aurum
Revista Multidisciplinar

ISSN: 3085-7783

DESENVOLVIMENTO DE UM CHATBOT COM API PARA OTIMIZAÇÃO DO SUPORTE AO CLIENTE - NEXUSSUPPORT IA

DEVELOPMENT OF A CHATBOT WITH API FOR CUSTOMER SUPPORT OPTIMIZATION - NEXUSSUPPORT AI

https://doi.org/10.63330/armv1n10-002

Submetido em: 08/12/2025 e Publicado em: 12/12/2025

Erick Leandro Santos de Mesquita
Graduando em Engenharia da Computação
Centro Universitário do Norte – Uninorte
E-mail: erick.lear99@gmail.com

Ives Gomes Firmo
Graduando em Engenharia da Computação
Centro Universitário do Norte – Uninorte
E-mail: ivesfirmo@live.com

Jamisson Lebre Pereira
Graduando em Engenharia da Computação
Centro Universitário do Norte – Uninorte
E-mail: lebrejamisson@gmail.com

Roneuane Grazielle da Gama Araujo
Especialista em Engenharia e Administração de Banco de dados Oracle
Centro Universitário do Norte
E-mail: roneuanegrazielle@gmail.com

RESUMO
A crescente demanda por eficiência no suporte ao cliente impulsiona a adoção de soluções baseadas em Inteligência Artificial (IA). Diante deste cenário, este trabalho apresenta a concepção e implementação do NexusSupport IA, um chatbot de suporte automatizado que utiliza a API de Large Language Model (LLM) do Google Gemini como seu motor de inferência. A tese central é que a integração de um LLM de ponta pode reduzir o First Contact Resolution Time (FCR) e otimizar a gestão de chamados, elevando a qualidade da experiência do usuário. O sistema é estruturado para interagir com usuários, oferecendo soluções detalhadas, mantendo o contexto histórico da conversa (via gerenciamento de sessão) e, criticamente, realizando o Handoff Inteligente para técnicos especializados quando a complexidade do problema excede o escopo do modelo. A pesquisa abrange a Engenharia de Prompt, o desenvolvimento de sua interface de autenticação (HTML/CSS) e a tela de chat (frontend com React/JavaScript), comunicando-se diretamente com o serviço de backend que orquestra as chamadas para a API do Google. O projeto é estruturado com ferramentas modernas como Vite para build e NPM para gerenciamento de dependências.

Palavras-chave: Inteligência Artificial; Chatbot; Suporte ao Cliente; Automação; Google Gemini.

ABSTRACT
The growing demand for efficiency in customer support is driving the adoption of solutions based on Artificial Intelligence (AI). Given this scenario, this work presents the conception and implementation of NexusSupport IA, an automated support chatbot that utilizes the Google Gemini Large Language Model (LLM) API as its inference engine. The central thesis is that integrating a cutting-edge LLM can reduce the First Contact Resolution Time (FCR) and optimize ticket management, thus elevating the quality of the user experience. The system is structured to interact with users, offering detailed solutions, maintaining the historical context of the conversation (via session management), and, critically, performing Intelligent Handoff to specialized technicians when the problem's complexity exceeds the model's scope. The research encompasses Prompt Engineering, the development of its authentication interface (HTML/CSS) and the chat screen (frontend with React/JavaScript), communicating directly with the backend service that orchestrates calls to the Google API. The project is structured with modern web development tools like Vite for the build process and NPM for dependency management.

Keywords: Artificial Intelligence; Chatbot; Customer Support; Automation; Google Gemini.

1 INTRODUÇÃO

O avanço da Inteligência Artificial (IA) tem proporcionado soluções inovadoras para diversos setores (RUSSELL; NORVIG, 2022), incluindo o suporte ao cliente. Contemporaneamente, as empresas enfrentam desafios significativos na gestão eficiente do atendimento, visto que uma alta demanda pode resultar em atrasos e, consequentemente, na insatisfação dos usuários (HUANG; RUST, 2018).

Nesse contexto, os chatbots emergem como uma alternativa promissora para otimizar esse processo, oferecendo respostas rápidas e precisas, além de aliviarem a carga sobre as equipes de atendimento humano (ADAMOPOULOU; MOUSSIADES, 2020).

A adoção estratégica dessas tecnologias não apenas automatiza tarefas rotineiras, mas transforma a cadeia de valor do serviço ao cliente, permitindo que a inteligência humana foque em problemas complexos (DAVENPORT, 2018).

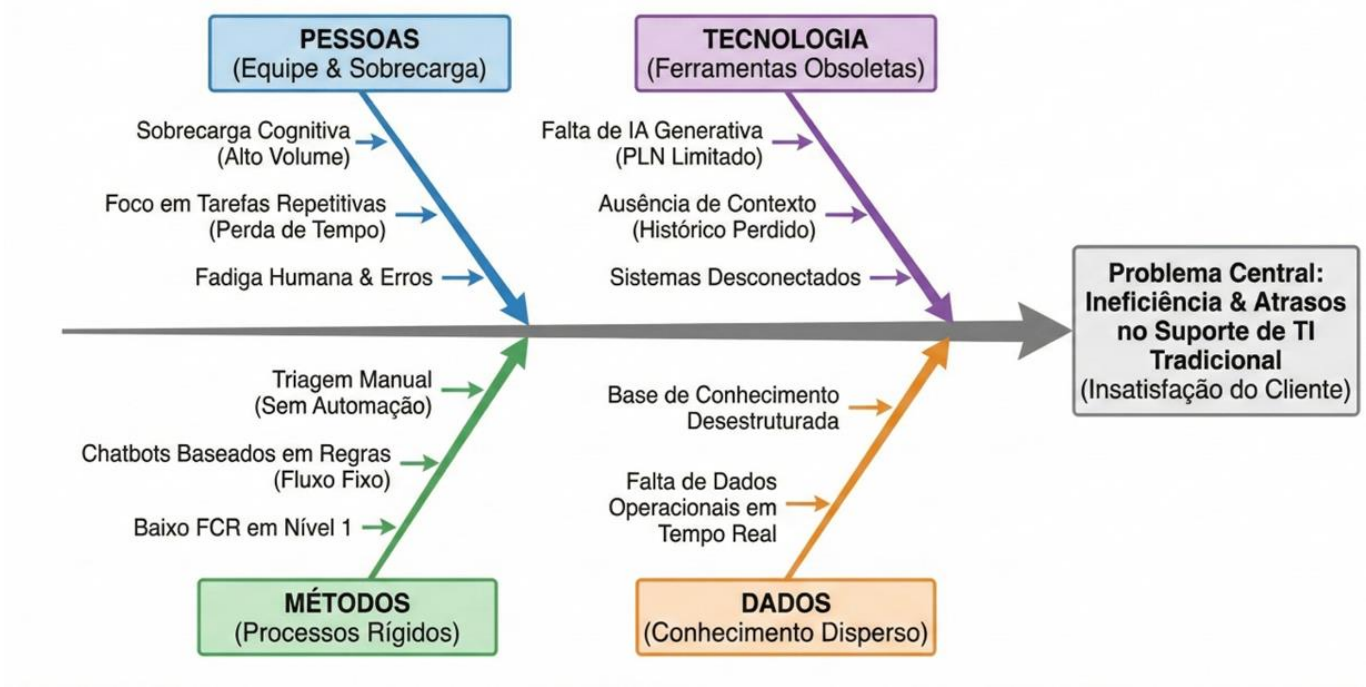
Apesar das vantagens inerentes à automação, existem limitações na capacidade de chatbots tradicionais interpretarem problemas de alta complexidade (HUANG; RUST, 2018), o que frequentemente exige a adoção de um modelo híbrido que combine suporte automatizado e intervenção humana qualificada (LUO et al., 2019).

As raízes da ineficiência no suporte de TI tradicional, que motivam o desenvolvimento desta solução, estão sintetizadas no Diagrama de Ishikawa apresentado na Figura 1.

Este mapeamento de causas permite identificar que gargalos operacionais e a alta rotatividade de pessoal comprometem a continuidade do conhecimento técnico dentro das organizações. Consequentemente, a implementação de sistemas inteligentes busca mitigar essas falhas por meio da padronização das interações e da disponibilidade ininterrupta do serviço.

Além disso, a análise preditiva, baseada em grandes volumes de dados, permite que a ferramenta antecipe incidentes recorrentes antes mesmo que eles impactem o usuário final. Tal proatividade reduz o tempo médio de resolução e eleva o nível de maturidade digital da infraestrutura tecnológica; assim, a IA deixa de ser apenas um recurso reativo para se tornar um pilar estratégico na sustentabilidade do suporte de TI. Por fim, essa transição assegura uma experiência mais fluida, alinhando as expectativas do negócio às demandas de um mercado cada vez mais imediatista.

Figura 1- Diagrama de Causa e Efeito da Ineficiência no Suporte de TI



Fonte: Autores (2025)

Este trabalho investiga a implementação de um chatbot que se fundamenta em IA para não apenas responder a perguntas frequentes, mas também para identificar com autonomia os casos que necessitam de assistência técnica especializada, encaminhando-os automaticamente para a equipe de suporte designada.

Diante disso, a problemática central deste estudo consiste em investigar como um chatbot baseado em IA pode agilizar o atendimento técnico, sem comprometer a qualidade da assistência.

Esta pesquisa se alinha à crescente necessidade de suporte ágil (HUANG; RUST, 2018) e aos avanços em APIs de Modelos de Linguagem de Grande Escala (LLMs), orientando-se pelo desenvolvimento de um chatbot utilizando a API do Google (Gemini), com uma interface de chat desenvolvida em React e integrado a um sistema de encaminhamento.

Objetivo Geral:

- Desenvolver uma ferramenta de suporte ao usuário baseada em Inteligência Artificial voltada à otimização e agilidade no atendimento de chamados de TI.

Objetivos Específicos:

- Definir a persona do chatbot conforme o contexto da área de atuação.
- Estruturar e formatar a base de conhecimento inicial com problemas técnicos, causas e soluções.
- Projetar e desenvolver o protótipo funcional em React, utilizando Vite como ferramenta de build.
- Integrar o chat à API Google Gemini, priorizando a agilidade e eficiência no atendimento aos usuários.

2 REFERENCIAL TEÓRICO

Esta seção apresenta os principais conceitos teóricos e tecnologias que fundamentam o desenvolvimento da solução proposta neste trabalho.

2.1 INTELIGÊNCIA ARTIFICIAL E PROCESSAMENTO DE LINGUAGEM NATURAL:

A Inteligência Artificial (IA) é um campo da ciência da computação dedicado à criação de sistemas capazes de realizar tarefas que normalmente exigiriam inteligência humana, como aprendizado, raciocínio e percepção (RUSSELL; NORVIG, 2021).

Dentro da IA, o Processamento de Linguagem Natural (PLN) é uma subárea focada na interação entre computadores e a linguagem humana. O PLN engloba técnicas e modelos computacionais que permitem às máquinas ler, entender e gerar linguagem de forma natural e coesa, processando grandes volumes de dados textuais para extrair sentido semântico (JURAFSKY; MARTIN, 2024). Em chatbots, o PLN é essencial para interpretar a intenção do usuário a partir de suas mensagens e formular respostas relevantes (ADAMOPOULOU; MOUSSIADES, 2020).

2.2 CHATBOTS, LLMS E ENGENHARIA DE PROMPT:

Chatbots são programas de computador projetados para simular conversas humanas, automatizando o atendimento e o fornecimento de informações (ADAMOPOULOU; MOUSSIADES, 2020). Grandes empresas de tecnologia têm investido pesadamente neste setor, criando plataformas robustas para facilitar essa interação, como o Azure Bot Service e o IBM Watson Assistant (IBM, [s.d.]; MICROSOFT, 2024).

Tradicionalmente, muitos chatbots operavam com base em regras predefinidas. No entanto, o advento dos Modelos de Linguagem de Grande Escala (LLMs) revolucionou essa área. LLMs, como a família de modelos Gemini do Google, são redes neurais massivas treinadas em vastos conjuntos de dados textuais, capazes de compreender nuances, manter o contexto da conversa e gerar respostas fluidas e de alta qualidade (GOOGLE, [s.d.]b). Este projeto utiliza a API do Google Gemini, conhecido por sua alta velocidade e eficiência em tarefas de conversação.

Para direcionar o modelo, foi aplicada a técnica de Engenharia de Prompt. Esta técnica consiste em estruturar cuidadosamente a entrada de texto (o prompt) fornecida ao modelo de IA para obter as respostas mais precisas e relevantes. No NexusSupport IA, isso é feito fornecendo um contexto inicial que define sua persona e base de conhecimento, uma prática essencial para alinhar modelos generativos a regras de negócios específicas (DAVENPORT, 2018).

2.3 ARQUITETURA DE APLICAÇÃO WEB MODERNA:

Para a materialização do chatbot, optou-se por uma arquitetura de frontend moderna, desacoplada de um backend tradicional.

A arquitetura do sistema NexusSupport IA foi projetada seguindo o modelo de aplicação de página única (SPA), focada em alta performance e baixa latência na comunicação com o motor de IA. Conforme ilustrado na Figura 2, a estrutura compõe-se de camadas integradas que permitem o fluxo de dados em tempo real.

Figura 2 – Arquitetura de Integração entre Frontend React e API Google Gemini

Fonte: Autores (2025)

A organização apresentada na Figura 2 demonstra a separação entre a interface e o processamento de inteligência. O frontend, desenvolvido em React e orquestrado pelo Vite, gerencia o estado da sessão localmente para manter o histórico da conversa. Esta camada comunica-se diretamente com a API do Google Gemini através de requisições HTTP (Fetch API), enviando o histórico completo e o prompt de contexto oculto para que o modelo retorne respostas precisas e personalizadas. Essa estrutura desacoplada permite que a lógica de suporte seja escalável sem sobrecarregar o dispositivo do usuário final.

2.3.1 React:

A Biblioteca para Interfaces de Usuário React (ou React.js) é uma biblioteca JavaScript de código aberto para a construção de interfaces de usuário (UIs) interativas e componentizadas. Desenvolvida pelo Facebook (agora Meta), sua principal vantagem é a reatividade: o React atualiza eficientemente a UI apenas nos componentes necessários quando o estado da aplicação muda (META, [s.d.]).

Sua abordagem baseada em componentes e renderização reativa o torna a escolha ideal para o desenvolvimento de aplicações dinâmicas, permitindo a criação de interfaces modulares onde o fluxo de dados é unidirecional, facilitando a manutenção e a previsibilidade do código (BANKS; PORCELLO, 2020).

2.3.2 API REST e Comunicação Assíncrona:

A comunicação entre o frontend e modelos de IA avançados, como o Gemini do Google, é realizada através de uma API REST (Representational State Transfer). APIs REST são um padrão de arquitetura fundamental para a comunicação entre sistemas, utilizando métodos HTTP (como POST, GET) para trocar dados, tipicamente no formato JSON (FIELDING, 2000).

Essa arquitetura assíncrona é crucial para o desempenho do chatbot, pois permite que o frontend continue responsivo enquanto aguarda o processamento complexo e a resposta do modelo de IA. No NexusSupport IA, o frontend utiliza a Fetch API nativa do JavaScript para enviar uma requisição POST para a API Gemini.

2.3.3 Ferramentas de Desenvolvimento:

Vite e NPM O ambiente de desenvolvimento do projeto é sustentado por ferramentas modernas que garantem agilidade e eficiência. O NPM (Node Package Manager) é a ferramenta essencial para gerenciar as dependências do projeto (como react e react-dom), consolidando-se como o padrão para o ecossistema JavaScript (FLANAGAN, 2021).

Como ferramenta de build e servidor de desenvolvimento, foi escolhido o Vite (VITE, [s.d.]). O Vite se destaca por oferecer um tempo de inicialização de servidor extremamente rápido e Hot Module Replacement (HMR) instantâneo, fatores que agilizam significativamente o ciclo de desenvolvimento de aplicações React.

3 METODOLOGIA

A condução do desenvolvimento do chatbot foi fundamentada em um modelo híbrido que integra a estrutura sequencial da **Metodologia em Cascata** com a flexibilidade das **Metodologias Ágeis**. Essa abordagem visou conferir rigor no planejamento das etapas de engenharia e dinamismo na execução, permitindo iterações rápidas e ajustes contínuos baseados em feedback (PRESSMAN; MAXIM, 2021). O modelo em cascata organizou o fluxo lógico do levantamento de requisitos à implantação, enquanto os princípios ágeis foram aplicados no refinamento da engenharia de prompt e da interface, ajustando o sistema conforme os testes com a API Google Gemini revelava novas necessidades.

Inicialmente, foram levantados os requisitos funcionais (como interface interativa, consulta à base de conhecimento e encaminhamento de chamados) e não funcionais (como usabilidade e segurança dos dados). Na sequência, foi escolhida a API de Inteligência Artificial Gemini do Google, considerando especialmente sua robustez em tarefas de Processamento de Linguagem Natural (PLN) em língua portuguesa, rapidez e a facilidade de integração via API REST.

A solução foi estruturada conforme a arquitetura de Aplicação de Página Única (SPA). O frontend do sistema compreende uma tela de chat desenvolvida com a biblioteca JavaScript React (React.js), permitindo a criação de uma aplicação web reativa e componentizada. A estilização foi realizada com CSS puro. O projeto foi inicializado e gerenciado com Vite e NPM.

A lógica da aplicação é gerenciada inteiramente no frontend utilizando React e seus hooks. O sistema gerencia o estado da sessão de chat e processa a comunicação com a API. A integração com a API do Google (Gemini) é realizada no lado do cliente (client-side) através de requisições HTTP (Fetch API).

Engenharia de Prompt e Base de Conhecimento O núcleo da metodologia de IA reside na Engenharia de Prompt. Esta técnica foi fundamental para adaptar o Modelo de Linguagem de Grande Escala (LLM) a um domínio altamente específico (suporte técnico em TI). Em vez de depender apenas da inteligência pré-treinada do modelo, uma extensa base de conhecimento contextualizada contendo problemas comuns de TI, suas causas e soluções, além de dados operacionais (persona, localização e horário de atendimento) foi estruturada e incorporada ao prompt inicial da conversação.

A inclusão estratégica deste prompt garante que o chatbot NexusSupport IA:

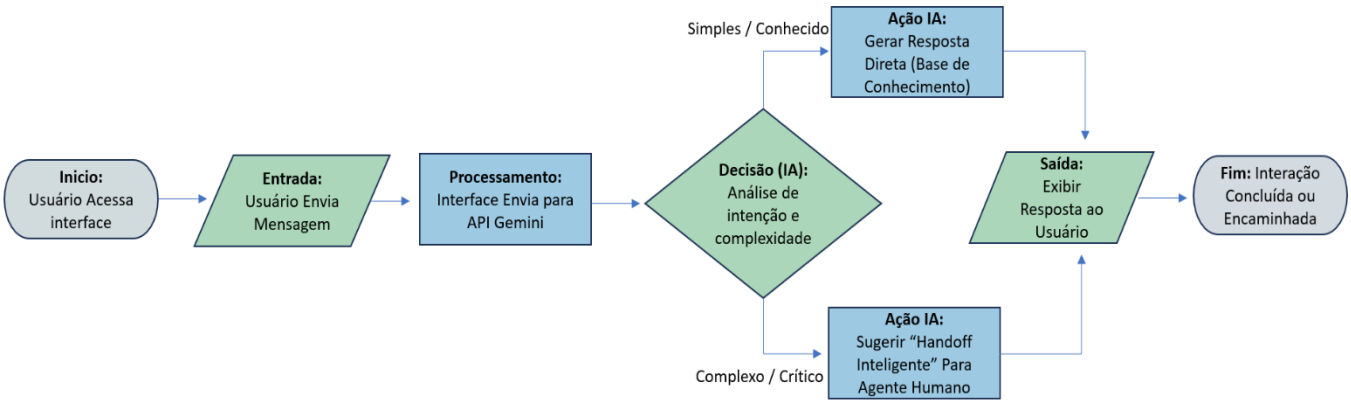
1. Mantenha a Persona: Responda de forma consistente com seu papel de suporte técnico.
2. Forneça Soluções Específicas: Baseie suas respostas nas soluções técnicas exatas fornecidas na base de conhecimento.
3. Haja Contextualmente: Utilize os dados contextuais operacionais (como horário de atendimento) para encaminhamento quando necessário.

Essa abordagem de Engenharia de Prompt assegura que a IA opere como um especialista configurado dentro do ecossistema de TI, otimizando o First Contact Resolution (FCR) e minimizando desvios de assunto.

Detalhes de Implementação (Lógica do Frontend): O componente principal da aplicação gerencia o estado da sessão de chat (chatHistory). O núcleo da lógica de comunicação é a formatação do histórico completo (incluindo o prompt oculto de contexto) e o envio via requisição POST para a API do Gemini. A resposta da IA é então recebida, tratada e utilizada para atualizar o estado do React, o que, por sua vez, renderiza a nova mensagem na interface.

O fluxo operacional dessa interação, desde a entrada do usuário até a tomada de decisão da IA para resolução ou encaminhamento, está detalhado na Figura 3.

Figura 3 – Fluxograma de Processamento e Decisão do NexusSupport IA



Fonte: Autores (2025)

4 RESULTADOS E DISCUSSÃO

A fase de implementação do protótipo do chatbot permitiu a materialização de componentes-chave do sistema. O desenvolvimento da interface de chat utilizando React resultou em uma interface de usuário funcional, reativa e estilizada com CSS. A arquitetura do protótipo foi validada, consistindo em uma Aplicação de Página Única (SPA) em React, construída com Vite.

A estrutura é componentizada, onde o estado principal (chatHistory) é gerenciado no componente principal e passado como props para os componentes filhos. A entrada do usuário é capturada em um componente de formulário, e cada balão de diálogo é renderizado por um componente de mensagem.

O mecanismo de Engenharia de Prompt implementado demonstrou ser eficaz. Ao inicializar o estado de conversa com a base de conhecimento ocultada, o chatbot foi capaz de responder a perguntas específicas sobre problemas de TI (ex: "Minha impressora não imprime") com as soluções exatas fornecidas na base de conhecimento (ex: "Verificar papel e cartucho, reconectar impressora"). Além disso, o chatbot demonstrou estar ciente de seu contexto operacional, respondendo corretamente sobre sua localização (Manaus-AM) e horário de atendimento.

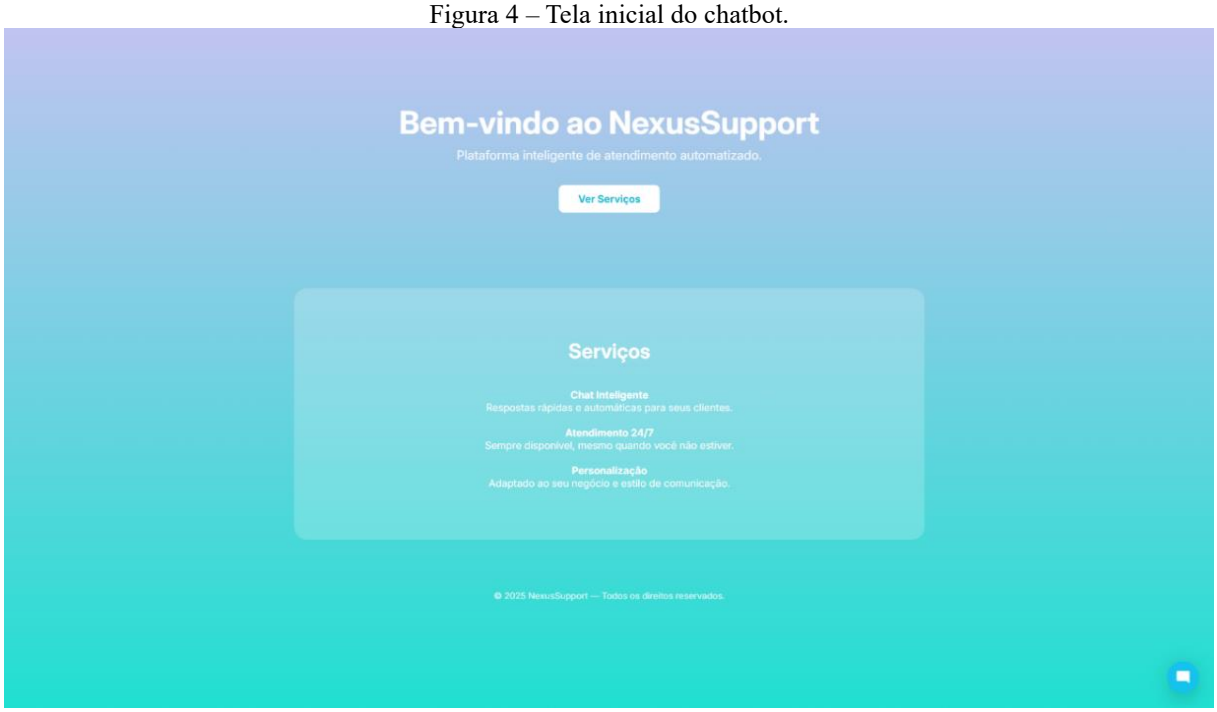
A comunicação com a IA é assíncrona (usando async/await na função fetch), tratando a resposta da API do Gemini (o JSON contendo o texto gerado) e atualizando o estado do React para renderizar a nova mensagem do bot. O uso de timeouts para exibir uma mensagem de processamento e a rolagem automática da tela de chat melhoraram a experiência do usuário.

Durante o desenvolvimento do NexusSupport IA, foram previstos desafios. O principal é a gestão da chave de API no lado do cliente. Embora funcional para prototipagem (carregada no ambiente do Vite), em um ambiente de produção, isso exigiria uma camada de backend (como uma cloud function) para servir como proxy e proteger a chave.

O resultado concreto obtido foi a implementação bem-sucedida da lógica de conversação contextual. Utilizando a API Gemini do Google e JavaScript (React), o protótipo funcional do frontend demonstrou ser eficiente em: 1) Receber a entrada do usuário; 2) Manter um histórico (chatHistory) das interações; 3) Enviar o histórico completo (incluindo o prompt de contexto) para a API do Gemini; e 4) Receber e exibir a resposta contextualizada da IA. Este mecanismo é o principal avanço técnico do projeto, validando a capacidade do NexusSupport IA de manter diálogos coerentes.

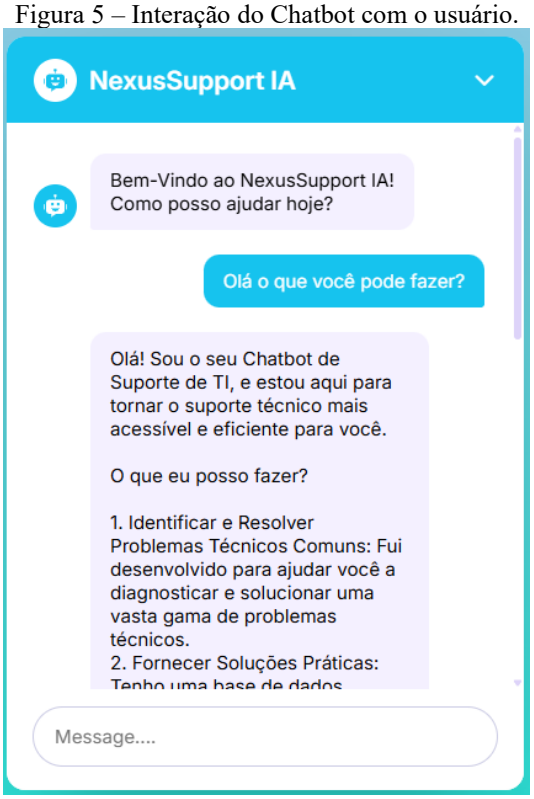
A eficácia dessa arquitetura reflete-se na redução do tempo de resposta percebido, uma vez que o processamento assíncrono evita o travamento da interface durante a requisição à API. Além da agilidade, a precisão das respostas contextuais válida a estratégia de Engenharia de Prompt adotada, provando que a IA pode atuar como uma primeira linha de suporte altamente confiável. Essa capacidade de triagem automática minimiza o encaminhamento de chamados triviais para os técnicos de nível superior, otimizando o fluxo de trabalho da equipe de TI. Portanto, o protótipo não apenas cumpre os requisitos técnicos estabelecidos, mas também demonstra viabilidade prática para ser escalado em cenários corporativos reais. Em última análise, a robustez da integração entre o React e a API do Gemini estabelece uma base sólida para futuras expansões, como a integração com bancos de dados de chamados em tempo real.

Figura 4 – Tela inicial do chatbot.



Fonte: Autores (2025)

Figura 5 – Interação do Chatbot com o usuário.



Fonte: Autores (2025)

5 CONSIDERAÇÕES FINAIS

O desenvolvimento do protótipo do NexusSupport IA evidenciou o potencial desta ferramenta para otimizar o suporte ao cliente, proporcionando maior rapidez e eficiência, especialmente no atendimento de primeiro nível. A arquitetura proposta, com uma interface de chat em React (construída com Vite) suportada por lógica de frontend em JavaScript e uma integração direta com a API Google Gemini, permitiu um fluxo de atendimento dinâmico e contextualizado.

Os objetivos delineados para este trabalho foram alcançados. A implementação do protótipo funcional validou a arquitetura de integração com a API do Google Gemini. A aplicação eficaz da engenharia de prompt (utilizando a base de conhecimento de TI) confirmou a capacidade do sistema em automatizar respostas de forma coerente e precisa.

Como perspectivas para trabalhos futuros, sugere-se o aprimoramento da compreensão semântica, a expansão da base de conhecimento, e a implementação de um backend simples (ex: Firebase Functions ou Node.js) para atuar como proxy de API, aumentando a segurança ao não expor a chave da API no lado do cliente. Adicionalmente, o desenvolvimento de um dashboard administrativo para monitoramento e a integração com outras plataformas (como WhatsApp ou Slack) ampliariam seu alcance.

Por fim, o projeto NexusSupport IA representa uma contribuição relevante para a aplicação prática da IA no aprimoramento do atendimento ao cliente. A sua concepção e desenvolvimento permitiram consolidar conhecimentos teóricos e práticos em desenvolvimento de software (web com React, JavaScript

e Vite), interação humano-computador e, crucialmente, na integração prática e engenharia de prompts com APIs de Modelos de Linguagem de Grande Escala (Google Gemini), reforçando a importância da inovação tecnológica para a solução de desafios contemporâneos no ambiente corporativo.

REFERÊNCIAS

ADAMOPOULOU, E.; MOUSSIADES, L. An Overview of Chatbot Technology. In: MAGLOGIANNIS, I. et al. (Eds.). Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, v. 584. Cham: Springer, p. 373-383, 2020.

BANKS, A.; PORCELLO, E. Learning React: Modern Patterns for Developing React Apps. 2. ed. Sebastopol: O'Reilly Media, 2020.

DAVENPORT, T. H. The AI Advantage: How to Put the Artificial Intelligence Revolution to Work. Cambridge: MIT Press, 2018.

FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. 2000. 180 f. Tese (Doutorado em Informação e Ciência da Computação) – University of California, Irvine, 2000. Disponível em: https://roy.gbiv.com/pubs/dissertation/fielding_dissertation.pdf. Acesso em: 10 dez. 2025.

FLANAGAN, D. JavaScript: O Guia Definitivo. 7. ed. Porto Alegre: Bookman, 2021.

GOOGLE. Google Cloud Dialogflow documentation. [S.l.]: Google, [s.d.]a. Disponível em: https://cloud.google.com/dialogflow/docs. Acesso em: 2 nov. 2025.

GOOGLE. Documentação Oficial do Google Gemini. [S.l.]: Google, [s.d.]b. Disponível em: https://ai.google.dev/docs. Acesso em: 5 nov. 2025.

HUANG, Ming-Hui; RUST, Roland T. Artificial Intelligence in Service. Journal of Service Research, Thousand Oaks, v. 21, n. 2, p. 155-172, 2018. Disponível em: https://journals.sagepub.com/doi/10.1177/1094670517752459. Acesso em: 10 dez. 2025.

IBM. IBM Watson Assistant. [S.l.]: IBM, [s.d.]. Disponível em: https://www.ibm.com/products/watson-assistant. Acesso em: 12 out. 2025.

JURAFSKY, D.; MARTIN, J. H. Speech and Language Processing. 3. ed. Upper Saddle River: Pearson, 2024.

LUO, X. et al. Frontiers: Machines vs. Humans: The Impact of Artificial Intelligence Chatbot Disclosure on Customer Purchases. Marketing Science, Catonsville, v. 38, n. 6, p. 937–947, 2019.

META. Documentação Oficial do React. [S.l.]: Meta, [s.d.]. Disponível em: https://react.dev/. Acesso em: 12 jun. 2025.

MICROSOFT. Azure AI Bot Service documentation. [S.l.]: Microsoft Learn, 2024. Disponível em: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-overview-introduction. Acesso em: 10 dez. 2025.

PRESSMAN, R. S.; MAXIM, B. R. Engenharia de Software: Uma Abordagem Profissional. 9. ed. Porto Alegre: AMGH, 2021.

RUSSELL, S.; NORVIG, P. Inteligência Artificial: Uma Abordagem Moderna. 4. ed. Rio de Janeiro: GEN LTC, 2022.

VITE. Documentação Oficial do Vite. [S.l.]: VITE, [s.d.]. Disponível em: https://vitejs.dev/. Acesso em: 2 nov. 2025.