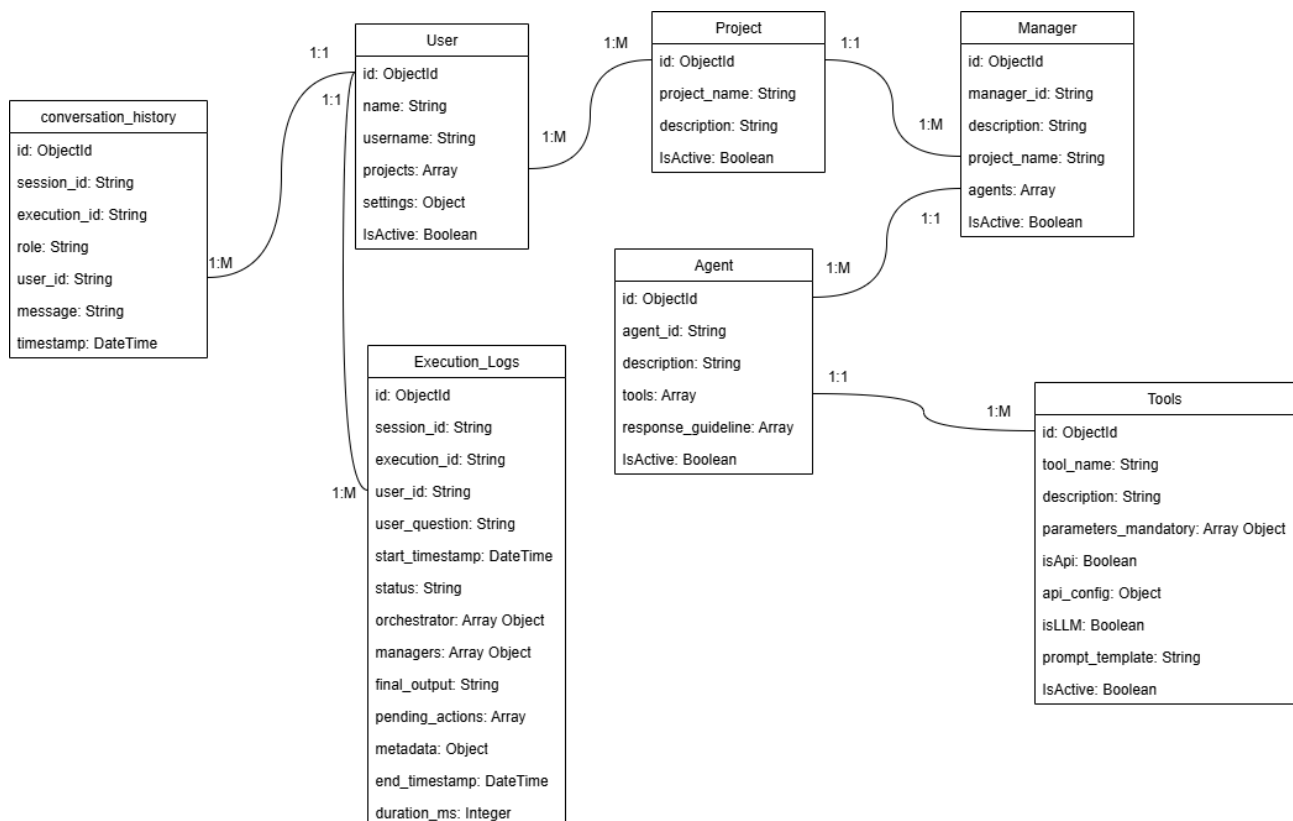


Figura 19 – Modelagem do Banco de Dados

Fonte: O autor, 2025

Além dessas tabelas, foram incluídas *Conversation_History* e *Execution_Logs*. A primeira é responsável por armazenar o histórico das conversas, funcionando como uma memória

de curto prazo. Já a segunda registra os *logs* de execução, desde a parte de orquestração até a finalização das tarefas, garantindo rastreabilidade no processo.

Para exemplificar o funcionamento, utilizamos a mesma pergunta empregada no processo de *RAG*, conforme ilustrado na Figura 20.

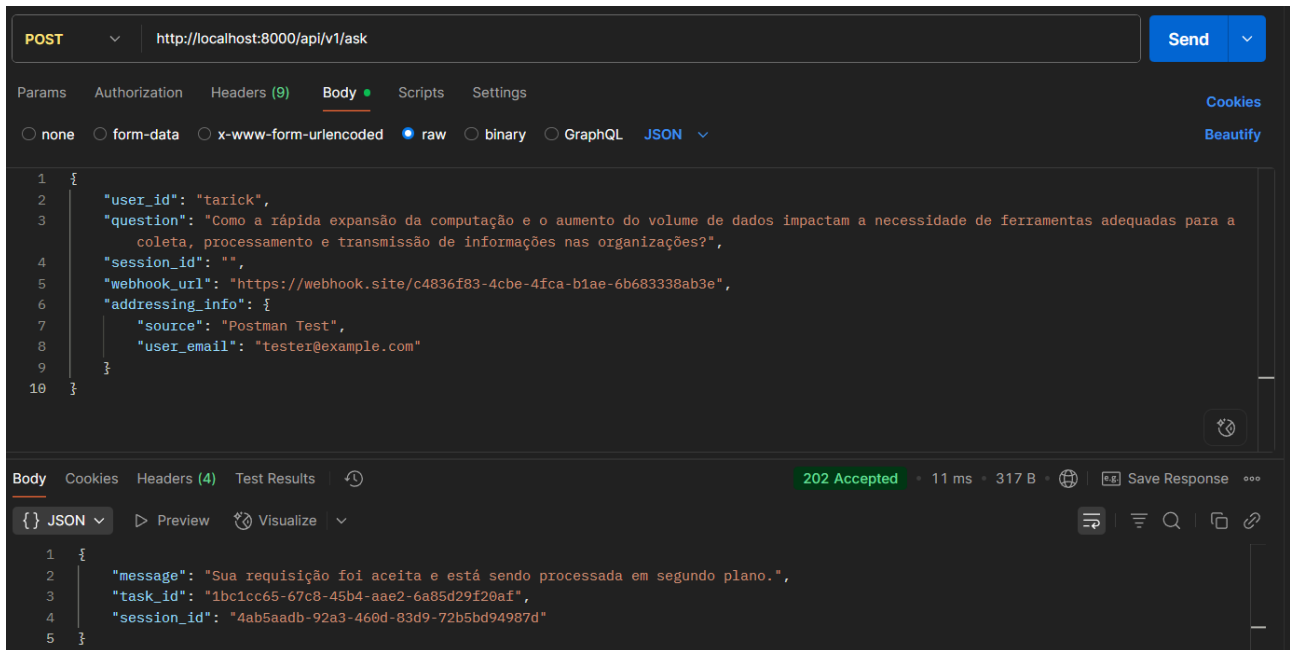


Figura 20 – Envio da pergunta para o orquestrador

Fonte: Postman, 2025

Quando a rota recebe uma pergunta do usuário, é gerado um identificador único para a tarefa. Caso um *session_id* já exista, o sistema dá continuidade à conversa, recuperando o histórico armazenado, de curto ou longo prazo, para servir como contexto da interação.

Em seguida, é construído o *payload* no formato esperado pelo *worker*, e a requisição é enfileirada para processamento assíncrono.

```

1 @router.post("/ask", status_code=status.HTTP_202_ACCEPTED)
2 def ask_question(request: UserRequest):
3     """
4     Recebe uma pergunta, a enfileira para processamento assíncrono
5     e retorna imediatamente.
6     """
7     try:
8         task_id = request.task_id or str(uuid.uuid4())
9         session_id = request.session_id or str(uuid.uuid4())
10
11         job_payload = {
12             "task_id": task_id,
13             "user_id": request.user_id,

```

```

14         "session_id": session_id,
15         "user_input": request.question,
16         "callback_details": {
17             "webhook_url": request.webhook_url,
18             "addressing_info": request.addressing_info
19         }
20     }
21
22     process_ai_request.send(job_payload)

```

Quando o *worker* consome o *payload*, ele inicia chamando o orquestrador. O papel do orquestrador é interpretar a pergunta do usuário e decidir se a resposta pode ser produzida diretamente ou se é necessário delegar a tarefa a um gerente.

Neste momento, o orquestrador já cria o *log* de execução, que será atualizado até o término do processo e posteriormente armazenado no banco de dados. Ele também identifica os gerentes e seus respectivos agentes vinculados ao projeto, registrando essas informações no contexto.

Com o contexto definido, é iniciado a chamada da *LLM*, que recebe tanto esse contexto quanto o histórico da conversa:

```

1  async def _cooperative_execution_flow(self, context: ExecutionContext) -> dict:
2      """Executa um fluxo de delegação cooperativo, decidindo um passo de cada
   ↪ vez."""
3      MAX_CYCLES = 5
4
5      chat_history = conversation_history.get_last_messages(
6          context.session_id,
7          num_messages=10)
8
9      for cycle in range(MAX_CYCLES):
10
11         next_action_plan = await asyncio.to_thread(
12             self.gemini.decide_next_manager_action, context, chat_history
13         )
14
15         thought = next_action_plan.get('thought', 'Nenhum pensamento registrado.')
16         decision = next_action_plan.get('decision')
17
18         self.logger.info(f"[ORCHESTRATOR_THOUGHT]: {thought}")
19         context.react_history.append(f"[ORCHESTRATOR_THOUGHT]: {thought}")

```

A cada ciclo, a *LLM* gera um pensamento (*thought*) e uma decisão (*decision*), conforme ilustrado na Figura 21.

```
[2025-08-24 11:21:50,401] [PID 42180] [Thread-4] [services.orchestration.orchestrator] [INFO] [ORCHESTRATOR_THOUGHT]: A pergunta do usuário é conceitual e ampla, abordando o impacto da expansão da computação e do volume de dados nas organizações. Nenhum dos managers disponíveis parece ser um especialista direto em temas de tecnologia da informação/negócios de forma genérica. No entanto, o 'MA0001' possui a ferramenta 'ConsultarBaseConhecimento'. Embora a descrição do 'MA0001' mencione 'dúvidas sobre estágio', a ferramenta 'ConsultarBaseConhecimento' em si pode ser capaz de acessar uma base de conhecimento mais ampla que inclua tópicos como o impacto da tecnologia e dados em organizações. Assumindo que a base de conhecimento pode conter informações relevantes, vou delegar a pergunta para o 'MA0001' usando a ferramenta 'ConsultarBaseConhecimento' para tentar obter uma resposta.

[2025-08-24 11:21:50,401] [PID 42180] [Thread-4] [services.orchestration.orchestrator] [INFO] Decisão: Delegar para o Manager 'MA0001' com a tarefa: 'Como a rápida expansão da computação e o aumento do volume de dados impactam a necessidade de ferramentas adequadas para a coleta, processamento e transmissão de informações nas organizações?'
```

Figura 21 – IA Agent - Log do Orquestrador
Fonte: O autor, 2025

Na Figura 21, o orquestrador optou por delegar a pergunta a um gerente. Caso contrário, ele poderia encerrar a execução informando que não havia ferramentas disponíveis para responder ou até mesmo, caso a pergunta realizada já estivesse sido respondida o próprio orquestrador responderia a pergunta.

Uma vez definido o gerente, é iniciado a busca pelas ferramentas disponíveis nos agentes associados. Essa etapa utiliza o conceito de *ReAct*, planejando todo o ciclo de execução necessário. A *LLM* retorna o pensamento (*thought*), a ação (*action*) ou a resposta final (*final_answer*). O pensamento indica o raciocínio utilizado pelo gerente para a escolha da ferramenta; a ação determina qual agente deve ser executado; e, caso a resposta esteja completa, é retornado o *final_answer*.

```
[2025-08-24 19:01:33,221] [PID 43256] [asyncio_0] [services.orchestration.manager_executor] [INFO] [THOUGHT]: O objetivo deste passo é entender o impacto da rápida expansão da computação e do aumento do volume de dados na necessidade de ferramentas para coleta, processamento e transmissão de informações nas organizações. A ferramenta mais adequada para buscar essa informação é 'ConsultarBaseConhecimento' do agente AG0001. A query deve ser formulada para cobrir os principais pontos da pergunta.
```

Figura 22 – Agente de IA - Log do Gerente
Fonte: O autor, 2025

```
[2025-08-24 19:01:33,221] [PID 43256] [asyncio_0] [services.orchestration.manager_executor] [INFO] [ACTION]: ConsultarBaseConhecimento(query="impacto da expansão da computação e volume de dados na necessidade de ferramentas para coleta processamento e transmissão de informações em organizações")
```

Figura 23 – Agente de IA - Log da Ação do Gerente
Fonte: O autor, 2025

Com o agente e a ferramenta definidos, a execução é realizada conforme ilustrado na Figura 24.

```
[2025-08-24 19:21:21,441] [PID 17528] [asyncio_0] [services.orchestration.agent_executor] [INFO] Agente 'AG0001' executando a ferramenta 'ConsultarBaseConhecimento'
```

Figura 24 – Agente de IA - Log do Agente
Fonte: O autor, 2025

Após a execução, o resultado é retornado ao gerente, que avalia se a resposta é final. Caso positivo, a resposta segue para o orquestrador, que decide se deve envolver outro gerente

ou consolidar a resposta. Quando consolidada, a resposta é armazenada no banco de dados juntamente com o *log* de execução, sendo finalmente enviada ao usuário (Figura 25).

```
{
  "task_id": "f8a457b3-c976-4dd8-9b75-eccd03364de9",
  "status": "completed",
  "addressing_info": {
    "source": "Postman Test",
    "user_email": "tester@example.com"
  },
  "final_output": "A rápida expansão da computação e o aumento expressivo no volume de dados têm um impacto fundamental nas organizações. Eles impulsionam a necessidade crítica de instrumentos e ferramentas adequadas para a coleta, armazenamento, processamento e transmissão dessas informações.\n\nIsso ocorre porque a disponibilidade da informação correta, no momento certo, torna-se um requisito indispensável para a tomada de decisões estratégicas. Consequentemente, a adequação dessas ferramentas impacta diretamente na capacidade da organização de manter sua qualidade, produtividade e competitividade no mercado."
}
```

Figura 25 – Agente de IA - Resultado Final da Execução

Fonte: O autor, 2025

5.5.2 Criação dos Prompts

Para o desenvolvimento dos agentes, foi utilizados três *prompts* sendo um para o orquestrador analisar a solicitação e delegar para os gerentes, o que chamamos de *CoT*. O *prompt* dos gerentes para analisar e montar um fluxo de execução dos agentes com as suas ferramentas, que chamamos de *ReAct Prompting* e, por fim, um *prompt* tradicional para consolidar a resposta final.

A seguir, são apresentados exemplos de dois tipos de *prompts*, sendo o primeiro referente à técnica *CoT* e o segundo ao *ReAct Prompting*.

Prompt *CoT*:

```
1 {
2   "thought": "Seu raciocínio aqui. Explique por que você está escolhendo este
   ↳ manager e qual informação espera obter.",
3   "decision": "call_manager",
4   "manager_id": "ID_DO_MANAGER_ESCOLHIDO",
5   "new_question": "A pergunta/instrução clara e auto-suficiente para este manager,
   ↳ possivelmente usando resultados de passos anteriores. Ex: 'Com base no CEP X,
   ↳ busque o endereço completo.'"
6 }
```

ReAct Prompting:

```
1 [THOUGHT]:
2 (Seu raciocínio detalhado aqui. Analise o Objetivo Deste Passo e o Rascunho para
   ↳ decidir a próxima ação. Seja explícito sobre qual agente e ferramenta usar e por
   ↳ quê.)
3
```

```
4 [ACTION]:  
5 NomeDaFerramenta(parametro="valor")
```

5.6 Desenvolvimento do *ChatBot*

Esta Seção apresenta o desenvolvimento do *chatbot*, que atua como intermediário na comunicação entre os usuários e o Agente de IA. Durante a fase de testes, foi utilizado o *Microsoft Bot Framework Emulator*, que permitiu simular interações reais e validar o funcionamento do sistema.

5.6.1 Criação do aplicativo de Chat para Microsoft Teams

O desenvolvimento do aplicativo foi realizado de forma simplificada, sem integração com o *Office 365*, devido a políticas de privacidade da universidade. Portanto, seguimos os mesmos processos que a ferramenta *Microsoft Toolkit* realizaria.

O servidor *backend* foi implementado no arquivo *index.ts*, responsável por expor a rota para recebimento de mensagens dos usuários:

```
1 expressApp.post("/api/messages", async (req, res) => {  
2   await adapter.process(req, res, async (context) => {  
3     await bot.run(context);  
4   });  
5 });
```

Além disso, foi criada uma rota *webhook* para receber a resposta do agente e enviá-la ao usuário, utilizando a referência da conversa:

```
1 expressApp.post("/api/webhook", async (req, res) => {  
2   console.log("Webhook recebido:", req.body);  
3  
4   const conversationReferenceFromApi = req.body.addressing_info;  
5   const answer = req.body.final_output;  
6  
7   if (!conversationReferenceFromApi || !answer) {  
8     console.error("Webhook recebido com dados incompletos.");  
9     return res.status(400).send("Payload incompleto.");  
10  }  
11  
12  const sessionId = conversationReferenceFromApi.conversation.id;  
13  const storedConversationReference = conversationReferences[sessionId];
```

```

14
15     if (!storedConversationReference) {
16         console.error(`Referência da conversa não encontrada para session_id:
            ↳ ${sessionId}`);
17         return res.status(404).send("Conversation not found");
18     }
19
20     await adapter.continueConversationAsync(
21         process.env.MicrosoftAppId,
22         storedConversationReference,
23         async (proactiveContext) => {
24             await proactiveContext.sendActivity(answer);
25         }
26     );
27
28     res.sendStatus(200);
29 });

```

Para simular a execução do programa, foi utilizado a plataforma de teste *Microsoft Bot Framework Emulator*, que permite reproduzir a interação como se o aplicativo estivesse instalado no *Microsoft Teams*. A Figura 26 ilustra a configuração da *API* para envio de mensagens pelo usuário.

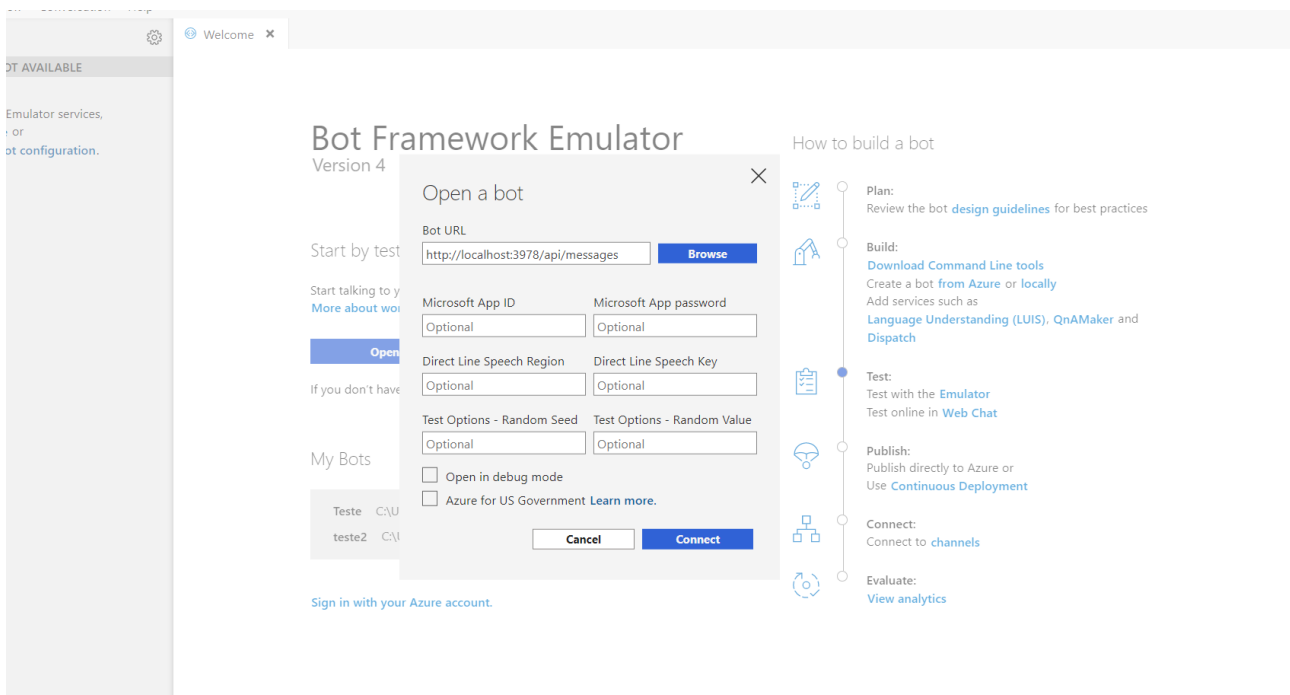


Figura 26 – Configuração do Bot Framework
Fonte: Microsoft Bot Framework Emulator, 2025

Quando o usuário envia uma mensagem, a *API* executa a classe instanciada *TeamsBot*

do arquivo *teamsBot.ts*, que contém a chamada à *API* do agente. Nessa etapa, é montado um *payload* com os dados necessários para o envio ao agente, destacando-se o parâmetro *conversationReference*, utilizado para identificar o usuário e manter o contexto da conversa:

```

1  const API_URL = "http://localhost:8000/api/v1/ask";
2
3  const WEBHOOK_URL = "http://localhost:3978/api/webhook";
4
5  const payload = {
6    user_id: user_id,
7    question: question,
8    session_id: session_id,
9    webhook_url: WEBHOOK_URL,
10   addressing_info: conversationReference
11 };

```

Por fim, a Figura 27 mostra a resposta recebida para a pergunta que definimos anteriormente.

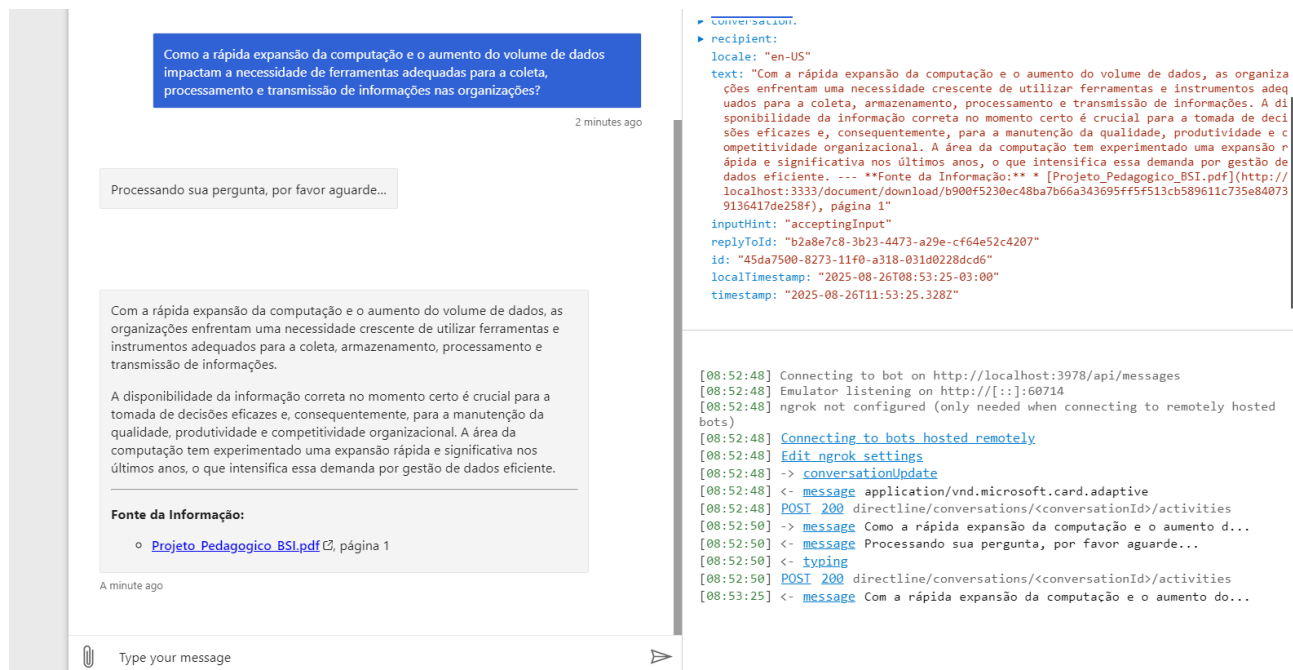


Figura 27 – Resposta da Pergunta no Chat Bot Framework
Fonte: Microsoft Bot Framework Emulator, 2025

5.7 Repositório e Controle de Versões

O código-fonte do sistema, incluindo o Agente de *IA*, o *RAG* e o *chatbot*, desenvolvido como parte deste trabalho, está disponível publicamente em repositórios no *GitHub*. Para clonar

os repositórios, recomenda-se seguir os passos descritos nos respectivos arquivos *README.md*, garantindo a correta implementação e execução do sistema localmente.

Caso sejam identificados *bugs*, sugestões de melhorias ou outros ajustes necessários, os repositórios estão abertos para contribuições e manutenção. Os links de acesso são:

- Agente de IA <https://github.com/taricklorran/AGENT_TCC>
- RAG <https://github.com/taricklorran/RAG_TCC>
- Chatbot <https://github.com/taricklorran/TEAMS_BOT_TCC>

6 Resultados

Esta Seção apresenta os resultados obtidos com a avaliação do sistema desenvolvido. O objetivo é verificar em que medida a solução proposta atendeu às expectativas quanto à clareza, consistência e eficiência no atendimento das solicitações dos usuários. Além disso, busca-se identificar pontos de melhoria e perspectivas para trabalhos futuros.

6.1 Perguntas, Respostas e Análises

Esta Seção apresenta os resultados obtidos nos testes realizados com o Agente de [IA](#). São exibidas as dez perguntas selecionadas, juntamente com suas respectivas respostas esperadas e as respostas fornecidas pelo agente, ilustradas em figuras correspondentes.

O objetivo é analisar a qualidade das respostas do sistema, conforme os critérios de avaliação definidos na Seção [4.4](#).

1ª Pergunta

“Como funciona a validação do estágio obrigatório? A documentação necessária deve ser apresentada apenas no final do estágio ou em qualquer etapa do processo?”

Resposta Esperada

“A validação de atividades profissionais funciona de forma semelhante ao estágio regular. Primeiro, é necessário registrar o plano de atividades e apresentar os documentos comprobatórios. A partir disso, o tempo começa a ser contabilizado. Ao final do período previsto, o estudante deve enviar o relatório e os pareceres de avaliação do supervisor e do orientador.

Em casos excepcionais, pode haver validação posterior, geralmente quando o estudante já está próximo da formatura.”

Resposta Obtida

Como funciona a validação do estágio obrigatório?

A validação do estágio obrigatório envolve as seguintes etapas:

1. **Envio da Documentação Inicial:** O processo começa com o envio da documentação completa de estágio pelo estudante à Coordenação de Estágio. Esta documentação deve estar datada e assinada por todas as partes envolvidas: estudante, supervisor (da empresa), representante legal da empresa (se aplicável), Coordenador de Estágios e orientador acadêmico. O orientador acadêmico assina apenas após a análise da documentação pela Coordenação de Estágio.
2. **Processamento e Formalização:** A Coordenação de Estágio tem um prazo de **10 dias úteis** após o recebimento para processar a documentação. A matrícula no Estágio Supervisionado Obrigatório (ESO) é formalizada após a chancela da documentação pelo Setor de Estágios da UFU (SESTA). Se o estágio for a partir de vínculo empregatício, a própria Coordenação de Estágio pode realizar a chancela.

Quando a documentação necessária deve ser apresentada?

A documentação é apresentada em diferentes momentos ao longo do processo de estágio:

- **No início do estágio:** A documentação necessária para a formalização e matrícula deve ser enviada antes do início ou no começo do processo de estágio.
- **Durante o estágio:** Se o estágio tiver duração superior a 6 meses, você deverá apresentar relatórios de atividades **semestrais**. Estes relatórios devem ser assinados pelo orientador acadêmico antes de serem enviados à Coordenação de Estágio.
- **No final do estágio:** Um relatório final de atividades é obrigatório. Ele deve ser apresentado em até **30 dias** após a data de término do estágio e deve conter a carga horária total cumprida. A Coordenação de Estágio, por sua vez, tem até **10 dias úteis** para emitir um parecer final (aprovado ou reprovado) sobre este relatório.

Fontes da Informação:

- [Normas de Estágio BSIMC – 2022.pdf](#) (páginas 5, 6, 8)
- [resolucao_facom.pdf](#) (página 7)
- [Estágio Supervisionado.pdf](#) (páginas 4, 5)

Figura 28 – Resposta da Pergunta 1
Fonte: Microsoft Bot Framework Emulator, 2025

Análise

O agente conseguiu fornecer uma resposta completa e bem estruturada referente a dúvida do estudante. Além disso, incluiu informações complementares citando as fontes e páginas. Entretanto, percebemos que o agente não conseguiu tomar iniciativa para informar sobre os casos excepcionais, como na resposta esperada.

2ª Pergunta

“Para a previsão de encerramento, basta indicar uma data que complete a carga horária mínima? No caso de vínculo empregatício integral, considera-se o prazo do contrato? E qual é o tempo mínimo exigido para convalidar o estágio obrigatório?”

Resposta Esperada

“Sim. O período deve ser suficiente para completar a carga horária mínima prevista para sua grade. Só não recomendo colocar muito apertado. É melhor passar do que faltar.”

Resposta Obtida

Olá! Entendo suas dúvidas sobre o Estágio Supervisionado Obrigatório (ESO) e a convalidação. Com base nas informações, esclareço os pontos:

1. **Previsão de encerramento do estágio:** Não basta apenas indicar uma data que complete a carga horária mínima. O término do Estágio Supervisionado Obrigatório (ESO) é considerado a **data de aprovação do Relatório de Atividades pela Coordenação de Estágio**. A carga horária mínima, por sua vez, deve estar em conformidade com o Projeto Pedagógico do Curso (PPC).
2. **Estágio com vínculo empregatício integral e prazo do contrato:** Em regime excepcional, o estudante que possuir vínculo empregatício ou for empreendedor pode solicitar que sua atividade profissional seja considerada para quitação do componente curricular de Estágio Supervisionado Obrigatório. Quanto ao prazo, a duração do estágio **não poderá exceder 2 (dois) anos** na mesma parte concedente, exceto quando se tratar de pessoa com deficiência.
3. **Tempo mínimo exigido para convalidar o estágio obrigatório:** Não foi encontrado um tempo mínimo específico exigido para convalidar o estágio obrigatório. Contudo, a atividade profissional com vínculo empregatício ou como empreendedor poderá ser requisitada, em caráter retroativo, para quitação do componente curricular de Estágio Supervisionado Obrigatório.

Para mais detalhes, você pode consultar os seguintes documentos:

- [INSTRUÇÃO NORMATIVA Nº 213.pdf](#) (páginas 2, 4, 6)
- [Estágio Supervisionado.pdf](#) (páginas 2, 3, 5)
- [Processo para Estágio Externo.pdf](#) (página 1)
- [resolucao_facom.pdf](#) (páginas 3, 5, 6, 8, 9)
- [cartilha_do_ministerio_do_trabalho_2010.pdf](#) (páginas 8, 18, 22)
- [Normas Gerais de Estágio de Graduação da UFU.pdf](#) (páginas 3, 4, 5, 8, 10, 11, 12, 13)
- [Normas de Estágio BSIMC – 2022.pdf](#) (páginas 4, 5, 6, 7, 8)
- [projeto_pedagogico_bsi_mc_2022.pdf](#) (páginas 28, 52)

Figura 29 – Resposta da Pergunta 2
Fonte: Microsoft Bot Framework Emulator, 2025

Análise

Podemos perceber que o agente conseguiu gerar a resposta para as três perguntas do estudante. Na primeira pergunta o agente conseguiu responder sobre o encerramento do estágio. Já para a segunda pergunta percebemos que o agente não deu uma resposta direta referente a “considerar o prazo de contrato”, citando a regra geral de 2 anos. Por fim, referente a terceira pergunta sobre o “tempo mínimo exigido” o agente não conseguiu obter a resposta no *RAG* e evitou de inventar uma possível resposta afirmando que não foi encontrado a informação, agindo corretamente.

3ª Pergunta

“Em estágio não obrigatório na modalidade remoto a empresa deve pagar ao estagiário o auxílio transporte? Há a possibilidade do valor deste auxílio ser detinado a outro benefício como, por exemplo, auxílio alimentação?”

Resposta Esperada

“Para estágios obrigatórios, benefícios como auxílio transporte não são obrigatórios. Para deixar isso mais claro, a opção deve ser preenchida no documento (TCE), por

exemplo, indicando o valor do auxílio em reais e, se não houver, “R\$ 0,0” ou “não há”.”

Resposta Obtida



Figura 30 – Resposta da Pergunta 3
Fonte: Microsoft Bot Framework Emulator, 2025

Análise

Nesta resposta, podemos perceber que o agente conseguiu interpretar a pergunta do estudante respondendo diretamente as duas questões. Mesmo não encontrando uma resposta específica para o estágio na modalidade remota, conseguiu aplicar corretamente a regra geral. Além disso, podemos perceber que o agente não conseguiu encontrar uma resposta para a segunda pergunta, que neste caso, não tinha a informação cadastrada na base de conhecimento, dando uma resposta correta.

4ª Pergunta

“Fui aprovado em um processo seletivo para estágio, com contrato previsto entre 1 ano e 1 ano e meio. Atualmente, tenho apenas as disciplinas de TCC 2 e Estágio Curricular a cumprir. Minha dúvida é: caso eu defenda o TCC neste semestre, poderei continuar no estágio curricular até o fim do contrato, mantendo o vínculo com a Universidade?”

Além disso, gostaria de confirmar quais documentos preciso enviar para validar corretamente o estágio.”

Resposta Esperada

“Sim, neste caso, você mantém o vínculo com a universidade se ainda estiver em estágio obrigatório. O limite é de dois anos na mesma empresa.”

Resposta Obtida

Olá!

Compreendo perfeitamente suas dúvidas em relação à continuidade do seu estágio e à validação dos documentos.

No entanto, após consultar nossa base de conhecimento, **não encontramos informações suficientes** para responder especificamente sobre a possibilidade de você continuar no estágio curricular após a defesa do TCC, mantendo seu vínculo com a universidade, nem sobre a lista exata de documentos necessários para a validação do estágio.

Para ter a certeza das informações e evitar qualquer tipo de problema, sugiro que você entre em contato diretamente com o setor responsável por estágios da sua Universidade. Eles poderão fornecer as orientações precisas, considerando o regimento interno da sua instituição, que é quem define essas regras.

Figura 31 – Resposta da Pergunta 4
Fonte: Microsoft Bot Framework Emulator, 2025

5ª Pergunta

“Trabalho para uma empresa no exterior na área de desenvolvimento de software, gostaria de saber se posso validar minhas atividades profissionais para equivalência e dispensa do componente de estágio obrigatório.”

Resposta Esperada

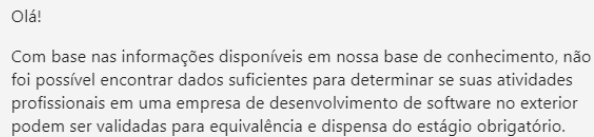
“Com as novas normas, é possível solicitar equivalência de estágio de duas formas: Pedido de equivalência pelas horas já trabalhadas – indicado em situações excepcionais, quando o estudante está próximo da formatura ou não pretende continuar na empresa. Nesse caso, o número de horas já cumpridas deve ser igual ou superior à carga exigida para o ESO.

Formalização do estágio – apresentando a documentação e o plano de atividades, para que as horas passem a ser contabilizadas daqui em diante.

Em ambos os casos, é necessário apresentar comprovação de vínculo e um termo de responsabilidade/veracidade assinado digitalmente. Também é obrigatória a entrega de relatório de atividades e pareceres do supervisor e do orientador: já no momento do pedido (1ª opção) ou ao final do período previsto (2ª opção).

Para mais detalhes, consulte a postagem da Equipe de Estágios: Normas de Estágio Atualizadas (Resolução 20/2024 CONFACOM). A documentação está disponível em: Vínculo Profissional.”

Resposta Obtida



Olá!

Com base nas informações disponíveis em nossa base de conhecimento, não foi possível encontrar dados suficientes para determinar se suas atividades profissionais em uma empresa de desenvolvimento de software no exterior podem ser validadas para equivalência e dispensa do estágio obrigatório.

Figura 32 – Resposta da Pergunta 5
Fonte: Microsoft Bot Framework Emulator, 2025

Análise

Tanto para a 4ª Pergunta, quanto a 5ª Pergunta, o Agente de IA não conseguiu recuperar a resposta correta na base de conhecimento, embora houvesse documentos disponíveis que poderiam auxiliar. Durante o processo de consulta, o agente reformula a pergunta do usuário para realizar a busca, mas, neste caso, a reformulação não foi adequada, o que possivelmente comprometeu a recuperação da informação correta.

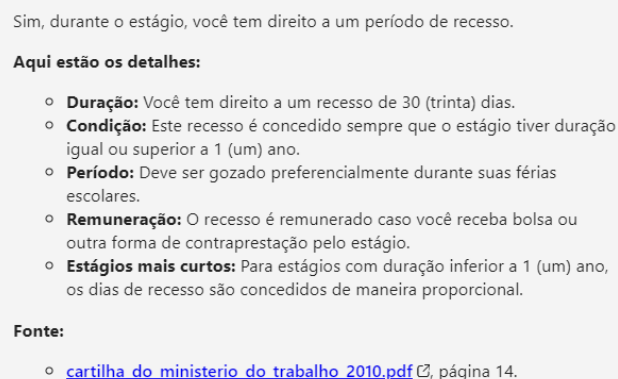
6ª Pergunta

“Tenho direito a férias durante o estágio?”

Resposta Esperada

“Sim. A legislação assegura um período de recesso de 30 dias sempre que o estágio tiver duração igual ou superior a 1 ano, que deve ser gozado preferencialmente durante suas férias escolares. Caso o estágio dure menos de um ano, o recesso será concedido de forma proporcional. Se você receber bolsa ou outra forma de contraprestação, esse recesso deverá ser remunerado.”

Resposta Obtida



Sim, durante o estágio, você tem direito a um período de recesso.

Aqui estão os detalhes:

- **Duração:** Você tem direito a um recesso de 30 (trinta) dias.
- **Condição:** Este recesso é concedido sempre que o estágio tiver duração igual ou superior a 1 (um) ano.
- **Período:** Deve ser gozado preferencialmente durante suas férias escolares.
- **Remuneração:** O recesso é remunerado caso você receba bolsa ou outra forma de contraprestação pelo estágio.
- **Estágios mais curtos:** Para estágios com duração inferior a 1 (um) ano, os dias de recesso são concedidos de maneira proporcional.

Fonte:

- [cartilha do ministerio do trabalho 2010.pdf](#), página 14.

Figura 33 – Resposta da Pergunta 6
Fonte: Microsoft Bot Framework Emulator, 2025

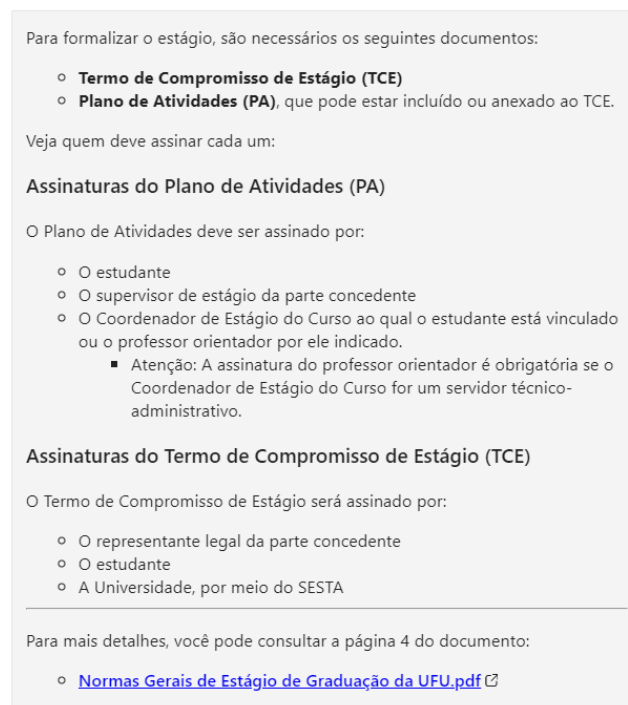
7ª Pergunta

“Quais são os documentos necessários para formalizar o estágio e quem deve assiná-los?”

Resposta Esperada

“O documento central para a formalização é o Termo de Compromisso de Estágio (TCE). Este termo, juntamente com o seu plano de atividades, deve ser assinado por você (estagiário), pelo seu supervisor na empresa, por um representante legal da empresa concedente, pelo seu professor orientador acadêmico e pelo Coordenador de Estágios da FACOM.”

Resposta Obtida



Para formalizar o estágio, são necessários os seguintes documentos:

- **Termo de Compromisso de Estágio (TCE)**
- **Plano de Atividades (PA)**, que pode estar incluído ou anexado ao TCE.

Veja quem deve assinar cada um:

Assinaturas do Plano de Atividades (PA)

O Plano de Atividades deve ser assinado por:

- O estudante
- O supervisor de estágio da parte concedente
- O Coordenador de Estágio do Curso ao qual o estudante está vinculado ou o professor orientador por ele indicado.
 - Atenção: A assinatura do professor orientador é obrigatória se o Coordenador de Estágio do Curso for um servidor técnico-administrativo.

Assinaturas do Termo de Compromisso de Estágio (TCE)

O Termo de Compromisso de Estágio será assinado por:

- O representante legal da parte concedente
- O estudante
- A Universidade, por meio do SESTA

Para mais detalhes, você pode consultar a página 4 do documento:


- [Normas Gerais de Estágio de Graduação da UFU.pdf](#) 

Figura 34 – Resposta da Pergunta 7

Fonte: Microsoft Bot Framework Emulator, 2025

8ª Pergunta

“Quem é o responsável pela contratação do seguro contra acidentes pessoais?”

Resposta Esperada

“A contratação de um seguro contra acidentes pessoais é uma obrigação da parte concedente (a empresa onde você estagia). No entanto, especificamente para o caso de estágio obrigatório, a legislação permite que essa responsabilidade seja, alternativamente, assumida pela instituição de ensino.”

Resposta Obtida

A responsabilidade pela contratação do seguro contra acidentes pessoais para o estagiário é, em geral, da parte concedente do estágio. No entanto, no caso de estágio obrigatório, essa responsabilidade pode ser alternativamente assumida pela UFU.

Referência:

- [Normas Gerais de Estágio de Graduação da UFU.pdf](#) (Página 7)

Figura 35 – Resposta da Pergunta 8
Fonte: Microsoft Bot Framework Emulator, 2025

Análise

Tanto para a 6ª Pergunta, 7ª Pergunta e para a 8ª Pergunta, percebemos que o agente conseguiu entregar uma resposta completa e correta para o estudante de forma objetiva, agregando valor ao estruturar a informação de forma clara e legível, além de disponibilizar a fonte.

9ª Pergunta

“Quais são os pré-requisitos acadêmicos para que eu possa iniciar o Estágio Supervisionado Obrigatório?”

Resposta Esperada

“Para iniciar o Estágio Supervisionado Obrigatório, você precisa ter concluído um conjunto de disciplinas cuja soma das cargas horárias seja igual ou superior à carga horária total prevista para os quatro primeiros períodos do curso.”

Resposta Obtida

Para iniciar o Estágio Supervisionado Obrigatório (ESO), o principal pré-requisito acadêmico é:

- Ter integralizado um conjunto de disciplinas cuja soma das cargas horárias seja igual ou superior à soma das cargas horárias das disciplinas previstas do 1º ao 4º período da grade curricular do seu curso.

Além disso, existem outros requisitos importantes para a matrícula no estágio:

- **Tempo e Carga Horária do Estágio:**
 - Para ingressantes até 2022: duração de 24 a 96 semanas, com carga horária mínima de 440 horas.
 - Para os demais alunos: duração máxima de 96 semanas, com carga horária mínima de 320 horas.
- **Tempo de Dedicação Semanal:**
 - O tempo máximo de dedicação ao Estágio é de 30 horas semanais.
 - Se você estiver matriculado apenas em Estágio Supervisionado, poderá ter uma dedicação de 40 horas semanais.

Você pode encontrar mais detalhes sobre esses requisitos no documento: [Estágio Supervisionado.pdf](#) (página 3).

Figura 36 – Resposta da Pergunta 9
Fonte: Microsoft Bot Framework Emulator, 2025

Análise

Embora o agente tenha conseguido responder corretamente à dúvida do estudante, a resposta apresentou menor objetividade em comparação às anteriores, trazendo informações adicionais que poderiam ser simplificadas.

10ª Pergunta

“Qual é a carga horária semanal máxima permitida para o estágio e qual a sua duração total?”

Resposta Esperada

“A jornada de atividades não pode ultrapassar 6 horas diárias e 30 horas semanais. Excepcionalmente, a carga horária poderá ser de até 40 horas semanais nos períodos em que não há aulas presenciais, desde que previsto no projeto pedagógico do curso, ou caso o Estágio Obrigatório e/ou o Trabalho de Conclusão de Curso sejam os únicos componentes que faltam para você se formar. A duração do estágio na mesma empresa não pode exceder 2 anos, exceto para estagiários com deficiência.”

Resposta Obtida

Conforme as regras, a carga horária semanal máxima permitida para o estágio é de **30 horas**. Em relação à duração total, o estágio não pode exceder **2 anos** na mesma empresa, exceto no caso de estagiários com deficiência.

Figura 37 – Resposta da Pergunta 10
Fonte: Microsoft Bot Framework Emulator, 2025

Análise

Apesar do agente fornecer uma resposta correta, ele ainda ficou incompleta. Ele não forneceu informação sobre as condições excepcionais em que a carga horária pode ser estendida o que demonstra a falta de capacidade para capturar os detalhes específicos e exceções importantes do regulamento.

6.2 Análise Geral

A análise geral dos resultados, discute o desempenho global do agente frente às métricas de avaliação e aos objetivos propostos conforme definidos na Seção 4.4. A seguir, a Figura 38 apresenta as avaliações das respostas para cada pergunta.

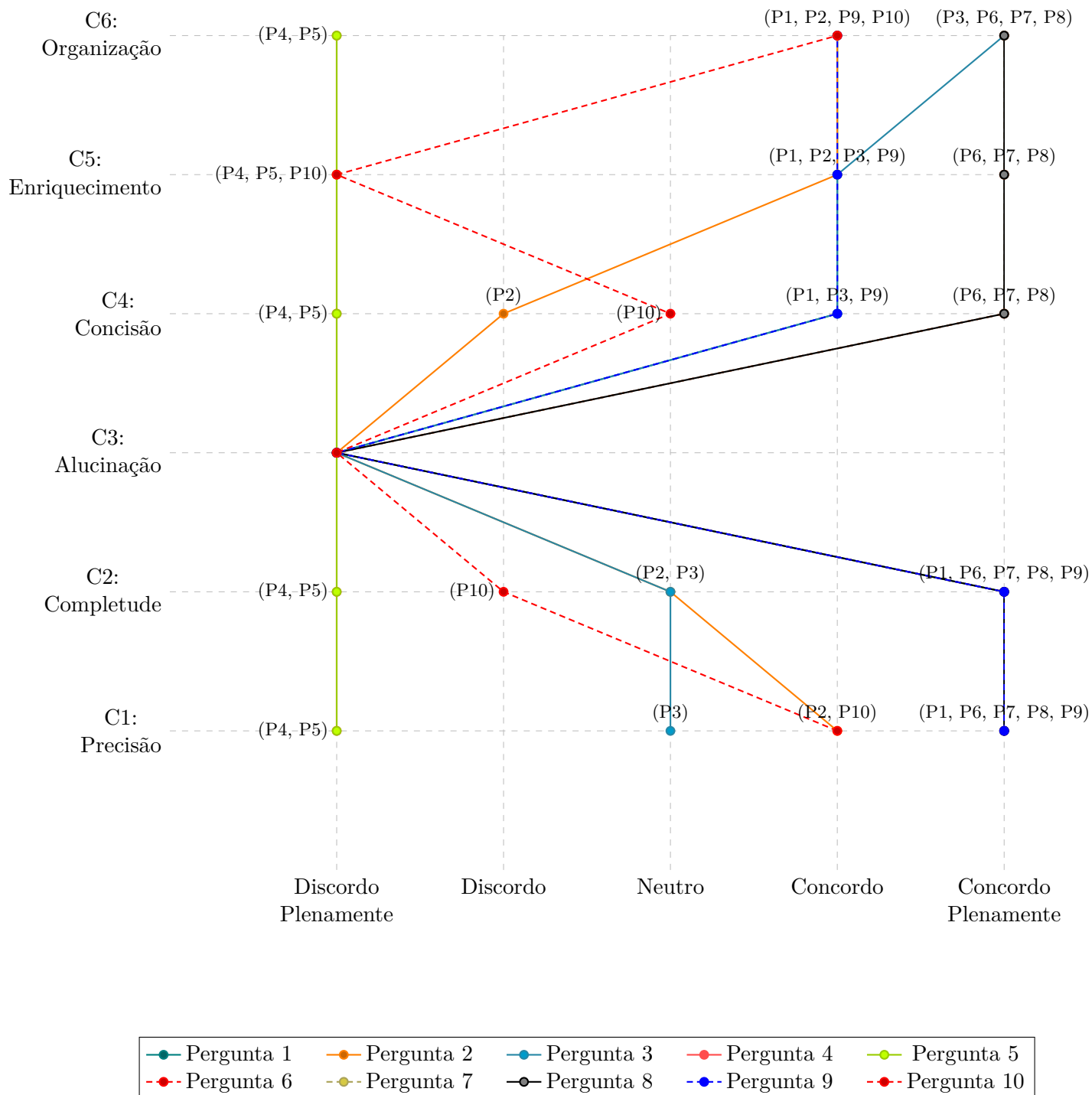


Figura 38 – Avaliação das respostas de cada pergunta

Das 10 perguntas avaliadas, os resultados distribuíram-se da seguinte forma:

- C1 - Precisão: 5 respostas foram classificadas como “Concordo Plenamente”, 2 como “Concordo”, 1 como “Neutro” e 2 como “Discordo Plenamente”.
- C2 - Completude: observamos 5 respostas em “Concordo Plenamente”, 2 em “Neutro”, 1 em “Discordo” e 2 em “Discordo Plenamente”.

- C3 - Alucinação: todas as respostas foram avaliadas como “Discordo Plenamente”, o que indica que não houve ocorrência de alucinação por parte do agente.
- C4 - Concisão: registramos 3 respostas como “Concordo Plenamente”, 3 como “Concordo”, 1 como “Neutro”, 1 como “Discordo” e 2 como “Discordo Plenamente”.
- C5 - Enriquecimento: os resultados apontaram 3 respostas em “Concordo Plenamente”, 4 em “Concordo” e 3 em “Discordo Plenamente”.
- C6 - Organização: foram contabilizadas 4 respostas como “Concordo Plenamente”, 4 como “Concordo” e 2 como “Discordo Plenamente”.

Os casos avaliados como “Neutro” ocorreram porque, embora as respostas apresentassem informações corretas, não abordaram diretamente o ponto central da pergunta. Essa limitação está relacionada ao fato de que a base de conhecimento não continha diretrizes suficientemente claras para determinadas situações, o que impediu a geração de respostas mais completas e alinhadas às expectativas do usuário.

Todas as perguntas submetidas ao sistema passam por um processo de reformulação automática, com o objetivo de melhorar a recuperação de informações no *RAG*. No entanto, para as perguntas 4 e 5, classificadas como “Discordo Plenamente”, essa reformulação acabou comprometendo o desempenho, pois o sistema não conseguiu localizar informações adequadas na base. Esse problema pode estar associado tanto à ausência de contexto suficiente na consulta quanto à falta de documentos com orientações específicas.

Na pergunta 4, o usuário questionava a possibilidade de realizar estágio por um período de um ano e meio, tendo apenas os componentes curriculares de TCC2 e Estágio pendentes, além de solicitar informações sobre os documentos necessários para validação. O gerente, ao reformular a pergunta, gerou a seguinte versão:

“Regulamentação estágio não obrigatório após defesa TCC com vínculo universitário. Documentos necessários para validação de estágio.”

Essa reformulação pode ter prejudicado a busca semântica, fazendo com que o sistema não recuperasse a resposta adequadamente. Caso a pergunta original tivesse sido mantida, a resposta correta teria sido a seguinte:

“Com base nas informações encontradas, se você estiver matriculado apenas no componente curricular de Estágio Obrigatório e/ou Trabalho de Conclusão de Curso, a jornada de atividades de estágio poderá ser de até 40 horas semanais, desde que isso esteja previsto no Projeto Pedagógico ou nas Normas Complementares de Estágio do Curso. Após a aprovação no TCC, o Colegiado do Curso deve decidir sobre a

convalidação do estágio. Em relação aos documentos, é necessário verificar a documentação exigida para a matrícula do estágio, que pode ser encontrada na página de estágio do curso ou com a coordenação de estágio.”

Situação semelhante ocorreu na pergunta 5, que foi reformulada da seguinte forma:

“Validação atividades profissionais empresa exterior desenvolvimento software equivalência dispensa estágio obrigatório.”

Nesse caso, a reformulação também comprometeu a busca, impossibilitando a recuperação da resposta. Se a versão original da pergunta tivesse sido utilizada, o sistema teria retornado a seguinte resposta:

“Sim, é possível validar suas atividades profissionais para equivalência e dispensa do componente de estágio obrigatório, desde que você possua vínculo empregatício formal na área de Informática e que suas atividades profissionais sejam supervisionadas e diretamente relacionadas às competências técnicas e habilidades gerais previstas no Projeto Pedagógico do curso. Para isso, você deve encaminhar a documentação necessária por e-mail para a Coordenação de Estágio. Se você é funcionário, precisará enviar uma cópia da sua carteira digital de trabalho ou uma declaração da empresa com carimbo e/ou assinatura virtual com data inferior a 20 dias úteis à data do envio, além do Histórico Escolar, Plano de Atividades da FACOM com um cronograma e um documento contendo seu NOME, MATRÍCULA, TELEFONE e E-MAIL.”

6.3 Limitações e Trabalhos Futuros

A avaliação do agente de [IA](#) evidenciou que, embora tenha sido capaz de responder corretamente à maioria das perguntas, algumas limitações ainda comprometem a experiência do usuário. Um dos principais pontos está relacionado ao processo de reformulação das consultas para o mecanismo de recuperação de informações ([RAG](#)). Em determinadas situações, mesmo havendo documentos na base capazes de fornecer a resposta, o agente não conseguiu resgatar o conteúdo adequado, resultando em respostas incompletas ou incorretas, como ocorreu nas Perguntas 4 e 5.

Outro aspecto identificado refere-se à objetividade das respostas. Em alguns casos, o agente apresentou informações corretas, mas pouco diretas, o que pode dificultar a compreensão imediata por parte do estudante. Além disso, embora o sistema tenha sido capaz de estruturar as respostas de forma clara na maioria das vezes, ainda há espaço para aprimorar a inclusão de informações complementares, como links e documentos relevantes, que foram pouco explorados.

Como trabalhos futuros, podemos sugerir o aprimoramento do processo de reformulação das perguntas realizadas pelo agente e da recuperação das informações no *RAG*, de modo a aumentar a precisão. Outro ponto relevante é a melhoria dos *prompts* utilizados, tanto no *RAG* quanto no agente, para que considerem detalhes específicos que possam auxiliar o estudante, ampliando a qualidade das respostas geradas.

Também se destaca a necessidade de desenvolver uma *interface frontend* para cadastro e atualização dos agentes, conforme recomendado na Seção 5.3, a fim de facilitar a experiência dos usuários responsáveis pela administração da solução.

Por fim, podemos considerar a ampliação do escopo de atuação do agente, com a inclusão de funcionalidades de automação de processos administrativos, como análise de documentos e emissão de relatórios. Essas melhorias podem tornar o sistema mais completo e eficiente, potencializando sua contribuição no apoio às atividades da coordenação de Estágio Supervisionado e no atendimento aos estudantes.

7 Conclusão

O desenvolvimento do agente de IA proposto, demonstrou a viabilidade de utilizar técnicas de IA Generativa integradas ao *Microsoft Teams* como suporte ao atendimento da Coordenação de Estágio Supervisionado. Os resultados indicaram que o sistema foi capaz de responder adequadamente a diferentes tipos de perguntas, mostrando-se funcional no esclarecimento de dúvidas dos estudantes. Além disso, a arquitetura de agentes implementada se mostrou flexível, permitindo a integração de novas ferramentas e funcionalidades, o que amplia o potencial de uso da solução em outros contextos administrativos. Nesse sentido, a proposta representa uma possibilidade de avanço na aplicação de agentes inteligentes no ambiente acadêmico, oferecendo uma alternativa para otimizar o atendimento e apoiar as atividades da coordenação.

Apesar dos avanços alcançados, o estudo também evidenciou algumas limitações. A principal delas está relacionada à dependência da qualidade dos documentos indexados, que influencia diretamente a precisão das respostas. Outra dificuldade observada refere-se ao processo de reformulação automática das perguntas, que em alguns casos comprometeu a recuperação adequada das informações.

Como propostas de melhorias futuras temos o uso de mecanismos de reformulação e busca semântica para minimizar perdas de contexto; a implementação métricas de monitoramento contínuo de desempenho, permitindo ajustes dinâmicos no limiar de similaridade e nas estratégias de recuperação; o desenvolvimento de agentes capazes de executar tarefas administrativas como, por exemplo, análise e leituras de documentos enviados pelos estudantes; e adaptação para múltiplos canais de comunicação, como *WhatsApp*, *Telegram*, entre outros.

De maneira geral, este trabalho contribui ao apresentar uma solução prática e inovadora, baseada em agentes de IA, que poderá apoiar a coordenação de estágio e melhorar a experiência dos estudantes.

Referências

- ARAÚJO, E. F. dos S. **SOLUÇÃO CHATBOT NO AMBIENTE ACADÊMICO DA UFRJ**. 2020. <<https://pantheon.ufrj.br/handle/11422/18243>>. [Online; accessed 24-Março-2025]. Citado na página 16.
- ARIDAN, M.; IRWANDANI; VILLARAMA, J. A.; FUENTE, A. V. D. Post-launch public interest in deepseek vs.chatgpt: A comparative google trends analysis in indonesia and the philippines. **Blockchain, Artificial Intelligence, and Future Research**, v. 1, n. 1, p. 1–12, 2025. Disponível em: <<https://journal.wiseedu.co.id/index.php/bafrjournal/article/view/>>. Acesso em: 14 abr. 2025. Citado na página 14.
- BHAT, S. R.; RUDAT, M.; SPIEKERMANN, J.; FLORES-HERR, N. **Rethinking Chunk Size For Long-Document Retrieval: A Multi-Dataset Analysis**. 2025. <<https://arxiv.org/abs/2505.21700>>. [Online; accessed 09-Agosto-2025]. Citado na página 33.
- CLOUDFLARE. **O que são embeddings no aprendizado de máquina?** 2025. <<https://www.cloudflare.com/pt-br/learning/ai/what-are-embeddings/>>. [Online; acessado em 04-Outubro-2025]. Citado na página 20.
- CORDEIRO, K. M. de A. **O IMPACTO DA PANDEMIA NA EDUCAÇÃO: A UTILIZAÇÃO DA TECNOLOGIA COMO FERRAMENTA DE ENSINO**. 2020. <<https://l1nq.com/effjO>>. [Online; accessed 10-Março-2025]. Citado na página 11.
- COSTA, T. S. da. **A UTILIZAÇÃO DE CHATBOT COMO INSTRUMENTO PARA SE COMUNICAR COM CONSUMIDORES ONLINE: UM ESTUDO DE CASO EM UMA EMPRESA DE VAREJO**. 2023. <<https://repositorio.ufsc.br/handle/123456789/248266>>. [Online; accessed 23-Março-2025]. Citado na página 15.
- CUNHA, M. B. da. Os possíveis impactos do chatbot no setor de mecanismo de busca. **Revista Ibero-Americana de Ciência da Informação**, v. 16, n. 1, p. 1–4, 2023. Disponível em: <<https://periodicos.unb.br/index.php/RICI/article/view/47662>>. Acesso em: 16 mar. 2025. Citado na página 14.
- 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. <<https://arxiv.org/abs/2312.10997>>. [Online; accessed 09-Agosto-2025]. Citado na página 15.
- GONZALEZ, M.; LIMA, V. L. S. de. **Recuperação de Informação e Processamento da Linguagem Natural**. 2003. <<https://www.marilia.unesp.br/Home/Instituicao/Docentes/EdbertoFerneda/mri-06---gonzales-e-lima-2003.pdf>>. [Online; accessed 19-Março-2025]. Citado na página 13.
- IGNITE, M. **Visão geral do Microsoft Teams Toolkit**. 2025. <<https://learn.microsoft.com/pt-br/microsoftteams/platform/toolkit/teams-toolkit-fundamentals>>. [Online; acessado em 23-Setembro-2025]. Citado na página 20.
- MANFIO, E. R.; MORENO, F. **A EVOLUÇÃO DOS CHATTERBOTS: PLN, I.A. E DIFUSÃO CULTURAL**. 2018. <<https://pesquisafatec.com.br/ojs/index.php/efatec/article/view/87>>. [Online; accessed 20-Março-2025]. Citado na página 14.

- MARTINS, B. R.; DINIZ, W. G.; SILVA, R. oliveira da. A complexibilidade da uml e seus diagramas. **TECNOLOGIAS EM PROJEÇÃO**, v. 8, n. 1, p. 86–99, 2017. Disponível em: <<http://www.reference.com/go/http://csrc.nsl.nist.gov/publications/nistpubs/800-94/SP800-94.pdf>>. Citado 2 vezes nas páginas 24 e 27.
- MARTINS, D. Bia, a ia do bradesco, se tornou uma ‘conciierge’, diz cio do banco. **Finsiders Brasil**, 2023. [Online; accessed 16-Março-2025]. Disponível em: <<https://finsidersbrasil.com.br/ia/bia-a-ia-do-bradesco-se-tornou-uma-conciierge-diz-cio-do-banco>>. Citado na página 11.
- MASOOD, A. Optimizing chunking, embedding, and vectorization for retrieval-augmented generation. **Medium**, 2025. [Online; accessed 10-Agosto-2025]. Disponível em: <<https://medium.com/%40adnanmasood/optimizing-chunking-embedding-and-vectorization-for-retrieval-augmented-generation-ea3b083b68f7>>. Citado na página 33.
- MICROSOFT. Bradesco alcança +80**Finsiders Brasil**, 2024. [Online; accessed 16-Março-2025]. Disponível em: <<https://www.microsoft.com/pt-br/customers/story/18828-banco-bradesco-azure-ai-services>>. Citado na página 11.
- MONTEIRO, G. S. **Helena: um chatbot para auxílio dos discentes do DECOM em trâmites universitários**. 2021. <<http://www.monografias.ufop.br/handle/35400000/3331>>. [Online; accessed 07-Abril-2025]. Citado na página 17.
- OPENAI. **Por que os modelos de linguagem alucinam?** 2025. <<https://openai.com/pt-BR/index/why-language-models-hallucinate/>>. [Online; accessed 15-Setembro-2025]. Citado na página 15.
- PACHECO, F. C. **Estudo e desenvolvimento de um ChatBot para automação de atendimento ao cliente**. 2021. <<https://repositorio.ufu.br/handle/123456789/36482>>. [Online; accessed 20-Março-2025]. Citado 3 vezes nas páginas 11, 13 e 17.
- RAMOS, A. S. M. **Inteligência Artificial Generativa baseada em grandes modelos de linguagem - ferramentas de uso na pesquisa acadêmica**. 2023. <<https://preprints.scielo.org/index.php/scielo/preprint/view/6105>>. [Online; accessed 19-Março-2025]. Citado na página 14.
- ROCHA, V. C. **Uma aplicação Web para auxiliar o fluxo de estágios da FACOM: Detalhamento de sua arquitetura, dos fluxos e da máquina de estados**. 2023. <<https://repositorio.ufu.br/handle/123456789/39771>>. [Online; accessed 10-Abril-2025]. Citado na página 17.
- SAHOO, P.; SINGH, A. K.; SAHA, S.; JAIN, V.; MONDAL, S.; CHADHA, A. **A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications**. 2025. <<https://arxiv.org/abs/2402.07927>>. [Online; accessed 25-Agosto-2025]. Citado na página 16.
- SANTOS, M. C. dos. Entrevistando um robô: notas sobre a aplicação experimental da metodologia eeaf usando a ferramenta chatgpt de inteligência artificial. **Comunicação Inovação**, v. 24, n. 49, p. 1–17, 2023. Disponível em: <https://seer.uscs.edu.br/index.php/revista_comunicacao_inovacao/article/view/8987>. Acesso em: 16 mar. 2025. Citado na página 14.

- SCHUNK, L. M. **O USO DE INTELIGÊNCIA ARTIFICIAL POR MEIO DE CHATBOTS NO PROCESSO DE ATENDIMENTO AO CLIENTE: UM ESTUDO SOBRE SEUS BENEFÍCIOS**. 2020. <<https://hdl.handle.net/10438/29943>>. [Online; accessed 10-Março-2025]. Citado na página 11.
- SOUZA, A. F. D.; SILVA, C. D. D. **DESENVOLVIMENTO DE CHATBOT PARA EMPRESAS DE PEQUENO PORTE**. 2021. <<https://ric.cps.sp.gov.br/handle/123456789/10136>>. [Online; accessed 23-Março-2025]. Citado na página 16.
- SÁ, A. M. B. d. **Uso de Large Language Models (LLMs) para Auxílio na Correção de Bugs**. 2024. <<https://repositorio.ufu.br/handle/123456789/44072>>. [Online; accessed 08-Abril-2025]. Citado na página 14.
- VOGEL, D.; RAMOS, A. M.; FRANZONI, A. M. B. Transformando a educação com large language models (llms): benefícios, limitações e perspectivas. **REVISTA CADERNO PEDAGÓGICO**, v. 22, n. 4, p. 1–20, 2025. Disponível em: <<https://ojs.studiespublicacoes.com.br/ojs/index.php/cadped/article/view/13846>>. Acesso em: 20 mar. 2025. Citado na página 14.