UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO FACULDADE DE CIÊNCIAS
CAMPUS BAURU

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Leonardo Vieira Maurino
Guilherme Mendonça Pinheiro

TecnoHelp chatbot para suporte técnico

Bauru, 2025

UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO FACULDADE DE CIÊNCIAS
CAMPUS BAURU

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Leonardo Vieira Maurino
Guilherme Mendonça Pinheiro

TecnoHelp chatbot para suporte técnico

Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Sistemas de Informação, como requisito para obtenção do título de Bacharel em Sistemas de Informação, sob orientação do Prof. Dr. José Remo Ferreira Brega.

Bauru, 2025

Maurino, Leonardo Vieira.
TecnoHelp chatbot para suporte técnico / Leonardo Vieira Maurino, Guilherme Mendonça Pinheiro. – Bauru, 2025
40 f. : il., tabs.

Trabalho de conclusão de curso (Bacharelado – Sistemas de Informação)–Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientador: José Remo Ferreira Brega

1. Software Desenvolvimento. 2. Gestão do conhecimento. 3. Assistência técnica. I. Pinheiro, Guilherme Mendonça. II. Universidade Estadual Paulista. Faculdade de Ciências. III. Título.

RESUMO

Este trabalho propõe o desenvolvimento de um chatbot especializado em suporte técnico remoto, com ênfase na centralização de documentações técnicas dispersas, alinhado às boas práticas de gestão do conhecimento e governança de TI. Em um cenário de crescente demanda por soluções ágeis e econômicas, a ferramenta utiliza Inteligência Artificial para interpretar perguntas e fornecer respostas de acordo com uma base de dados estruturada, reduzindo o tempo de busca por informações e aumentando a eficiência no atendimento. Além de reunir e organizar o conhecimento técnico de forma acessível, o chatbot também atua como recurso educacional interativo para o treinamento de novos profissionais, podendo ser constantemente atualizado para garantir acesso a conteúdos confiáveis e atualizados.

Palavras-chave: Suporte técnico remoto, chatbot, centralização de documentos, inteligência artificial e treinamento técnico.

ABSTRACT

This work proposes the development of a chatbot specialized in remote technical support, with an emphasis on centralizing dispersed technical documentation in accordance with best practices in knowledge management and IT governance. In a context of growing demand for agile and cost-effective solutions, the tool leverages Artificial Intelligence to interpret questions and provide answers in accordance with a structured database, reducing the time spent searching for information and increasing support efficiency. In addition to organizing and providing access to technical knowledge, the chatbot also serves as an interactive educational resource for training new professionals, and can be continuously updated to ensure reliable and up-to-date content.

Keywords: Remote technical support, chatbot, document centralization, artificial intelligence, technical training.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

No mundo cada vez mais digitalizado, garantir que equipamentos e sistemas funcionem de forma eficiente é essencial para a produtividade das empresas e a satisfação dos clientes. O suporte técnico remoto surgiu como uma solução fundamental, permitindo a resolução de problemas sem a necessidade de deslocamento físico, o que resulta na redução de custos operacionais com transporte, hospedagem e logística. Almeida (2017, p. 34) destaca que o valor do suporte técnico é intrínseco à capacidade de oferecer uma solução que garanta a satisfação do cliente, considerando a qualidade do atendimento, a eficácia da solução proposta e o tempo dedicado à resolução do problema.

No entanto, os técnicos de campo enfrentam desafios significativos ao buscar informações, manuais e documentos que se encontram dispersos em diferentes repositórios, seja na web, em pastas variadas ou ainda como conhecimento tácito. Essa dispersão de documentação técnica e conhecimento, que abrange elementos cruciais como diagramas elétricos, manuais de peças, listas de códigos de erro e procedimentos específicos de reparo, compromete a eficiência dos processos operacionais. Em atividades que demandam respostas rápidas, como o suporte técnico, a dificuldade de acesso imediato à informações precisas resulta em diagnósticos lentos e, por vezes, imprecisos. A ineficácia na localização e no uso de informações essenciais, somada à demora na capacitação de novos técnicos, prolonga o tempo de atendimento e, em certas situações, impede a resolução do problema em um único contato ou visita.

A Gestão do Conhecimento emerge, nesse contexto, como um pilar fundamental para a eficiência operacional. A construção de uma base de conhecimento centralizada e integrada é essencial para mitigar o problema da dispersão, garantindo que as informações estejam acessíveis a todos os colaboradores, independentemente de sua posição. Nonaka e Takeuchi (2008) enfatizam que, em um ambiente de rápidas transformações, o conhecimento tende a se tornar obsoleto logo após sua criação, o que exige um processo contínuo de criação e disseminação. Assim, a gestão do conhecimento é definida como o processo ininterrupto de criar, disseminar e incorporar novos conhecimentos em produtos, serviços, tecnologias e sistemas.

Um exemplo notável da aplicação prática desses princípios é a General Electric (GE), que estruturou um sistema capaz de armazenar 1,5 milhão de problemas e suas respectivas soluções. Essa centralização permitiu que um operador localizasse a informação correta em apenas dois segundos, graças ao uso de tecnologias de Inteligência Artificial e sistemas de busca eficientes (Nonaka e Takeuchi, 2008, p. 67). Dessa forma, a centralização, reclassificação e organização dos documentos representa passos fundamentais para a melhoria da eficiência operacional, pois um repositório bem estruturado de conhecimento explícito não apenas reduz o tempo de busca por informações, mas também promove sua disseminação ágil, favorecendo a continuidade dos processos e a capacidade de resposta da organização (Nonaka e Takeuchi, 2008, p. 28). No modelo de organização Hipertexto, a centralização, reclassificação e organização dos documentos técnicos é um passo fundamental para a melhoria da eficiência operacional.

A relevância do problema da dispersão e da gestão do conhecimento estende-se por diversos setores da sociedade, impactando diretamente a qualidade dos serviços prestados. Empresas que prestam suporte técnico remoto tendem a sofrer com a falta de uma estrutura otimizada para a capacitação de novos técnicos e o acesso ágil a informações essenciais, o que pode comprometer a qualidade do serviço, a satisfação dos clientes e a reputação da empresa. Dessa forma, fabricantes de máquinas e equipamentos também podem enfrentar dificuldades na prestação de suporte eficiente, resultando em custos operacionais elevados e possíveis impactos negativos na experiência do cliente. Os clientes finais, sejam indústrias, comércios ou consumidores domésticos, também são afetados, pois a demora na resolução de problemas técnicos pode levar a perdas financeiras, paralisação de atividades produtivas e insatisfação generalizada.

Os impactos econômicos desses problemas são consideráveis. Empresas enfrentam prejuízos quando suas máquinas permanecem inoperantes devido à demora no suporte técnico, causando perda de produtividade e redução no faturamento. No comércio, a ineficiência no suporte a equipamentos pode comprometer a operação de estabelecimentos que dependem diretamente de máquinas e dispositivos para suas atividades diárias. Já no âmbito doméstico, a impossibilidade de utilizar um eletrodoméstico essencial, como uma máquina de lavar, pode gerar transtornos significativos na rotina dos consumidores. A

implementação da solução proposta busca minimizar esses impactos, aumentando a agilidade dos atendimentos e reduzindo custos operacionais, beneficiando tanto as empresas fornecedoras de suporte quanto seus clientes.

Para solucionar esses desafios, este trabalho propõe o desenvolvimento de um chatbot especializado, baseado em geração aumentada via recuperação (RAG) que centraliza documentos técnicos e agiliza o acesso a informações essenciais. Essa ferramenta utiliza Inteligência Artificial por meio de modelos de LLM (que significa *Large Language Model* ou Grande Modelo de Linguagem, um tipo de Inteligência Artificial treinada com imensos volumes de dados de texto e código para entender, gerar e processar linguagem humana) para interpretar perguntas e fornecer respostas rápidas e precisas, garantindo que os técnicos tenham sempre à disposição conteúdos atualizados e confiáveis. Além de otimizar o suporte técnico, o chatbot também será útil no treinamento de novos profissionais, tornando o aprendizado mais dinâmico e acessível. Conforme a implementação de Almeida (2017, p. 39-42), que resultou em um ganho de 40% na agilidade, este projeto busca reduzir o tempo de treinamento e de chamados técnicos através da plataforma proposta.

A implementação dessa solução, contudo, apresenta desafios consideráveis. O principal obstáculo será garantir que o chatbot compreenda corretamente perguntas técnicas variadas, fornecendo respostas precisas sem margem para erros que possam comprometer diagnósticos e reparos. Além disso, será necessário integrar a ferramenta a diferentes bases de dados e documentos técnicos, padronizando informações de forma eficaz. Outro desafio relevante será a adaptação da ferramenta às mudanças tecnológicas e à evolução constante dos equipamentos, garantindo que o chatbot permaneça atualizado e relevante para os técnicos.

Com a execução deste projeto, busca-se não apenas melhorar a eficiência do suporte técnico remoto, mas também transformar a maneira como o conhecimento técnico é acessado e compartilhado, proporcionando benefícios significativos tanto para os profissionais quanto para as empresas.

2. SOLUÇÕES EXISTENTES E OBJETIVOS DA PROPOSTA

O avanço da tecnologia tem impulsionado o desenvolvimento de diversas soluções voltadas para o suporte técnico remoto, tanto em ambientes comerciais quanto acadêmicos. Soluções tais como: automatização de atendimentos, otimização do acesso a informações técnicas e maior agilidade na capacitação de profissionais. Contudo, apesar da variedade de ferramentas disponíveis, muitas delas não se adaptam com eficiência ao contexto específico de suporte técnico, que necessita de respostas rápidas, precisas e que integram documentos técnicos de diferentes fontes.

Entre as soluções comerciais analisadas - vide Tabela 1 -, temos o ChatGuru (Guru, 2023) que se destaca como plataforma especializada em atendimento automatizado para suporte técnico. Seu principal diferencial reside na integração com bases de conhecimento técnico, permitindo respostas precisas a dúvidas frequentes através de Inteligência Artificial. A ferramenta oferece funcionalidades avançadas como consulta a documentos técnicos e histórico de atendimentos, além de suporte a múltiplos canais de comunicação. No entanto, como solução comercial, apresenta custos que podem ser proibitivos para pequenas empresas e exige adaptações para atender às necessidades específicas de técnicos que trabalham com diferentes tipos de equipamentos.

Outra solução significativa encontrada no mercado é o CS-Chatbot da Prodesp (Prodesp, 2022), desenvolvido especificamente para atendimento técnico especializado. Esta ferramenta demonstra eficácia na automatização de respostas e resolução de problemas técnicos recorrentes em sistemas públicos. Apesar de seu bom desempenho em contextos institucionais, apresenta limitações na personalização para diversos setores e tipos de equipamentos, além de depender fortemente da infraestrutura tecnológica da organização que a implementa.

No âmbito acadêmico, diversas pesquisas têm explorado aplicações de chatbots para suporte técnico, particularmente em ambientes educacionais e institucionais. Embora essas iniciativas demonstrem o potencial da tecnologia, muitas ainda se baseiam em modelos de respostas pré-definidas, com capacidade limitada para interpretar linguagem técnica especializada ou integrar documentos complexos de diferentes fontes.

Esta análise aparenta revelar uma lacuna significativa no mercado: a falta de uma solução verdadeiramente flexível, acessível e adaptável às necessidades específicas do suporte técnico remoto e de campo. Como tal, visando enfrentar este desafio, um dos objetivos centrais deste projeto foi o desenvolvimento de uma solução baseada em RAG (técnica que permite que grandes modelos de linguagem recuperem e incorporem novas informações, além do treinamento padrão), além disso, a solução combina um modelo de Inteligência Artificial generativa, a LLM Llama 3. Com ambos os fatores combinados, a solução desenvolvida visou apresentar um canal que permita aos técnicos sanarem suas dúvidas e solicitarem informações de documentos diretamente ao chatbot via interface desenvolvida. Dessa forma, esse assistente virtual é capaz de interpretar a linguagem técnica especializada e tem como objetivos reduzir o tempo de treinamento de novos técnicos, permitir um aumento na eficiência do suporte remoto com diminuição de visitas presenciais e redução no tempo total de atendimento em chamados, além da implementação de um mecanismo para atualização contínua da base de conhecimento, garantindo que o sistema permaneça alinhado com os avanços tecnológicos.

Dessa forma, a solução desenvolvida visa atingir um nicho de mercado apresentado na Tabela 1. Nela é possível observar os principais pontos que nortearam e geraram as motivações para a existência e desenvolvimento desta aplicação.

Tabela 1 - Comparação de soluções existentes

Característica / Solução	ChatGuru	CS-Chatbot (Prodesp)	Chatbots Acadêmicos	TecnoHelp
Natureza / Base	Comercial, plataforma especializada em atendimento automatizado.	Governamental, desenvolvido para sistemas públicos.	Focados em ambientes educacionais e institucionais.	Projeto desenvolvido para suporte técnico utilizando ferramentas gratuítas
Custo	Alto Custo(foco em médias e grandes empresas)	Não especificado, mas demanda infraestrutura maior	Não especificado (Geralmente não possuem foco comercial)	Baixo custo e utilização de plataformas gratuitas.
Personalização	Média, exige grandes adaptações	Baixa, modelo generalista	Capacidade limitada para atender outros cenários e setores comerciais/produtivos	Alta, pois integra bases de conhecimento.
Limitações	Custos elevados e necessidade de adaptações para demandas específicas.	Pouca personalização e forte dependência de infraestrutura	Pouca personalização, modelos generalistas	Projetado para superar essas limitações, visando baixo custo e média/alta personalização
Diferenciais	Integração com bases de conhecimento; Consulta a documentos e históricos;	Eficácia na automação de respostas.	Demonstram o potencial da tecnologia para aplicações futuras.	Foco em nicho, baixo custo e uso de plataforma gratuíta; melhora de eficiência operacional e diminuição do tempo de chamados.

Fonte: Elaborado pelos autores.

3. DESCRIÇÃO DA APLICAÇÃO

Foi desenvolvido um sistema de chatbot inteligente voltado para o suporte técnico remoto baseado em geração aumentada via recuperação, com o objetivo de centralizar informações técnicas e agilizar o atendimento de chamados. Tendo em vista o foco de otimizar o acesso a manuais, procedimentos e documentações técnicas, ao mesmo tempo em que contribui para a redução do tempo de treinamento de novos técnicos. A interação com o sistema ocorre por meio da API do WhatsApp, permitindo ao técnico conversar com o chatbot como se estivesse dialogando com um atendente humano, de forma prática e acessível.

A base do sistema está construída no motor LLama 3, que permite ao chatbot aprender padrões de diálogo com base em treinamentos prévios e respostas configuradas. Para armazenamento das informações e controle de acesso, foi utilizado um banco de dados relacional, o PostgreSQL. O banco de dados armazena os logs de interações e históricos de conversas.

O sistema é dividido logicamente em quatro módulos principais. O primeiro módulo, responsável pela interface com o usuário, será baseado na API do WhatsApp “Baileys”, que recebe as mensagens dos técnicos e devolve as respostas do chatbot. Este canal facilita a comunicação, já que o WhatsApp é amplamente utilizado no ambiente corporativo e não exige que o usuário instale novos aplicativos. Esse módulo de interface também inclui uma página web como segunda opção e uma interface de administrador para a inclusão de arquivos e documentos na base de conhecimento, além da gestão do software em si.

O segundo módulo, o motor de interpretação, foi implementado com o uso do modelo de LLM LLama 3. Este módulo é responsável por interpretar as mensagens recebidas, buscar padrões nas perguntas e mapear respostas apropriadas com base em sua base de conhecimento. Além disso, ele permitirá o treinamento contínuo do sistema para melhorar a assertividade das respostas ao longo do tempo.

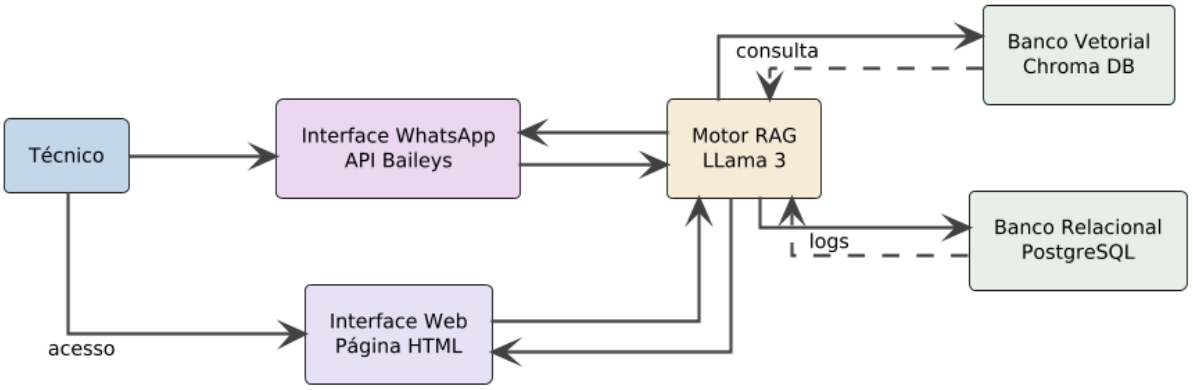
O terceiro módulo é o banco de dados, que armazena os logs de interações e históricos de conversas. Este módulo também contém metadados e categorias para facilitar buscas mais eficientes.

Por fim, o quarto módulo cuida da melhoria contínua do sistema, permitindo que o chatbot seja constantemente atualizado com novas informações, adaptando-se à evolução dos equipamentos e às necessidades dos técnicos. Esse

módulo é baseado e construído na forma de um banco de dados vetorial (Chroma DB) que armazena os "*embeddings* vetoriais" que nada mais são que representações numéricas de dados não estruturados.

Essa estrutura modular garantiu que cada parte do sistema fosse desenvolvida, testada e mantida de forma independente. Ao final, espera-se que a solução proporcione ganhos significativos em eficiência, qualidade no atendimento técnico e economia operacional. Na FIGURA 1 temos uma representação visual de como se dá a estrutura modular do sistema.

FIGURA 1 - Fluxograma de funcionamento dos módulos



Fonte: Elaborado pelos autores.

3.1 ESPECIFICAÇÃO DA APLICAÇÃO

Esta seção tem como objetivo apresentar de forma visual e sintética a arquitetura e funcionamento do TecnoHelp através de seus diagramas principais, estabelecendo uma compreensão clara da estrutura do sistema antes de detalhes técnicos posteriores. Os diagramas incluídos - de Blocos, Casos de Uso, Classes, Sequência e Relacionamento do banco de dados - foram cuidadosamente elaborados para: demonstrar a organização modular do sistema, especificar as interações entre usuários e funcionalidades, modelar os componentes estáticos e dinâmicos, e representar a estrutura de dados subjacente.

Através desta abordagem visual, busca-se proporcionar uma visão holística imediata do projeto, mostrando como as diferentes camadas (interface, processamento e armazenamento) se integram para cumprir os objetivos de otimizar o suporte técnico. A especificação por diagramas permite verificar a coerência

arquitetural antes da implementação, assegurando que todos os requisitos do sistema estejam adequadamente contemplados na estrutura projetada. Esta representação gráfica serve ainda como documento de referência para o desenvolvimento e futuras expansões do sistema. Esta seção visa detalhar e fornecer uma visão abrangente sobre a construção e os componentes do sistema e do projeto.

Entre as principais funcionalidades do aplicativo estão:

- **Consulta Técnica via Chatbot:** O técnico interage com o TecnoHelp pelo WhatsApp (ex: "Como resolver erro X no equipamento Y?" ou “Forneça o diagrama elétrico do equipamento Z”) e o chatbot retorna a documentação solicitada ou sugere soluções baseadas em documentos técnicos armazenados no banco de dados vetorial ou quando necessário, permite escalonar para suporte humano. Essa funcionalidade está descrita na FIGURA 5.

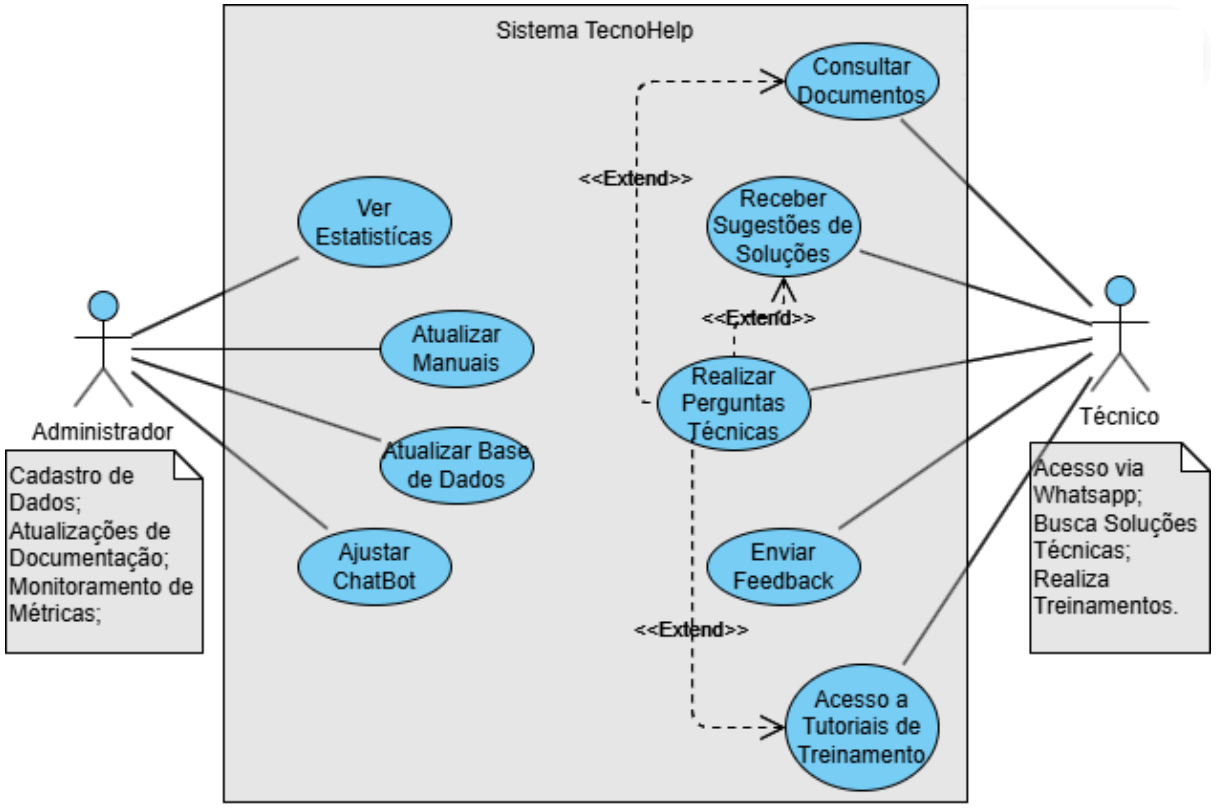
- **Centralização de Documentos:** Acesso unificado às informações contidas em manuais, procedimentos e FAQs, armazenados e organizados em um banco de dados vetorial (Chroma DB) através de processos como “tokenização” e “*embeddings*”.
- **Gerenciamento por Administradores:** Os Administradores realizam o cadastro e atualização dos documentos técnicos, além de monitorar as interações e permitir ações como: ver estatísticas do chatbot e do sistema como um todo, atualizar manuais, atualizar base de dados e ajustar o chatbot.

- **Escalonamento para Suporte Humano:** Se o chatbot não resolver, o técnico pode solicitar ajuda humana diretamente pela plataforma. Essa funcionalidade está inclusa na FIGURA 5.

3.1.1 FLUXOGRAMA DE FUNCIONALIDADES

Através da FIGURA 2 que se encontra a seguir, é possível ter uma visão geral das funcionalidades do sistema.

FIGURA 2 - Diagrama de casos de uso



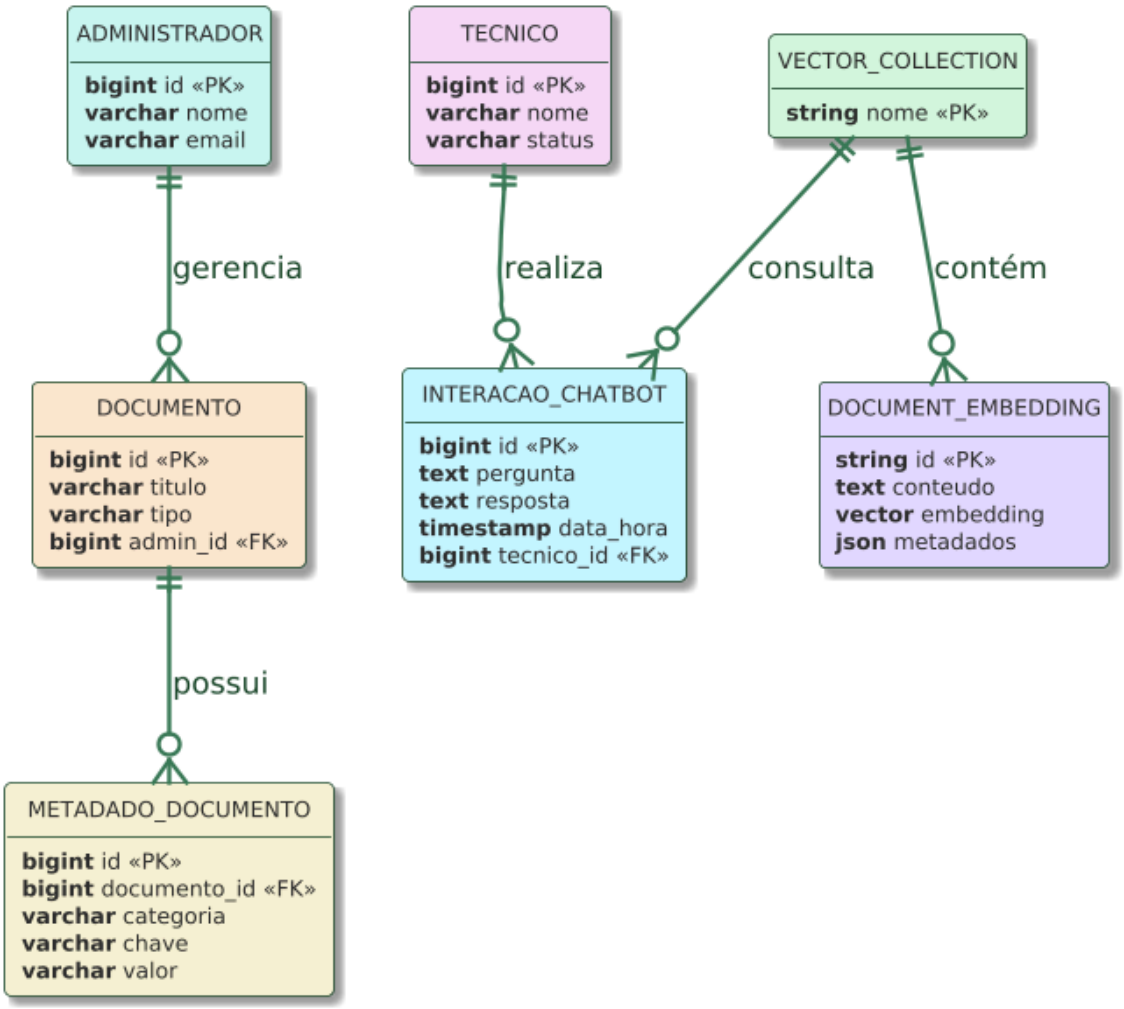
Fonte: Elaborado pelos autores.

O sistema possui dois atores, Técnico e Administrador. Suas principais funções incluem consultas técnicas (com acesso a documentos, sugestões e tutoriais), envio de feedback, gerenciamento da base de dados (estatísticas, atualização de manuais e ajustes no chatbot) e centralização de documentos. A relação «extend» indica funções opcionais, como solicitar suporte humano.

3.1.2 FLUXOGRAMA DO BANCO DE DADOS

Na FIGURA 3, temos o detalhamento das tabelas do Banco de Dados e das conexões entre as diferentes entidades.

FIGURA 3 - Diagrama de relacionamento



Fonte: Elaborado pelos autores.

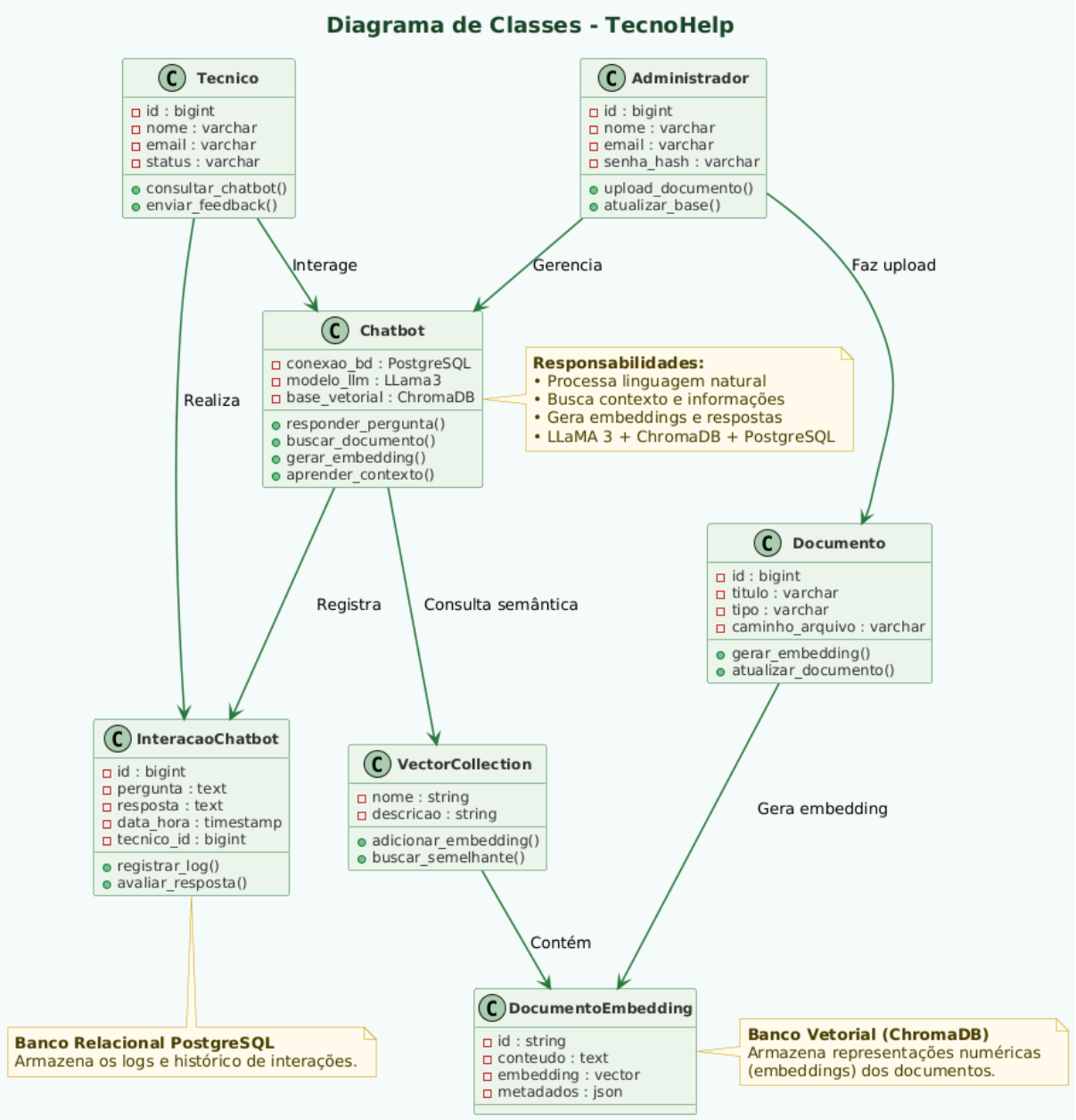
.

O diagrama representa a estrutura das principais entidades do sistema e como elas se conectam entre si. Na imagem, PK indica uma chave primária, FK indica uma chave estrangeira e Enum corresponde a valores restritos, como o status do técnico. As tabelas centrais do modelo são TÉCNICO, DOCUMENTO e INTERACAO_CHATBOT, que organizam os dados referentes aos usuários técnicos, aos documentos disponíveis e às interações realizadas com o chatbot.

3.1.3 FLUXOGRAMA DAS CLASSES DO SISTEMA

Na FIGURA 4, temos um diagrama que detalha e permite observar a estrutura estática do sistema.

FIGURA 4 - Diagrama de classe



Fonte: Elaborado pelos autores.

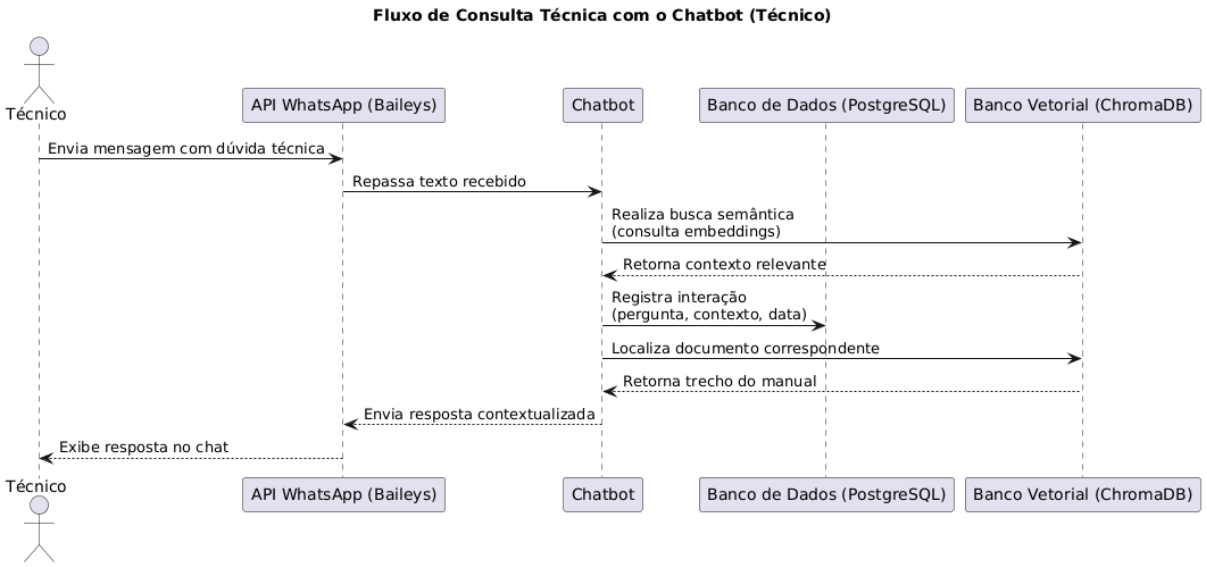
As interações são representadas pela entidade “InteracaoChatbot”, responsável por registrar perguntas, respostas e avaliações. O banco vetorial é composto por “DocumentoEmbedding”, que armazena *embeddings*, e “VectorCollection”, que organiza coleções semânticas. Entre os principais atributos

estão: id (chave primária), nome, email, tipo, caminho_arquivo, pergunta, resposta e *embedding*. Os métodos previstos incluem “consultar_chatbot()”, “upload_documento()”, “responder_pergunta()”, “buscar()” e “feedback()”. Os relacionamentos mapeados no modelo são “Gerencia”, “Consulta”, “Registra”, “Realiza”, “Gera embedding” e “Contém”.

3.1.4 FLUXOGRAMA DE UMA CONSULTA TÉCNICA

Na FIGURA 5, temos uma representação do fluxo de uma consulta técnica visualizado através do Diagrama de Sequência.

FIGURA 5 - Diagrama de consulta técnica



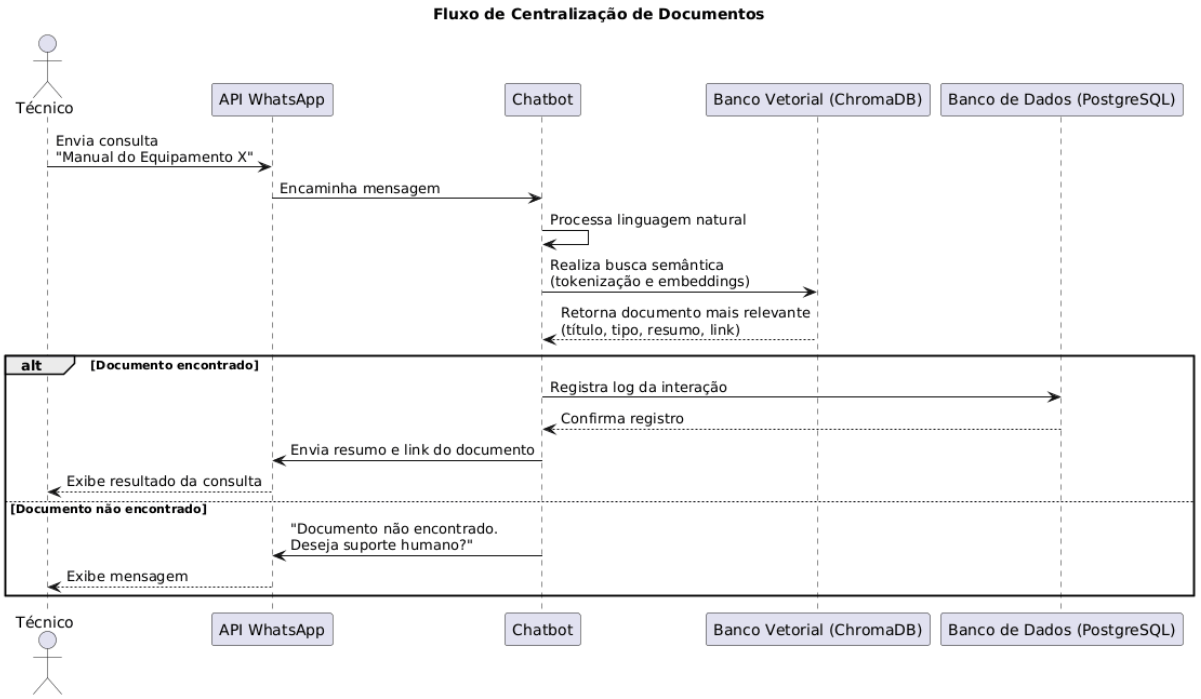
Fonte: Elaborado pelos autores.

O fluxo de uma consulta técnica funciona da seguinte forma: o técnico envia uma pergunta pelo WhatsApp, o chatbot realiza uma busca no banco de dados e, em seguida, retorna uma solução ou encaminha a solicitação para atendimento humano. No diagrama, as barras verticais representam o tempo de processamento e as setas numeradas indicam a ordem das ações.

3.1.5 FLUXOGRAMA DA CENTRALIZAÇÃO DE DOCUMENTOS

Na FIGURA 6, temos uma representação do fluxo da centralização de documentos visualizado através do Diagrama de Sequência.

FIGURA 6 - Diagrama de centralização de documentos



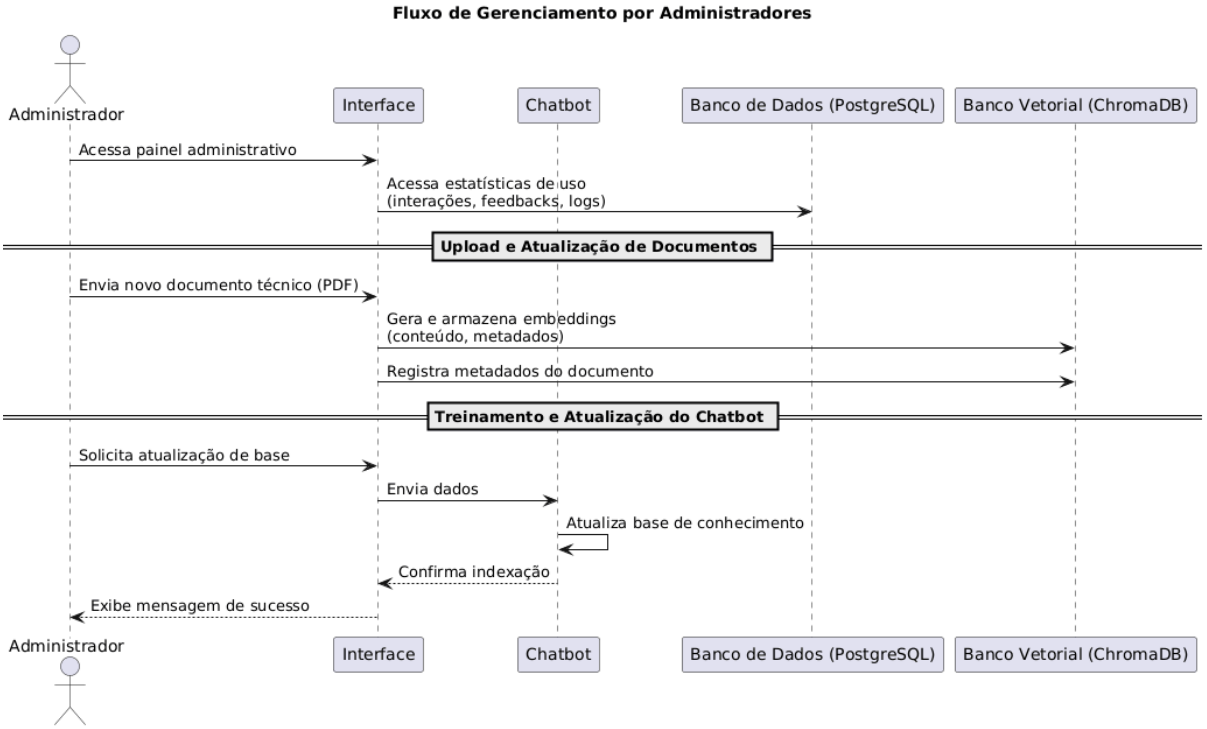
Fonte: Elaborado pelos autores.

No fluxo de centralização de documentos, o técnico solicita um manual por meio do WhatsApp, o chatbot consulta o banco de dados e o sistema de arquivos e, em seguida, retorna o documento ou sugere suporte humano. As barras verticais representam a ativação de cada componente durante o processo, enquanto as setas sólidas mostram o fluxo principal de comunicação. Além disso, blocos *alt* indicam decisões condicionais no fluxo.

3.1.6 FLUXOGRAMA DO GERENCIAMENTO DE ADMINISTRADORES

Na FIGURA 7, temos uma representação do fluxo da gestão dos administradores visualizado através do Diagrama de Sequência.

FIGURA 7 - Diagrama de gerenciamento de administradores



Fonte:Elaborado pelos autores.

No fluxo de gerenciamento realizado pelos administradores, o administrador acessa o painel de controle, o sistema permite a visualização de logs de registro e o chatbot é atualizado com novos conhecimentos. Nesse processo, um loop destaca operações repetitivas, como o envio de múltiplos arquivos; as setas tracejadas representam respostas ou confirmações; a ativação prolongada nas barras verticais indica processamento contínuo; e o alinhamento vertical evidencia a sequência temporal exata das ações.

4. FERRAMENTAS

Para a implementação do sistema proposto, foram pesquisadas diversas alternativas tecnológicas para cada um dos blocos funcionais do projeto. A escolha inicial considerou fatores como compatibilidade, robustez, curva de aprendizado e aderência às necessidades do suporte técnico remoto. Contudo, durante a fase de desenvolvimento e prototipagem, algumas dessas escolhas foram reavaliadas e substituídas por tecnologias mais modernas e eficientes, que se mostraram mais alinhadas aos objetivos de performance e qualidade da aplicação. A seguir, detalha-se essa evolução, justificando as mudanças realizadas.

O pilar do sistema, o motor de interpretação, representa a principal evolução técnica do aplicativo. O plano original previa o uso da biblioteca ChatterBot em Python. A escolha se baseava em sua natureza de código aberto, facilidade de implementação e integração com bancos de dados relacionais. No entanto, durante os testes práticos, percebeu-se que uma abordagem baseada em padrões de conversação, como a do ChatterBot, seria limitada para a complexidade das dúvidas técnicas. Assim, tomou-se a decisão estratégica de substituir o ChatterBot por um modelo de linguagem de última geração. Após uma análise das limitações de cada modelo, a tecnologia escolhida foi o modelo LLaMA 3 70B, acessado através da API da Groq. Essa mudança elevou drasticamente o potencial do sistema, trocando um modelo simples por um modelo capaz de compreender contextos complexos e gerar respostas muito mais precisas e humanas. A viabilidade dessa mudança foi garantida pelo desempenho e pelo baixo custo oferecido pela API da Groq, superando as barreiras financeiras e de complexidade que antes limitavam o uso de soluções de ponta como as da Google ou IBM.

A arquitetura da base de conhecimento e do banco de dados também foi refinada. Inicialmente, planejava-se usar um banco de dados relacional (PostgreSQL ou MySQL) como um repositório central para manuais, tutoriais e históricos de atendimento. Na implementação final, essa ideia foi especializada: o PostgreSQL foi efetivamente o escolhido, com sua função focada no armazenamento do histórico de interações (perguntas, respostas, datas e usuários) para fins de auditoria e análise.

A base de conhecimento, por sua vez, foi desacoplada do banco de dados relacional e passou a ser processada e armazenada no banco vetorial ChromaDB.

Para isso, os arquivos PDF são lidos e convertidos em texto por meio da biblioteca PyPDF2, que realiza a extração das informações brutas de cada manual técnico. Esses textos são então divididos em blocos, tokenizados e transformados em *embeddings* com o modelo "*SentenceTransformer*", sendo finalmente indexados no ChromaDB. Essa abordagem se mostrou mais eficaz para lidar com altos volumes de dados e permite que a equipe atualize a base de conhecimento do sistema simplesmente adicionando ou substituindo arquivos PDF, sem a necessidade de intervir na estrutura do banco de dados.

Em relação à interface com o usuário, o plano original era desenvolver o sistema diretamente sobre a API do WhatsApp, visando aproveitar sua ampla adoção no ambiente corporativo. Contudo, durante o desenvolvimento, percebeu-se que seria mais produtivo adotar uma abordagem em fases. Primeiramente, foi criada uma interface web com o *framework* Flask, utilizando HTML, CSS e JavaScript. Este protótipo web foi fundamental para validar e refinar o motor de IA de forma rápida e controlada. O objetivo final, no entanto, permanece o mesmo: integrar a solução ao WhatsApp. O planejamento atual, agora inclui o uso da API "Baileys", que permite gerenciar o fluxo de mensagens, garantindo um funcionamento estável e escalável para a integração final.

Em resumo, a arquitetura final do protótipo, baseada em Python, Groq (LLaMA 3), PostgreSQL, Chroma DB e PyPDF2, é um reflexo da evolução e aprendizado durante o ciclo de vida do aplicativo. A transição de um sistema baseado em ChatterBot e com integração direta ao WhatsApp para uma arquitetura com um LLM de ponta, base de conhecimento desacoplada e um plano de implantação em fases, demonstra uma adaptação estratégica em busca de uma solução final mais poderosa, moderna e eficiente.

5. ARQUITETURA, IMPLEMENTAÇÃO E USO

O sistema TecnoHelp foi desenvolvido com arquitetura modular, composta por quatro componentes principais que trabalham de forma integrada para oferecer suporte técnico inteligente via chatbot. Esta seção detalha cada módulo, apresentando suas interfaces, implementações e fluxos de processamento.

5.1 MÓDULO INTERFACE

Este módulo constitui a camada de interação direta com os técnicos e administradores e é implementada através de três interfaces complementares (Acesso via WhatsApp, Plataforma Web e uma interface somente para administradores). A API do WhatsApp (Baileys) permite o atendimento imediato e acessível, aproveitando a familiaridade dos usuários com a plataforma. Paralelamente, a interface web oferece características semelhantes, proporcionando flexibilidade no acesso ao sistema para usuários que preferirem optar pelo uso do navegador e, para os administradores, temos a interface que permite realizar a gestão do software, permitindo também a inclusão de documentos para expansão da base de conhecimento. Nas FIGURAS 8,9,10 e 11 temos um exemplo de como estão elaboradas as interfaces.

5.1.1 INTERFACE WHATSAPP

A interface via WhatsApp constitui o principal canal de interação entre os técnicos e o sistema, desenvolvida através da API Baileys para garantir atendimento imediato e acessível. Esta opção aproveita a familiaridade dos usuários com a plataforma, permitindo que os técnicos consultem o chatbot de forma natural e intuitiva, como se estivessem conversando com um colega de trabalho. A escolha do WhatsApp como canal prioritário elimina a necessidade de aprendizado de novas ferramentas, reduzindo a curva de adoção e facilitando o acesso remoto ao suporte especializado. Na FIGURA 8 é possível visualizar como está construída a interface via Whatsapp do chatbot.

FIGURA 8 - Interface whatsapp

Fonte: Elaborado pelos autores.

5.1.2 INTERFACE WEB

A interface web representa uma alternativa versátil ao canal do WhatsApp, desenvolvida para oferecer flexibilidade de acesso aos técnicos que preferem utilizar navegadores web ou necessitam de uma opção complementar ao aplicativo de mensagens. Esta plataforma mantém paridade funcional completa com a versão mobile, assegurando a mesma qualidade de interação e todos os recursos do chatbot, o que garante uma experiência consistente independentemente do canal escolhido pelo usuário. Como solução redundante, a interface web garante a continuidade do suporte técnico em diversos cenários operacionais, sendo particularmente útil em ambientes corporativos com restrições ao uso do WhatsApp ou para técnicos que se sentem mais confortáveis com plataformas baseadas em navegador. Conforme ilustrado na FIGURA 9, a interface web replica integralmente as funcionalidades disponíveis via WhatsApp, permitindo que os usuários aproveitem todo o potencial do sistema de forma adaptável às suas preferências e necessidades específicas.

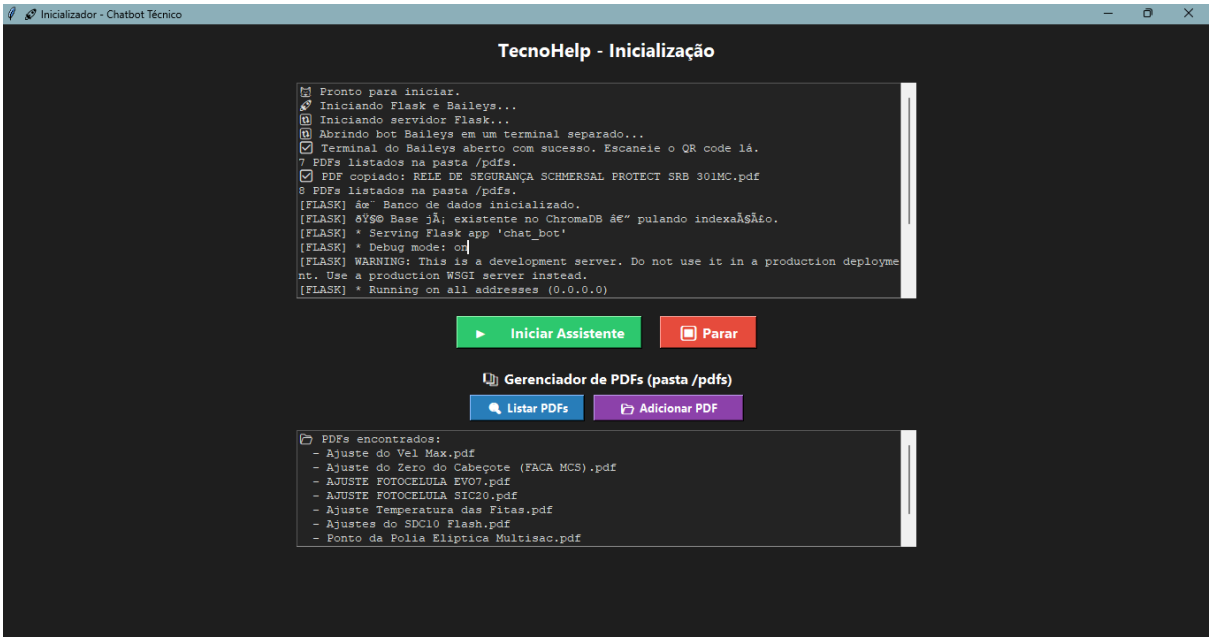
FIGURA 9 - Interface web

Fonte: Elaborado pelos autores.

5.1.3 INTERFACE ADMINISTRATIVA (GESTÃO DO CONHECIMENTO)

A interface administrativa fornece ferramentas completas para a gestão do sistema e expansão da base de conhecimento. Através deste painel, os administradores podem monitorar o funcionamento do chatbot, acompanhar logs de inicialização, identificar possíveis erros em tempo real e realizar a indexação de novos documentos técnicos. A plataforma permite o upload organizado de manuais, tutoriais e procedimentos, que são automaticamente processados e incorporados ao banco de dados vetorial para enriquecimento das respostas. Além disso, inclui funcionalidades de auditoria que listam todos os arquivos indexados, proporcionando controle total sobre o conteúdo disponível para os técnicos. Nesta imagem é visível a interface dos administradores.

FIGURA 10 - Interface administradores



Fonte: Elaborado pelos autores.

5.2 MÓDULO BANCO DE DADOS RELACIONAL

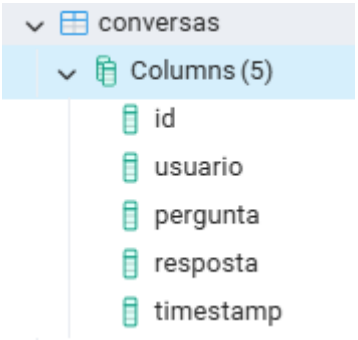
Responsável pelo armazenamento de dados estruturados do sistema, incluindo logs de interações, histórico de conversas e metadados de usuários. Garante a persistência confiável de informações operacionais e fornece base para análises de desempenho e métricas de uso. Nas FIGURAS 10 e 11, temos dois exemplos de como está sendo estruturado o banco de dados relacional.

FIGURA 11 - Inicialização banco de dados sql

```python
def get_db_conn():
    return psycopg2.connect(**DB_CONFIG)

def inicializar_banco():
    try:
        conn = get_db_conn()
        cur = conn.cursor()
        cur.execute("""
            CREATE TABLE IF NOT EXISTS conversas (
                id SERIAL PRIMARY KEY,
                usuario VARCHAR(50),
                pergunta TEXT,
                resposta TEXT,
                timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            );
        """)
        conn.commit()
        cur.close()
        conn.close()
```

Fonte: Elaborado pelos autores.

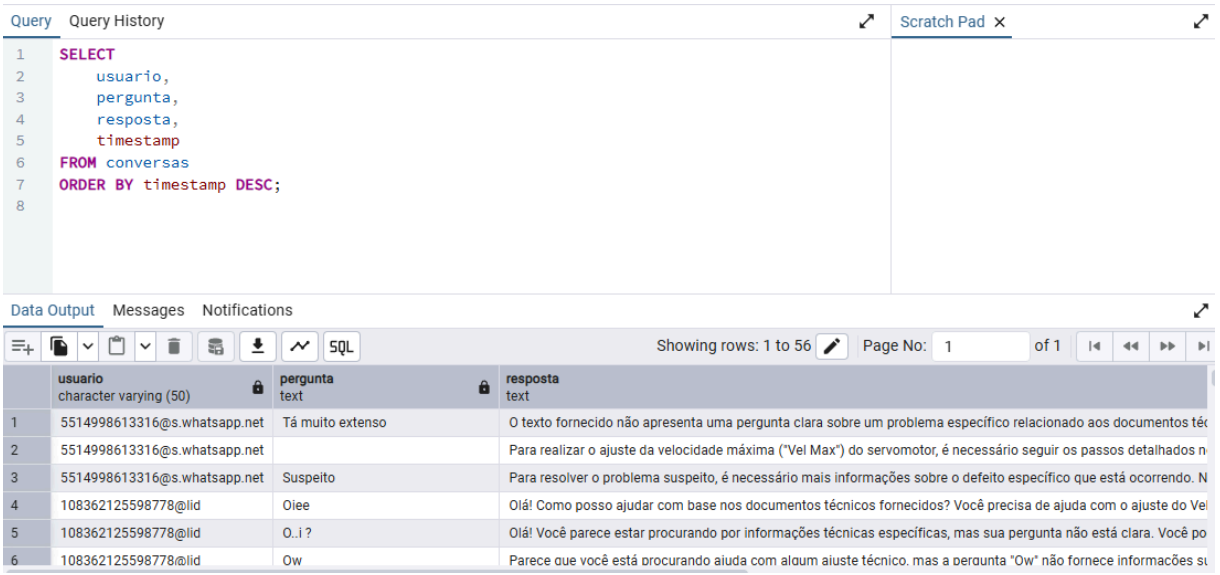
FIGURA 12 - Estrutura banco de dados sql



Fonte: Elaborado pelos autores.

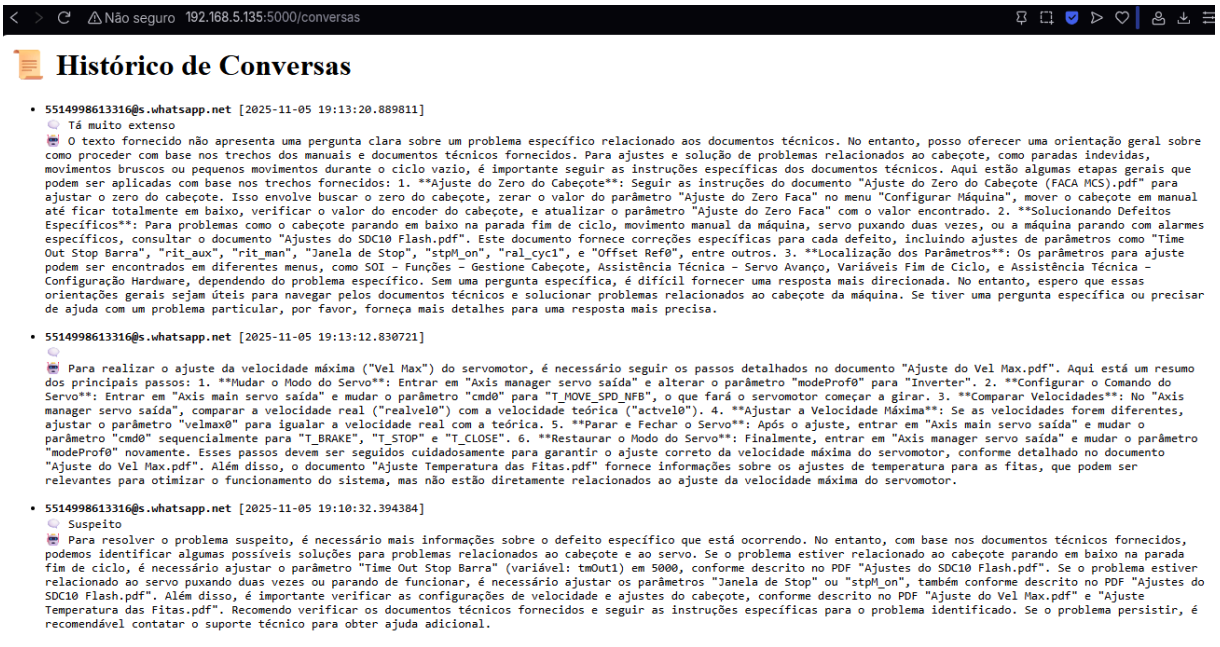
Nas FIGURAS 13 e 14, temos uma consulta ao banco de dados relacional que permite recuperar dados e logs de histórico das conversas.

FIGURA 13 - Query banco de dados sql



Fonte: Elaborado pelos autores.

FIGURA 14 - Histórico de conversas interface web



Fonte: Elaborado pelos autores.

5.3 MÓDULO BANCO DE DADOS VETORIAL (APRENDIZAGEM CONTÍNUA)

Módulo especializado no armazenamento e recuperação de "*embeddings* vetoriais", representações numéricas do conhecimento técnico. Permite buscas semânticas eficientes, encontrando documentos similares mesmo com variações terminológicas. O Chroma DB organiza os manuais técnicos em espaços vetoriais

multidimensionais para recuperação precisa. Nas FIGURAS 13, 14, 15 e 16, temos um exemplo de como estão implementados no aplicativo.

FIGURA 15 - Inicialização banco de dados vetorial

```
# === Inicializa modelo de embeddings e banco vetorial Chroma ===
modelo_embeddings = SentenceTransformer("all-MiniLM-L6-v2")
chroma_client = chromadb.PersistentClient(path="chroma_db")
colecao = chroma_client.get_or_create_collection("documentos_pdf")

# ===------------------------------------------------------------===
```

Fonte: Elaborado pelos autores.

FIGURA 16 - Extração de textos do pdf

```
def extrair_texto_pdf(caminho_pdf):
    """Extrai texto puro de um PDF."""
    try:
        with open(caminho_pdf, "rb") as f:
            leitor = PyPDF2.PdfReader(f)
            texto = ""
            for pagina in leitor.pages:
                texto_pagina = pagina.extract_text()
                if texto_pagina:
                    texto += texto_pagina + "\n"
            return texto.strip()
    except Exception as e:
        print(f"Erro ao ler {caminho_pdf}: {e}")
        return ""
```

Fonte: Elaborado pelos autores.

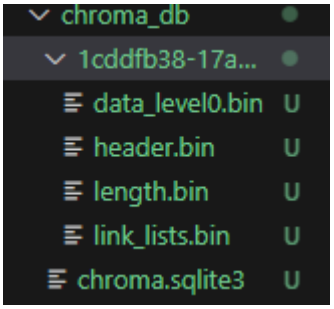
FIGURA 17 - Indexação de documentos no bd vetorial

```python
def carregar_pdfs_para_chroma(pasta="pdfs"):
    """Extrai texto dos PDFs e armazena embeddings no ChromaDB."""
    print("📂 Verificando PDFs para indexação...")

    # PDFs presentes na pasta
    arquivos = [f for f in os.listdir(pasta) if f.lower().endswith(".pdf")]

    # Carrega lista de PDFs já indexados
    if os.path.exists(INDEX_LOG):
        with open(INDEX_LOG, "r", encoding="utf-8") as f:
            indexados = set(json.load(f))
    else:
        indexados = set()

    novos_pdfs = [f for f in arquivos if f not in indexados]

    if not novos_pdfs:
        print("🧩 Nenhum PDF novo detectado — mantendo base atual.")
        return

    print(f"📘 {len(novos_pdfs)} novos PDFs detectados. Iniciando indexação...\n")

    for nome_arquivo in novos_pdfs:
        caminho = os.path.join(pasta, nome_arquivo)
        texto = extrair_texto_pdf(caminho)
        if not texto:
            continue

        blocos = [texto[i:i+1000] for i in range(0, len(texto), 1000)]
        embeddings = modelo_embeddings.encode(blocos).tolist()
        ids = [f"{nome_arquivo}_{i}" for i in range(len(blocos))]

        colecao.add(
            ids=ids,
            documents=blocos,
            metadatas=[{"arquivo": nome_arquivo}] * len(blocos),
            embeddings=embeddings
        )

        indexados.add(nome_arquivo)
        print(f"✅ {nome_arquivo} indexado ({len(blocos)} blocos).")

    # Atualiza o arquivo de controle
    with open(INDEX_LOG, "w", encoding="utf-8") as f:
        json.dump(list(indexados), f, ensure_ascii=False, indent=2)

    print("🧠 Base de conhecimento atualizada com sucesso!")
```

Fonte: Elaborado pelos autores.

Na FIGURA 18, é possível observar os arquivos gerados pelo banco de dados vetorial após a indexação da documentação na base de conhecimento e processamento e geração dos *embeddings*.

FIGURA 18 - Estrutura de arquivos do bd vetorial



Fonte: Elaborado pelos autores.

5.4 MÓDULO MOTOR DE INTERPRETAÇÃO

Esse é o núcleo inteligente do sistema, este módulo implementa a arquitetura RAG utilizando o modelo LLama 3. Responsável por compreender consultas técnicas, buscar informações relevantes na base de conhecimento e gerar respostas contextualizadas. O processamento inclui análise semântica, recuperação de documentos e síntese de respostas. Nesse trecho de código abaixo (FIGURA 17) temos a função responsável por realizar a busca semântica no banco de dados vetorial.

FIGURA 19 - Função de busca de contexto

```python
def buscar_contexto(pergunta, top_k=3):
    """Busca os trechos mais relevantes no ChromaDB."""
    embedding_pergunta = modelo_embeddings.encode([pergunta]).tolist()[0]
    resultados = colecao.query(query_embeddings=[embedding_pergunta], n_results=top_k)
    textos = resultados["documents"][0]
    metadados = resultados["metadatas"][0]

    contexto = ""
    for t, m in zip(textos, metadados):
        contexto += f"PDF: {m['arquivo']}\nTrecho: {t}\n\n"

    return contexto.strip() if contexto else "Nenhuma informação relevante encontrada."
```

Fonte: Elaborado pelos autores.

Na FIGURA 20, temos a função responsável por gerar as respostas do chatbot, baseado na API Groq e utilizando o modelo de LLM Llama 3, temos todo o processamento da linguagem natural, do contexto e, integrando com a busca semântica permite a LLM gerar as respostas ao usuário.

FIGURA 20 - Função de respostas

```python
def gerar_resposta_groq(pergunta, contexto):
    """Envia prompt contextualizado para o modelo Groq."""
    prompt = (
        f"O usuário perguntou:\n{pergunta}\n\n"
        f"Abaixo estão trechos dos manuais e documentos técnicos:\n{contexto}\n\n"
        "Com base nesses documentos, responda da forma mais útil e técnica possível, "
        "lembre-se de ser breve e direto citando o nome do PDF quando relevante."
    )

    resposta = requests.post(
        "https://api.groq.com/openai/v1/chat/completions",
        headers={
            "Authorization": f"Bearer {GROQ_API_KEY}",
            "Content-Type": "application/json"
        },
        json={
            "model": GROQ_MODEL,
            "messages": [{"role": "user", "content": prompt}],
            "temperature": 0.7,
            "max_tokens": 1024
        }
    )

    try:
        data = resposta.json()
        return data["choices"][0]["message"]["content"].strip()
    except Exception as e:
        print("Erro ao obter resposta da Groq:", e)
        return "⚠️ Erro ao gerar resposta."

# ===Flask===

@app.route("/mensagem", methods=["POST"])
def receber_mensagem():
    dados = request.get_json()
    usuario = dados.get("usuario", "Desconhecido")
    pergunta = dados.get("mensagem", "")

    contexto = buscar_contexto(pergunta)
    resposta = gerar_resposta_groq(pergunta, contexto)
```

Fonte: Elaborado pelos autores.

6. PROCEDIMENTOS DE VALIDAÇÃO DO APLICATIVO

A validação da aplicação foi conduzida através de uma abordagem estruturada de testes, com o objetivo de garantir a funcionalidade, confiabilidade e desempenho do chatbot de suporte técnico. Para isso, foram implementadas duas metodologias complementares: testes de caixa branca, focados na avaliação interna dos componentes do sistema, e testes de caixa preta, destinados a verificar o comportamento externo da aplicação em condições reais de operação.

Os testes de caixa branca consistiram na avaliação de cada módulo que compõe a arquitetura do sistema. Na interface de usuário, desenvolvida com a API do WhatsApp, foram realizadas verificações quanto à capacidade de recepção e envio de mensagens, assegurando que os tempos de resposta permanecessem dentro de parâmetros aceitáveis para uma boa experiência de conversação. No módulo de interpretação, implementado com o motor Llama 3, a validação concentrou-se na precisão e relevância das respostas geradas, com ênfase na contextualização adequada das informações e na coerência das soluções apresentadas para os problemas técnicos reportados. O banco de dados PostgreSQL passou por testes de integridade, que confirmaram a consistência dos dados armazenados. O módulo de aprendizado contínuo foi submetido a ciclos iterativos de inserção de novos conteúdos técnicos, seguidos da verificação da capacidade do sistema em assimilar e incorporar esse conhecimento em respostas futuras.

Complementarmente, os testes de caixa preta simularam cenários adversos que poderiam ocorrer em situações operacionais. Foram reproduzidos cenários de instabilidade de rede, incluindo falhas temporárias de conexão, para avaliar a resiliência do sistema. Também foram submetidas mensagens com diferentes níveis de ambiguidade, terminologia incorreta ou com baixa coerência, além de comandos fora do escopo esperado, com o objetivo de verificar a capacidade do chatbot em lidar adequadamente com entradas inesperadas. Em todos esses cenários, o sistema demonstrou a capacidade de manter a estabilidade operacional e de fornecer respostas de acordo com o esperado, mantendo o contexto, mesmo quando confrontado com condições não ideais de operação.

7. CONCLUSÃO

O aplicativo teve como objetivo, ser uma solução inteligente voltada ao suporte técnico, esse assistente técnico foi concebido para solucionar dois desafios centrais: tempo excessivo na capacitação de novos técnicos e a descentralização de documentos e manuais técnicos. A solução integrou um chatbot atuando como assistente técnico virtual, responsável por centralizar informações essenciais e fornecer respostas automatizadas, contribuindo diretamente para maior agilidade e precisão nos atendimentos.

Entre os resultados visados, destacavam-se a redução no tempo de treinamento de novos profissionais e a otimização no atendimento de chamados técnicos. Esperava-se que essa melhoria decorresse do acesso facilitado às informações por meio de um sistema unificado, o que potencialmente elevaria a qualidade dos serviços prestados pelas equipes de suporte. Essa abordagem refletiu o conceito de organização Hipertexto proposto por Nonaka e Takeuchi (2008), em que a reclassificação e disseminação do conhecimento em uma base acessível tornam-se fundamentais para o aprimoramento contínuo da eficiência organizacional. Assim, o aplicativo desenvolvido buscou contribuir para reduzir o tempo de inatividade de equipamentos e aprimorar a experiência do cliente, fortalecendo a imagem da empresa ao promover uma gestão ativa do conhecimento como fonte de vantagem competitiva.

Durante o desenvolvimento, enfrentamos desafios significativos, especialmente na integração tecnológica. A escolha e a compatibilidade das APIs exigiram diversas adaptações, assim como a estruturação do banco de dados relacional e vetorial, que demandou ajustes para garantir desempenho e consistência entre as consultas do chatbot e os registros armazenados. Além disso, o versionamento da aplicação trouxe dificuldades no controle de alterações em equipe, exigindo organização nas ramificações e fusões do código.

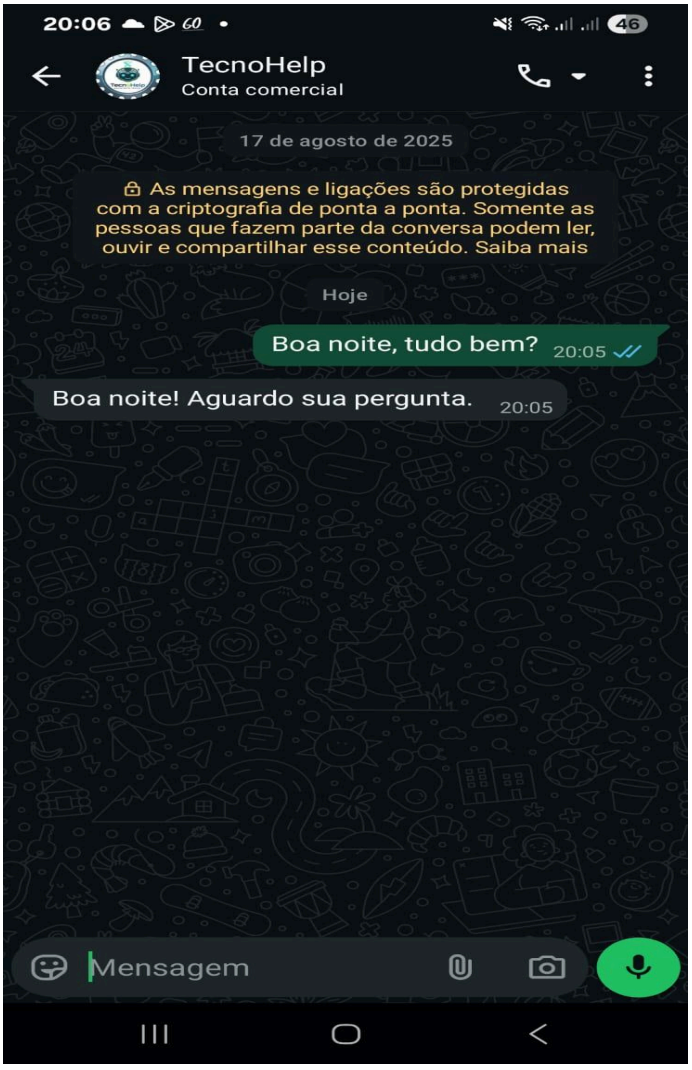
Como tal, foram necessárias diversas adaptações e mudanças de escolha ao longo do projeto, como a troca da biblioteca “Chatterbot” para o uso de um modelo de LLM, além de outros itens como o próprio cronograma em si que em mais de um momento provou que havia subestimado certos cenários no desenvolvimento do aplicativo, como os prazos para desenvolvimento e integração da interface Whatsapp com o módulo do motor de interpretação. Também foi necessário

armazenar a aplicação num repositório remoto, no caso o GitHub, em decorrência de problemas que ocorreram como a perda de artefatos, trechos de código e versões do projeto, inclusive um acontecimento fatídico que foi o fato de um Solid State Drive (SSD) corromper, o que acarretou na perda de grande parte do trabalho armazenado nele até aquele momento. Após isso, recuperamos o que podíamos de diferentes fontes e reconstruímos o código principal dessa vez realizando o versionamento de cada alteração. Isso também permitiu que testássemos diferentes tecnologias de forma mais organizada, sem afetar ou perder trechos de códigos que anteriormente funcionavam.

Apesar desses obstáculos, a implementação mostrou-se viável e eficiente, uma vez que utilizou tecnologias acessíveis, manteve custos controlados e atendeu a uma necessidade do mercado, evidenciada nos objetivos centrais da aplicação. O sistema resultante demonstrou potencial de aplicação prática em empresas e organizações, por conta de seu baixo custo de implantação e manutenção. Por fim, existem certos pontos que poderiam ser melhorados em aplicações futuras, como o teste de LLMs instaladas localmente no servidor da aplicação (o que demandaria um hardware mais potente), além de metodologias testes diferentes ou mais elaboradas que poderiam ser usadas no âmbito do aplicativo. A seguir, nas FIGURAS 21, 22 e 23 são demonstradas algumas interações entre o usuário e a aplicação desenvolvida.

Na FIGURA 21 é possível observar um exemplo de interação na qual o usuário realiza uma saudação ou cumprimento ao iniciar uma conversa com o chatbot.

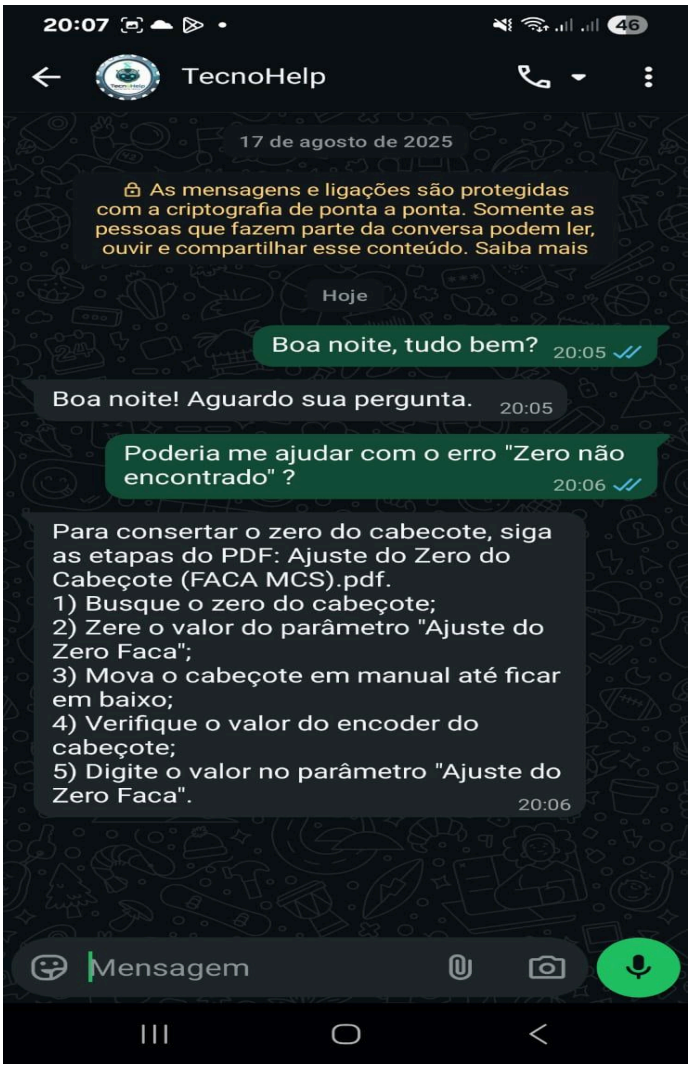
FIGURA 21 - Exemplo de saudação



Fonte: Elaborado pelos autores.

Na FIGURA 22 é possível visualizar a aplicação respondendo o usuário quanto a uma dúvida técnica.

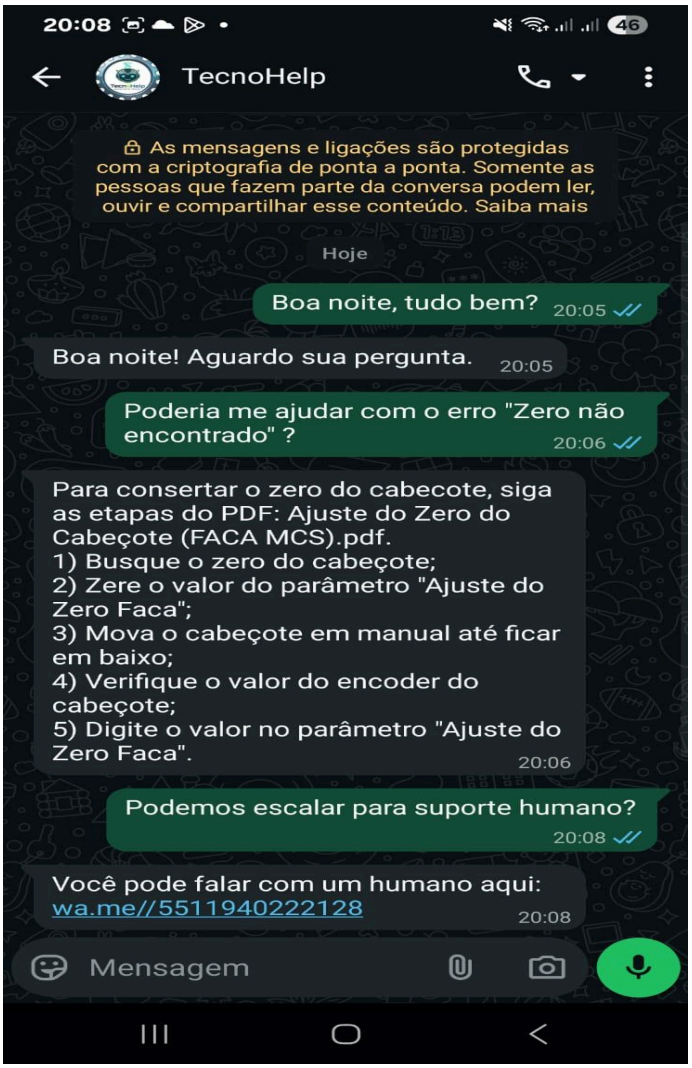
FIGURA 22 - Realizando uma pergunta



Fonte: Elaborado pelos autores.

Na FIGURA 23, o usuário solicita a aplicação que escale para suporte humano e, o chatbot por sua vez responde com um link que conecta o usuário diretamente a um contato predefinido.

FIGURA 23 - Escalonando para suporte humano



Fonte: Elaborado pelos autores.

8. REFERÊNCIAS

ALMEIDA, Marcus Vinícius de Lara. Aplicação dos princípios Lean em um setor de suporte técnico de uma indústria do ramo tecnológico. 2017. 52 f. Trabalho de Conclusão de Curso (Tecnologia em Sistemas de Telecomunicações) — Universidade Tecnológica Federal do Paraná, Curitiba, 2017. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/9720. Acesso em: 3 abr. 2025.

AMAZON WEB SERVICES (AWS). O que é geração aumentada por recuperação (RAG)? [S. l.], 2025. Disponível em: https://aws.amazon.com/pt/what-is/retrieval-augmented-generation/. Acesso em: 23 out. 2025.

CHROMA. Chroma: open-source search and retrieval database for AI applications. [S. l.], 2025. Disponível em: https://www.trychroma.com/. Acesso em: 9 nov. 2025.

GOOGLE. Conversational AI. [S. l.], 2025. Disponível em: https://cloud.google.com/products/conversational-agents#key-features. Acesso em: 30 jun. 2025.

GROQ. Groq. [S. l.], 2025. Disponível em: https://groq.com/. Acesso em: 1 jul. 2025.

GURU. ChatGuru: plataforma de atendimento automatizado. São Paulo, 2023. Disponível em: https://chatguru.com.br/. Acesso em: 3 abr. 2025.

IBM. IBM watsonx Assistant. [S. l.], 2025. Disponível em: https://www.ibm.com/br-pt/products/watsonx-assistant. Acesso em: 30 jun. 2025.

META. Meta-Llama-3-70B. [S. l.]: Hugging Face, 2025. Disponível em: https://huggingface.co/meta-llama/Meta-Llama-3-70B. Acesso em: 1 jul. 2025.

N8N. Workflow automation for technical people. [S. l.], 2025. Disponível em: https://n8n.io/. Acesso em: 1 jul. 2025.

NONAKA, Ikujiro; TAKEUCHI, Hirotaka. Gestão do conhecimento. Porto Alegre: Bookman, 2008.

ORACLE. MySQL. [S. l.], 2025. Disponível em: https://www.mysql.com/. Acesso em: 7 abr. 2025.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL. [S. l.], 2025. Disponível em: https://www.postgresql.org/. Acesso em: 7 abr. 2025.

PRODESP. CS-Chatbot: solução para atendimento técnico. São Paulo, 2022. Disponível em: https://solucoes.prodesp.sp.gov.br/cs-chatbot/. Acesso em: 3 abr. 2025.

PYTHON SOFTWARE FOUNDATION. Python. [S. l.], 2025. Disponível em: https://www.python.org/. Acesso em: 7 abr. 2025.

RASAHQ. Rasa: build the next generation of AI assistants. [S. l.], 2025. Disponível em: https://rasa.com/. Acesso em: 30 jun. 2025.

ZENDESK. Zendesk for Service: suporte técnico especializado. California, 2023. Disponível em: https://www.zendesk.com.br/lp/brand/. Acesso em: 3 abr. 2025.