INSTITUTO FEDERAL GOIANO - CAMPUS CERES
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

LUCAS RODRIGUES DIAS

CRIAÇÃO DE APIS REST PARA GESTÃO DE BOTS INTELIGENTES NO WHATSAPP UTILIZANDO OS MODELOS WHISPER E CHAT GPT PARA APLICAÇÃO NO SISTEMA DOUMI

CERES – GO
2024

Lucas Rodrigues Dias

Criação de APIs REST para Gestão de Bots Inteligentes no WhatsApp utilizando os modelos Whisper e Chat GPT para aplicação no sistema Doumi

Trabalho de conclusão de curso apresentado ao curso de Sistemas de Informação do Instituto Federal Goiano - Campus Ceres, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação, sob orientação do Prof. Vilson Soares de Siqueira.

CERES – GO
2024

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas (SIBI) – Instituto Federal Goiano

D541c

Dias, Lucas Rodrigues.

Criação de APIS REST para gestão de *bots* inteligentes no WhatsApp utilizando os modelos *Whisper* e Chat GPT para aplicação no Sistema Doumi [manuscrito] / Lucas Rodrigues Dias. – Ceres, GO: IF Goiano, 2024.
28 fls. : tabs.

Orientador: Prof. MSc. Vilson Soares de Siqueira.

Trabalho de Conclusão do Curso (Bacharelado em Sistemas de Informação) – Instituto Federal Goiano, Campus Ceres, 2024.

1. WhatsApp. 2. Chat GPT. 3. Whisper. 4. Bots. 5. Java Script. I. Siqueira, Vilson Soares de. II. Título. III. Instituto Federal Goiano.

CDU 004.4’2

Ficha elaborada por Johnathan Pereira Alves Diniz – Bibliotecário/CRB 1 nº 2376

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Lucas Rodrigues Dias

Matrícula:
2019103202030202

Título do trabalho:
CRIAÇÃO DE APIS REST PARA GESTÃO DE BOTS INTELIGENTES NO WHATSAPP UTILIZANDO OS MODELOS WHISPER E CHAT GPT PARA APLICAÇÃO NO SISTEMA DOUMI

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 21 / 12 / 2024

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Ceres — Local
21 / 12 / 2024 — Data

gov.br Documento assinado digitalmente
LUCAS RODRIGUES DIAS
Data: 26/12/2024 16:20:54-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

Assinatura do(a) orientador(a)

gov.br Documento assinado digitalmente
VILSON SOARES DE SIQUEIRA
Data: 26/12/2024 15:14:05-0300
Verifique em https://validar.iti.gov.br

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

Aos 13 dias do mês de dezembro do ano de dois mil e vinte e quatro, realizou-se a defesa de Trabalho de Curso do acadêmico **Lucas Rodrigues Dias**, do Curso de Bacharelado em Sistemas de Informação, matrícula 2019103202030202, cujo título é "**Desenvolvimento de APIs REST para Gestão de Bots Inteligentes no WhatsApp com Integração do Modelo Whisper no Sistema Doumi**". A defesa iniciou-se às 15 horas e 02 minutos, finalizando-se às 16 horas e 00 minutos. A banca examinadora considerou o trabalho **APROVADO** com média **8,6** no trabalho escrito, média **9,6** no trabalho oral, apresentando assim média aritmética final de **9,1** pontos, estando o estudante APTO para fins de conclusão do Trabalho de Curso.

Após atender às considerações da banca e respeitando o prazo disposto em calendário acadêmico, o estudante deverá fazer a submissão da versão corrigida em formato digital (.pdf) no Repositório Institucional do IF Goiano – RIIF, acompanhado do Termo Ciência e Autorização Eletrônico (TCAE), devidamente assinado pelo autor e orientador.

Os integrantes da banca examinadora assinam a presente.

(Assinado Eletronicamente)
Prof. Dr. Vilson Soares de Siqueira

Orientador

(Assinado Eletronicamente)
Prof. Me. Danillo Freire Pacheco

Membro

(Assinado Eletronicamente)
Prof. Me. Thalia Santos de Santana

Membro

Documento assinado eletronicamente por:

- **Vilson Soares de Siqueira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 13/12/2024 16:28:27.
- **Thalia Santos de Santana**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 13/12/2024 16:30:59.
- **Danillo Freire Pacheco**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO** , em 13/12/2024 17:08:13.

Este documento foi emitido pelo SUAP em 13/12/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 662003
Código de Autenticação: f5c1583de6

INSTITUTO FEDERAL GOIANO
Campus Ceres
Rodovia GO-154, Km 03, SN, Zona Rural, CERES / GO, CEP 76300-000
(62) 3307-7100

AGRADECIMENTOS

Agradeço ao apoio e incentivo do professor orientador, colegas, amigos e familiares para a realização deste trabalho, bem como na realização rotineira das atividades do curso. Agradeço também aos apoiadores e patrocinadores da ferramenta, que estiveram firmes durante 10 meses de desenvolvimento, contribuindo com um valor total de vinte e sete mil e quinhentos reais para o desenvolvimento dessas APIs. Agradeço também ao meu amigo Pedro Felipe Dias, que contribuiu com o projeto macro, desenvolvendo a interface visual da Doumi.

“A inteligência artificial é a nova eletricidade”.

Andrew Ng

RESUMO

O *WhatsApp* tem sido cada vez mais utilizado como canal de comunicação tanto pessoal quanto empresarial, sendo usado em 92% das empresas em atendimento ao cliente no Brasil, mas à medida que o interesse aumenta, oferecer um rápido e satisfatório atendimento a muitos clientes se torna difícil e caro, podendo levar à atrasos e insatisfação do cliente. Neste sentido, o presente trabalho de conclusão teve como objetivo construir duas APIs REST para gerenciar bots conectados ao *WhatsApp* via QR Code e alimentados pela API do Chat GPT e *Whisper*, possibilitando múltiplos atendimentos ao mesmo tempo, atendimento multilíngue e reconhecimento de áudios e vídeos em tempo real. Para isto foram criadas duas APIs com arquiteturas diferentes, baseadas em Javascript e Typescrypt, o projeto demorou cerca de dez meses de trabalho em um computador pessoal e contou com o apoio financeiro de patrocinadores.

Palavras-chave: WhatsApp. Chat GPT. Whisper. Bots. JavaScript.

ABSTRACT

WhatsApp has increasingly been used as a communication channel both personally and in business, with it being employed by 92% of companies for customer service in Brazil. However, as interest grows, providing fast and satisfactory service to many customers becomes challenging and expensive, potentially leading to delays and customer dissatisfaction. In this context, this thesis aimed to develop two REST APIs to manage bots connected to WhatsApp via QR Code, powered by the Chat GPT and Whisper APIs, enabling simultaneous multilingual customer service and real-time audio and video recognition. For this purpose, two APIs were created with different architectures, based on Javascript and Typescript. The project took about ten months of work on a personal computer and received financial support from sponsors.

Keywords: WhatsApp. Chat GPT. Whisper. Bots. JavaScript.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

A crescente digitalização das interações cotidianas tem elevado a importância das redes sociais, com o *WhatsApp* destacando-se como uma das mais notáveis no Brasil, utilizada por 99% dos brasileiros e por 92% das empresas formais no país, sendo considerada uma ferramenta importante para apoiar e envolver o público (O Globo, 2024). Neste sentido, a habilidade de gerenciar um grande volume de interações torna-se desafio crítico para essa dinâmica corporativa, com gastos excepcionais com mão de obra e muitas vezes a insatisfação do cliente com a demora no atendimento.

Tradicionalmente, os bots de atendimento ao cliente utilizam menus pré-configurados que podem limitar a interação e afetar negativamente a experiência do usuário, já que a capacidade de entender o cliente, fornecer respostas relevantes e resolver problemas efetivamente é essencial para a satisfação do cliente (Nicolescu ; Tudorache, 2022). Por outro lado, o Chat GPT (Generative Pré-trained Transformer), desenvolvido pela OpenAI, destaca-se na geração de texto usando linguagem natural. Este modelo é treinado com vastos volumes de dados, permitindo-lhe compreender e responder às solicitações sem configurações específicas para cada tarefa. Ele opera dividindo o texto inserido em unidades menores chamadas tokens, que podem ser palavras completas ou partes delas. Esta funcionalidade possibilita ao Chat GPT prever e formular respostas que se alinham ao contexto geral da conversa, tornando-o uma ferramenta valiosa na criação de chatbots e assistentes virtuais inteligentes (Caetano, 2024).

Outra funcionalidade importante destacada na ferramenta desenvolvida foi a possibilidade de compreender áudios e vídeos utilizando o *Whisper*, também desenvolvido pela OpenAI e treinado com 680 mil horas de conteúdo extraído da internet em diferentes línguas. De acordo com Reis e Delbuono, 2023, o *Whisper* é particularmente robusto:

> "Por ser um modelo robusto, ele permite realizar transcrições e entendimento de falas sob as mais variadas condições, como diferentes sotaques, presença de ruídos no áudio, termos mais técnicos, diferentes velocidades de interlocução, etc" (Reis ; Delbuono, 2023).

Esta capacidade torna o *Whisper* uma solução altamente eficaz para o processamento de linguagem natural em cenários complexos e dinâmicos.

A integração de bots de IA (Inteligência Artificial), como os que operam com tecnologias Chat GPT e *Whisper*, representa uma inovação promissora no atendimento ao cliente e corte de custos. Essa abordagem é destacada na dissertação de (Ambrioso, 2024).

Este trabalho é parte do projeto macro Doumi, que tem como objetivo projetar um sistema de gerenciamento de bots do *WhatsApp*. Aqui será apresentado e relatado o processo de desenvolvimento das APIs (Application Programming Interfaces) *back-end* e *Whatsapp*, de forma que permita ao *front-end* consumir a API desenvolvida por meio de um projeto complementar desenvolvido também como trabalho de conclusão de curso como a parte frontal do sistema Doumi.

O objetivo geral deste trabalho é desenvolver duas APIs REST (Representational State Transfer) para a criação e gerenciamento de bots inteligentes no *WhatsApp*, que simplifiquem a forma como os atendimentos online são realizados. Os objetivos específicos incluem a integração de serviços de conversão de mídia em texto com *Whisper*, com processamento eficaz em filas de mensagens e uso de *webhooks* para sincronização e respostas rápidas.

Considerando o exposto, infere-se que a ferramenta possa contribuir com diversos tipos de empresas, aprimorando a eficácia operacional e a satisfação dos clientes, preenchendo uma lacuna na disponibilização de uma ferramenta de fácil acesso e que possa ser adaptável e eficiente para diversos nichos e formas de atuação.

Para cumprir com este objetivo, a metodologia propõe o uso das tecnologias como Node.js, TypeScript e Nest.js e a integração com bancos de dados como MariaDB e MongoDB. Dessa maneira, a implementação será mantida escalar e segura para atender um volume de interações suficientemente alto, aceitável para o *WhatsApp*.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver APIs REST para gestão de bots inteligentes no *WhatsApp* utilizando frameworks Javascript e otimizando a experiência de interação comercial automatizada por meio da combinação de diversas tecnologias, tais como: Node.js, TypeScript, Nest.js e Whatsapp-web.js, além de utilizar os bancos de dados MariaDB e MongoDB para escalabilidade e segurança.

1.1.2 Objetivos Específicos

- Implementar funcionalidades que permitam a conversão de áudios e vídeos em texto utilizando a API do *Whisper*;
- Desenvolver mecanismos para a gestão eficaz das filas de mensagens, assegurando que cada interação seja tratada de forma ágil e ordenada;
- Configurar *webhooks* para a sincronização de dados entre o *back-end*, API *WhatsApp* e Asaas;

2 REFERENCIAL TEÓRICO

Nesta seção serão apresentadas as bases teóricas necessária para desenvolvimento do projeto. A seção foi organizada da seguinte forma: Na seção 2.1 discute-se os principais frameworks utilizados no desenvolvimento das APIs. A seção seguinte, 2.2 discute ferramentas parecidas ou com objetivos similares aos do presente trabalho.

2.1 FRAMEWORKS JAVASCRIPT UTILIZADOS

2.1.1 Node.js

A **API *back-end*** foi desenvolvida utilizando **Node.js**, uma plataforma baseada em JavaScript desenvolvida em 2009 para permitir a criação de aplicativos de alta escalabilidade. Esta é uma das ferramentas mais conhecidas devido à sua ampla gama de ferramentas e frameworks, tornando-se uma das opções mais populares para construir APIs por sua alta

eficiência e ferramentas fáceis de usar (Barsoti ; Gibertoni, 2020).

2.1.2 Express

Express é um **framework** para gerenciamento de rotas *server-side* da API em **Node.js**. Ele oferece facilidade e flexibilidade ao integrar esses caminhos e suas solicitações, e sua arquitetura simplificada facilita o desenvolvimento rápido de qualquer função solicitada (Carmo, 2023).

2.1.3 Sequelize

O Sequelize, um ORM (Object-Relational Mapping) para **Node.js**, foi configurado para operar o banco de dados relacional MariaDB. O **framework** permite uma conexão entre banco de dados e as rotas, simplificando a gestão do banco, o que otimiza o tempo de produção e o controle de recursos (Barsoti ; Gibertoni, 2020).

2.1.4 JsonWebToken

O JSON Web Token, ou JWT, é um método bastante usado para autenticação em APIs REST. Facilita o envio seguro de informações entre o cliente e o servidor, eliminando a necessidade de enviar repetidamente credenciais. Isso não apenas torna o processo mais seguro, mas também diminui a latência, melhorando assim a performance das aplicações web. (Montanheiro ; Carvalho ; Rodrigues, 2017).

2.1.5 BCrypt

O BCrypt foi usado na criptografia das senhas armazenadas no sistema, Este algoritmo é baseado no **Blowfish**, uma cifra de blocos simétrica que executa uma cifra de Feistel por 16 vezes, aumentando a segurança ao dificultar a previsibilidade do hash final. Uma característica marcante do BCrypt é sua fase inicial de configuração de chaves, que é intencionalmente custosa para tornar o processo de hash lento e exigente em termos de

memória, aumentando assim a segurança contra ataques rápidos (Queiroz, 2022).

2.1.6 Axios

O Axios é destacado como uma ferramenta essencial para a comunicação entre APIs em aplicações tanto *front-end* quanto *back-end*. Esta biblioteca simplifica a execução de requisições HTTP (Hypertext Transfer Protocol), proporcionando uma maneira eficiente e clara de interagir com serviços web. Sua utilização no projeto permite a manipulação de dados entre o as APIs e emissão de **webhooks**, facilitando tanto o envio de informações como a recepção de respostas, o que é fundamental para o funcionamento eficaz de aplicações modernas que dependem de interações ágeis e seguras com APIs (Cunha, 2024).

2.1.7 Mongoose

No projeto em questão, o MongoDB foi escolhido para a persistência de dados das instâncias, utilizando o **Mongoose** como uma ferramenta modeladora eficaz. Este ODM (*Object Document Mapper*) é essencial para traduzir as estruturas de dados da aplicação em documentos que se integram naturalmente às *collections* do MongoDB. Além disso, o Mongoose oferece uma estruturação de dados que facilita tanto o armazenamento quanto a consulta de grandes volumes de informação, sendo particularmente vantajoso para desenvolvedores iniciantes por sua configuração simples e documentação acessível, o que agiliza o processo de aprendizado (Rocha, 2022).

2.1.8 Nest.js

Usamos o **Nest.js** para desenvolver a API do *WhatsApp* em nosso sistema, aproveitando sua capacidade de integrar diferentes tipos de bancos de dados e sua estrutura modular eficaz. Este **framework**, que usa **TypeScript** e possui uma

arquitetura semelhante ao Angular, é ideal para criar **APIs *back-end*** escaláveis e confiáveis. Com o **Nest.js**, conseguimos gerenciar requisições e serviços de forma clara e organizada, facilitando tanto o desenvolvimento quanto a manutenção do projeto (Oliveira ; Lima ; Caridade, 2022).

2.1.9 Whatsapp-web.js

Para a implementação da API do *WhatsApp* não oficial no nosso sistema, optamos pelo uso do **whatsapp-web.js**, uma biblioteca que simula o *WhatsApp Web* em segundo plano. Essa escolha foi motivada pela sua ampla manutenção pela comunidade *open-source* e pela capacidade de operar sem custos adicionais, diferentemente da API oficial do *Whatsapp Business*. O **whatsapp-web.js** oferece uma API local que facilita o desenvolvimento, permitindo um controle eficiente e direto da plataforma (Abreu, 2023).

2.2 TRABALHOS RELACIONADOS

Os trabalhos que inspiraram este trabalho serão discutidos nesta seção.

O trabalho de Lans e Brilinger apresentou o processo de transição de uma operadora de planos de saúde do atendimento telefônico para o uso de aplicativos de mensagens, especificamente o *WhatsApp Business*, e posteriormente a tecnologia API, focando em como a automação via RPA (Robotic Process Automation) e a integração do *WhatsApp* ajudaram a melhorar a eficiência na gestão de grandes volumes de interações com clientes. Destacaram-se benefícios como a redução de erros, melhor integração de setores e controle aprimorado sobre os indicadores de atendimento. Contudo, enfrentou-se desafios como limitações de interação após 24 horas e dependência excessiva de suporte técnico (Lanz & Brilinger, 2022). Em contraste, neste projeto, o problema foi abordado com uma solução mais inovadora utilizando modelos de IA e uma API não oficial do *Whatsapp* desenvolvida durante o projeto, o que pode oferecer maior

flexibilidade e adaptabilidade às necessidades específicas das operações sem as restrições de tempo e suporte impostas pela API oficial e RPA.

No trabalho de Nascimento, foi detalhado a implementação de um chatbot via *Whatsapp* para centralizar o gerenciamento de chamados nos diversos setores da instituição. Segundo o autor, a escolha dessa plataforma baseou-se na familiaridade dos usuários com o *WhatsApp*, permitindo uma integração eficiente e simplificada que dispensa treinamentos adicionais para os servidores. O sistema desenvolvido conseguiu não apenas centralizar mas também simplificar o processo de abertura de chamados, expandindo a acessibilidade aos serviços através de dispositivos móveis e desktops. Esse avanço facilitou as solicitações de serviços pela comunidade acadêmica, melhorando significativamente a eficiência operacional da universidade. Além disso, a ferramenta mostrou-se robusta durante os testes, adaptando-se bem às diferentes necessidades dos usuários sem as restrições encontradas em soluções comerciais padrão (Nascimento *et al*, 2024). No artigo em questão, foram utilizados com sucesso menus para guiar a interação com os usuários finais, diferentemente do projeto Doumi, que pretende simular a interação humana com fornecimento de respostas com linguagem natural e configuração de fluxos por meio da base de conhecimento ao invés de menus pré-configurados, contudo, os dois projetos se aproveitam de **frameworks** javascript *open-source* para criação das APIs não oficiais do *Whatsapp*.

No mesmo sentido, o autor Sousa explorou a implementação de um chatbot com Chat GPT-3.5 para aprimorar o atendimento ao cliente na indústria de telecomunicações. Apresentado como trabalho de conclusão de curso na Coordenadoria do Curso de Engenharia de Telecomunicações, o estudo destaca o papel transformador dos chatbots em responder a demandas de clientes de forma eficiente e contextual. A implementação do sistema foi realizada utilizando tecnologias como React/Next.js, Node.js e a API da OpenAI, configuradas através da Vercel num chat online interativo (Souza,

2023). Neste projeto foram utilizadas tecnologias Javascript, como na Doumi, mas não foram utilizados bancos de dados nem integrações com plataformas externas, demonstrando um acesso direto do cliente ao chat de atendimento por IA, contudo, o processo de comunicação com a API do Chat GPT segue a mesma lógica.

Já no trabalho de Rejin e Rajest eles desenvolveram um chatbot para *WhatsApp* usando linguagem de programação Python, e utilizaram algoritmos de aprendizado de máquina que permitem ao chatbot processar grandes quantidades de dados e responder às consultas dos usuários em tempo real. Eles aplicam várias técnicas de IA para garantir que o chatbot possa compreender entradas do usuário e gerar respostas apropriadas. Estes algoritmos são programados para analisar mensagens recebidas, identificar termos chave e determinar a resposta mais relevante baseada no contexto da consulta. O sistema é treinado com um conjunto de dados pré-programados, permitindo que ele maneje uma ampla variedade de tarefas. Com o tempo, o desempenho do chatbot melhora à medida que ele interage com os usuários e aprende com suas entradas, sendo uma combinação de processamento de linguagem natural e algoritmos de aprendizado de máquina, que são componentes típicos em muitos sistemas modernos de chatbot para permitir uma comunicação eficaz e contextualmente apropriada sem intervenção humana direta (Regin ; Rajest, 2024).

No trabalho de Spinei são apresentados os passos para a implementação do Chat GPT-3.5 em robôs de companhia, como os AlphaMini, foi examinada com o objetivo de melhorar a naturalidade e empatia nas interações humano-robô. Utilizando tecnologias como *Whisper* para transcrição de fala e *Amazon Comprehend* para análise de sentimentos, o estudo buscou integrar respostas do Chat GPT-3.5 aprimoradas com análise de sentimentos em tempo real. No entanto, os resultados indicaram que não houve diferenças significativas na percepção de empatia e inteligência emocional entre as respostas padrão do Chat GPT e aquelas enriquecidas pelo *Amazon Comprehend*, sugerindo que as capacidades existentes do Chat GPT-3.5 já são suficientemente empáticas para uso em robótica social

(Spinei, 2023). Este projeto utilizou Chat GPT e *Whisper* numa arquitetura totalmente diferente e reforça a afirmação de que as conversas com IAs avançadas podem ser percebidas como autênticas e empáticas, mesmo sem intervenções complexas de análise de sentimentos, o que as torna uma ferramenta estratégica na conquista do cliente diante do cenário atual.

No trabalho de Bumann foi apresentado um chatbot automatizado integrado a um aplicativo móvel, utilizando APIs de reconhecimento de fala (STT) e de síntese de fala (TTS). A análise detalhada de várias APIs STT e TTS ajudou a selecionar as mais adequadas para implementação, com base em critérios como precisão, velocidade, custo e qualidade da documentação. O *Whisper* foi escolhido para STT devido à sua alta precisão e velocidade, enquanto o Microsoft Azure API foi selecionado para TTS, especialmente pela sua capacidade de gerar Visemes, melhorando a sincronização labial em animações. A implementação resultou em um chatbot funcional que foi integrado ao aplicativo móvel e testado quanto à usabilidade. A constante evolução da tecnologia de reconhecimento e síntese de fala sugere a necessidade de atualizações regulares para aproveitar novas soluções avançadas. Além disso, a importância de considerar a experiência do usuário foi enfatizada, recomendando a integração contínua de feedbacks para otimizar a interação humano-computador. Este projeto não apenas destacou a eficácia das APIs de STT e TTS na melhoria da interação do usuário em aplicações móveis, mas também forneceu um guia valioso para futuros projetos na área de aplicativos de fala (Bumann, 2023). O artigo analisado mostrou que entre as ferramentas *Whisper, Deepgram, AssemblyAI, Google's Speech-to-Text* e *Microsoft Azure's Speech-to-Text* o *Whisper* era a que mais se destacava em questão de precisão, velocidade, custo e qualidade da documentação, o que demonstra a capacidade ferramenta, que também foi utilizada nesse projeto, com o mesmo fim.

3 METODOLOGIA

3.1 ESCOPO DO PRODUTO:

O sistema é composto por duas APIs REST, projetadas especificamente para criar e gerenciar bots inteligentes no WhatsApp. Essas APIs foram desenvolvidas com foco em modularidade, escalabilidade e eficiência, garantindo uma comunicação efetiva e ágil entre os componentes. A escolha da arquitetura REST para as APIs sublinha o compromisso com a integração e flexibilidade, permitindo que o sistema se adapte facilmente a mudanças e evolua conforme as necessidades dos usuários.

3.1.1 Tecnologia REST em APIs

As APIs, ou Interfaces de Programação de Aplicações, representam um método crucial de comunicação entre diferentes sistemas, permitindo que um sistema ofereça serviços e informações que podem ser utilizados por outro sem que haja necessidade de compartilhar os detalhes de sua implementação interna. Essa capacidade de interconexão facilita o desenvolvimento de software ao permitir que sistemas distintos se beneficiem de funcionalidades específicas sem necessidade de replicar o código ou desenvolver funcionalidades do zero. Segundo Junior, Rocha e Maciel (2021), a API serve como um canal que possibilita a interação, oferecendo suas funcionalidades ou rotinas para mais de um sistema, de forma que esses sistemas não precisem conhecer os detalhes internos da implementação. Esse modelo de interação é amplamente utilizado em serviços modernos, como sistemas de geolocalização e plataformas de pagamento, onde múltiplas aplicações acessam a mesma funcionalidade através de uma interface comum, independente da linguagem de programação ou plataforma tecnológica na qual estão baseadas.

A tecnologia REST, ou Transferência de Estado Representacional, é um estilo de arquitetura que utiliza o

protocolo HTTP para comunicação. APIs REST são projetadas para serem leves, mantendo a comunicação entre cliente e servidor tanto eficiente quanto eficaz. Utilizando métodos padrão HTTP como GET, POST, PUT e DELETE, essas APIs permitem que os desenvolvedores manipulem recursos representados tipicamente em formatos como JSON (JavaScript Object Notation) ou XML (Extensible Markup Language). Esta abordagem não apenas simplifica a arquitetura de desenvolvimento, mas também maximiza a performance e a escalabilidade, facilitando a integração de diversas aplicações e sistemas de maneira flexível e descomplicada (Bruno, 2023).

3.1.2 API Back-end

A API foi construída utilizando Node.js como plataforma e MySQL para armazenamento de dados, com o *Sequelize* sendo usado para gerenciar as operações no banco, uma configuração que proporciona desempenho, escalabilidade e uma interface intuitiva para manipulação de dados, além de permitir uma estrutura de código mais organizada e fácil manutenção.

3.1.3 API Whatsapp

A API foi construída utilizando TypeScript com o framework Nest.js e whatsapp-web.js, integrando um banco de dados MongoDB. Essa abordagem oferece vantagens como tipagem estática para maior segurança no código, escalabilidade e facilidade de manutenção proporcionadas pelo Nest.js, além da capacidade de gerenciamento de mensagens em tempo real com o whatsapp-web.js.

3.2 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Para garantir o correto funcionamento em conjunto das APIs foram definidos requisitos funcionais e não funcionais que estão dispostos a seguir, os requisitos funcionais foram divididos em API *back-end* e API

Whatsapp, os requisitos não funcionais foram divididos em API *Whatsapp* e ambas as APIs.

3.2.1 Requisitos Funcionais:

3.2.1.1 API Back-end

Tabela 1 - RF01 - API *Back-end*: Cadastro e acesso de usuários

Identificação do Requisito:	RF01
Nome do Requisito	Cadastro de Usuários
Especificação do Requisito	
A API deve permitir que usuários possam se cadastrar e acessar a API utilizando os dados cadastrados.	

Tabela 2 - RF02 - API *Back-end*: Gerenciamento de Bots

Identificação do Requisito:	RF02
Nome do Requisito	Gerenciamento de Bots
Especificação do Requisito	
A API deve possibilitar a criação, edição, exclusão e mudança na versão da IA nos bots de *WhatsApp*.	

Tabela 3 - RF03 - API *Back-end*: Gerenciamento de Conversas

Identificação do Requisito:	RF03
Nome do Requisito	Gerenciamento de Conversas
Especificação do Requisito	
A API deve possibilitar que o usuário possa consumir históricos de conversas e chats, enviar mensagens, enviar arquivos e encerrar atendimento.	

Tabela 4 - RF04 - API *Back-end*: Gerenciamento da base de conhecimento

Identificação do Requisito:	RF04
Nome do Requisito	Gerenciamento da base de conhecimento
Especificação do Requisito	
A API deve permitir que o usuário cadastre bases de conhecimento e adicione respostas pré-definidas chamadas de acordo com as funções cadastradas.	

Tabela 5 - RF05 - API *Back-end*: Funções avançadas na base de conhecimento

Identificação do Requisito:	RF05
Nome do Requisito	Funções avançadas na base de

	conhecimento
Especificação do Requisito	
A API deve permitir que o envio de arquivos e redirecionamento automático para atendimento humano sejam utilizados na base de conhecimento.	

Tabela 6 - RF06 - API *Back-end*: Métricas de uso

Identificação do Requisito:	RF06
Nome do Requisito	Métricas de uso
Especificação do Requisito	
A API deve gerar métricas de mensagens e gastos por dia, quantidade de clientes, mensagens, atendimentos finalizados e atendimentos pendentes.	

Tabela 7 - RF07 - API *Back-end*: Gerenciamento de Tags

Identificação do Requisito:	RF07
Nome do Requisito	Gerenciamento de Tags
Especificação do Requisito	
A API deve permitir ao usuário criar, consultar, editar e excluir etiquetas (tags).	

Tabela 8 - RF08 - API *Back-end*: Etiquetamento de usuários

Identificação do Requisito:	RF08
Nome do Requisito	Etiquetamento de usuários
Especificação do Requisito	
A API deve permitir ao usuário atribuir etiquetas aos usuários do bot.	

Tabela 9 - RF09 - API *Back-end*: Processamento de mensagens com Chat GPT

Identificação do Requisito:	RF09
Nome do Requisito	Processamento de mensagens com Chat GPT
Especificação do Requisito	
A API deve enviar o registro de mensagens junto às instruções e base de conhecimento para o Chat GPT para que a ferramenta faça o processamento.	

Tabela 10 - RF10 - API *Back-end*: Integração com API *WhatsApp*

Identificação do Requisito:	RF10
Nome do Requisito	Integração com API *WhatsApp*

Especificação do Requisito
A API deve se integrar à API *WhatsApp* para solicitar o *QR Code* de conexão, enviar mensagens e enviar arquivos.

Tabela 11 - RF11 - API *Back-end*: Integração com API Asaas

Identificação do Requisito:	RF11
Nome do Requisito	Integração com API Asaas
Especificação do Requisito	
A API deve solicitar ao Asaas o pix para adição de saldo quando necessário e receber o webhook quando confirmado o pagamento.	

Tabela 12 - RF12 - API *Back-end*: Controle de resposta da IA por cliente

Identificação do Requisito:	RF12
Nome do Requisito	Controle de resposta da IA por cliente
Especificação do Requisito	
A API deve permitir que o usuário ative ou desative a IA para um cliente, assim como a base de conhecimento, quando for o caso.	

Tabela 13 - RF13 - API *Back-end*: Acionamento de Webhooks

Identificação do Requisito:	RF13
Nome do Requisito	Acionamento de Webhooks
Especificação do Requisito	
A API deve ter endpoints e configurações adequadas para os 3 webhooks utilizados: Adição de saldo, Mensagem recebida e Status de instância alterado.	

3.2.1.2 API Whatsapp

Tabela 14 - RF01 - API *Whatsapp*: Gerenciamento de chaves de API

Identificação do Requisito:	RF01
Nome do Requisito	Gerenciamento de chaves de API
Especificação do Requisito	
A API deve permitir o gerenciamento das chaves de API que serão utilizadas pelos demais sistemas integrados para autenticar a conexão.	

Tabela 15 - RF02 - API *Whatsapp*: Armazenamento de dados

Identificação do Requisito:	RF02
Nome do Requisito	Armazenamento de dados

Especificação do Requisito
A API deve armazenar os dados relativos às instâncias cadastradas e api-keys do sistema utilizando MongoDB.

Tabela 16 - RF03 - API Whatsapp: Criação de instância Whatsapp

Identificação do Requisito:	RF03
Nome do Requisito	Criação de instância Whatsapp
Especificação do Requisito	
A API deve possibilitar a criação de instâncias *Whatsapp* quando solicitado pela API *back-end*, iniciando a geração do QR Code de conexão por padrão.	

Tabela 17 - RF04 - API Whatsapp: Gerenciamento de Mensagens

Identificação do Requisito:	RF04
Nome do Requisito	Gerenciamento de Mensagens
Especificação do Requisito	
A API deve manipular envio e recebimento de arquivos e mensagens de texto de acordo com as requisições recebidas e eventos emitidos pelo whatsapp-web.js.	

Tabela 18 - RF05 - API Whatsapp: Conversão de áudios e vídeos em texto

Identificação do Requisito:	RF05
Nome do Requisito	Conversão de áudios e vídeos em texto
Especificação do Requisito	
Quando o evento emitido for de um arquivo o mesmo deve ser baixado, convertido em OGG e enviado para o *Whisper*, que deve fazer a conversão em texto antes do sistema colocar a mensagem na fila.	

3.2.2 Requisitos Não Funcionais:

3.2.2.1 API Whatsapp

Tabela 19 - RNF01 - API Whatsapp: Fila de mensagens

Identificação do Requisito:	RNF01
Nome do Requisito	Fila de mensagens
Especificação do Requisito	
Sempre que o whatsapp-web.js emitir um evento de mensagem a mensagem deve ser colocada na fila, a mensagem deve ser enviada depois de 10 segundos e caso outra mensagem do mesmo cliente entre na fila antes desse tempo elas devem ser concatenadas e enviadas numa única requisição para o webhook da API *back-end*.	

Tabela 20 - RNF02 - API Whatsapp: Parar de gerar QR Code após 5 tentativas

Identificação do Requisito:	RNF02
Nome do Requisito	Parar de gerar QR Code após 5 tentativas
Especificação do Requisito	
Quando uma instância estiver em processo de conexão, não devem ser gerados mais de 5 códigos QR para conexão, para evitar travamentos a geração só é iniciada novamente quando solicitado.	

3.2.2.2 API Whatsapp e API Back-end

Tabela 21 - Requisito Não Funcional 01: Disponibilidade

Identificação do Requisito:	RNF01
Nome do Requisito	Disponibilidade
Especificação do Requisito	
Especificação do Requisito As APIs devem operar continuamente (24/7), permitindo que os clientes enviem mensagens e recebam respostas da IA a qualquer momento.	

Tabela 22 - Requisito Não Funcional 02: Desempenho

Identificação do Requisito:	RNF02
Nome do Requisito	Desempenho
Especificação do Requisito	
A API deve ser capaz de lidar com um aumento no número de usuários e bots, garantindo um bom desempenho mesmo em picos de demanda.	

Tabela 23 - Requisito Não Funcional 03: Medidas de Segurança de Dados

Identificação do Requisito:	RNF03
Nome do Requisito	Medidas de Segurança de Dados
Especificação do Requisito	
A API deve implementar medidas de segurança robustas, incluindo a criptografia de dados em trânsito e em repouso, para proteger a confidencialidade e integridade das informações.	

Tabela 24 - Requisito Não Funcional 04: Manutenção e Atualização do Sistema

Identificação do Requisito:	RNF04
Nome do Requisito	Manutenção e Atualização do Sistema
Especificação do Requisito	
As APIs devem ser projetadas de forma a facilitar a manutenção e atualizações, permitindo evoluções contínuas e eficientes.	

3.3 ESTRUTURA DO SISTEMA

3.3.1 Arquitetura da integração entre as APIs

A arquitetura das APIs foi estruturada atendendo à comunicação eficaz entre as apis internas e externas, a API *WhatsApp* foi desenvolvida em TypeScript utilizando os frameworks Nest.js e Whatsapp-web.js, a qual é responsável pelas conexões, seu envio e recebimento de mensagens, a API ainda realiza a conversão de mídia nas mensagens em texto utilizando a API do *Whisper* e gerencia uma fila de mensagens para melhor funcionamento do sistema, quando a mensagem chega ao fim da fila um webhook é enviado ao *back-end,* que realiza o processamento da mensagem por meio de integração com Chat GPT e a envia por meio de requisição para API *Whatsapp*. Desta maneira, ambas as APIs atuam em comunicação através dos *webhooks*, garantindo comunicação em tempo real e processamento rápido desses dados. Esta modularidade possibilita maior escalabilidade do sistema, segurança, manutenção e atende ao requisito principal da automatização e personalização de bots para *WhatsApp*. Assim, deve-se garantir a robustez e eficiência no fluxo de trabalho, com cada API trabalhando independentemente e cuidando de suas tarefas.

3.3.2 Webhooks

Os *Webhooks* são fundamentais na mudança eficiente de informações entre as duas aplicações. Os *webhooks* são uma notificação enviada para uma URL (Uniform Resource Locator) quando um evento específico ocorre em tempo real, com a vantagem de não pedir a checagem constante do sistema que deseja notificar, isso se dá ao ponto de envio ou *endpoint,* uma URL que é designada a receber os eventos, para que o sistema possa ativar funções sem necessidade de requisições intervaladas. Neste sistema especificamente há três pontos em que os *webhooks* são aplicados:

3.3.2.1 Webhooks da aplicação

O sistema possui três *webhooks* principais que fazer comunicação entre as apis, os quais serão descritos abaixo e posteriormente apresentados em um fluxograma:

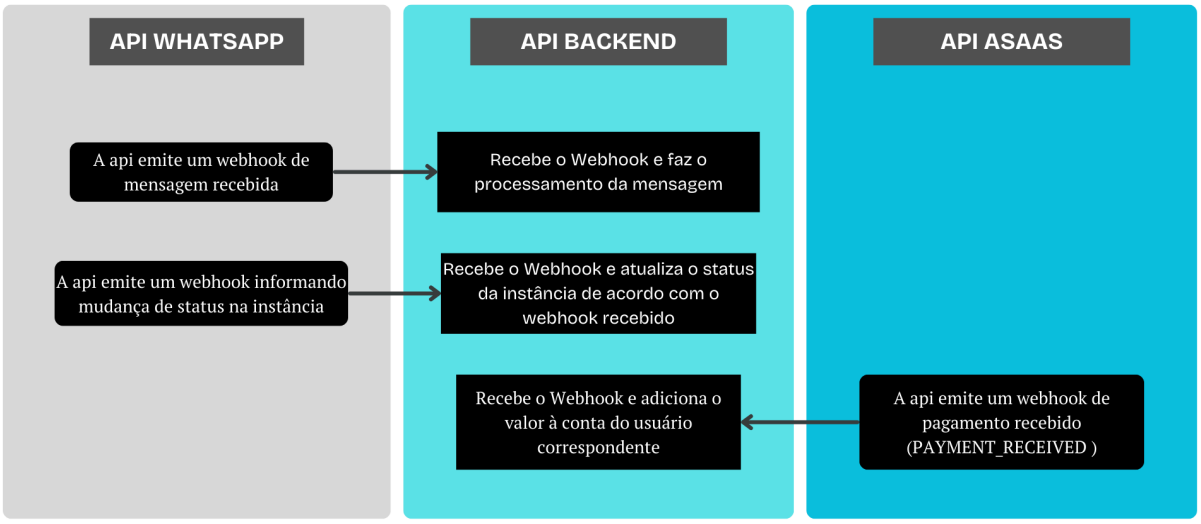
3.3.2.1.1 **Notificação de nova mensagem:** Após receber uma mensagem em um *whatsapp* conectado e a mensagem chegar ao fim da fila um *Webhook* é enviado para a api *back-end*, que realiza os procedimentos necessários para a obtenção da resposta.

3.3.2.1.2 **Notificação de pagamento recebido:** o segundo ponto de aplicação ocorre quando um pagamento por pix é realizado. Assim, quando uma transferência é realizada, é enviado um *Webhook* da Asaas para o *back-end*, que consulta as informações do pagante e atualiza seu saldo, assim as informações da transação são registradas com precisão e celeridade.

3.3.2.1.3 **Notificação de status de conexão do *whatsapp* alterado:** o terceiro ponto de aplicação ocorre quando uma das instâncias na api *Whatsapp* muda seu status, o que emite um *webhook* para a api

back-end que, por sua vez, realiza a alteração do status, garantindo que os status no banco principal estejam sempre sincronizados com o status real da instância.

Figura 1: Demonstração de *Webhooks* do sistema



Fonte: arquivo pessoal

3.3.3 API WhatsApp

A API foi desenvolvida utilizando TypeScript, baseado no Nest.js, porque organiza o sistema de forma modular e eficiente, permitindo assim que o sistema crie e se mantenha facilmente escalável. Utilizando a biblioteca whatsapp-web.js, o envio e recebimento de mensagens em tempo real são gerenciados para garantir interações rápidas e confiáveis. Baseando as operações no banco de dados não relacional com o MongoDB, que tem uma alta velocidade nas consultas e operações de atualização, além de garantir certa liberdade de como o sistema deve ser gerenciado, juntamente com a clareza e a segurança do TypeScript.

3.3.4 API Back-end

O *back-end* foi desenvolvido com Node.js, e o banco de dados relacional MySQL, suportado pelo Sequelize, uma interface de dados intuitiva e organizada para tornar o sistema fácil de manter e escalar. É

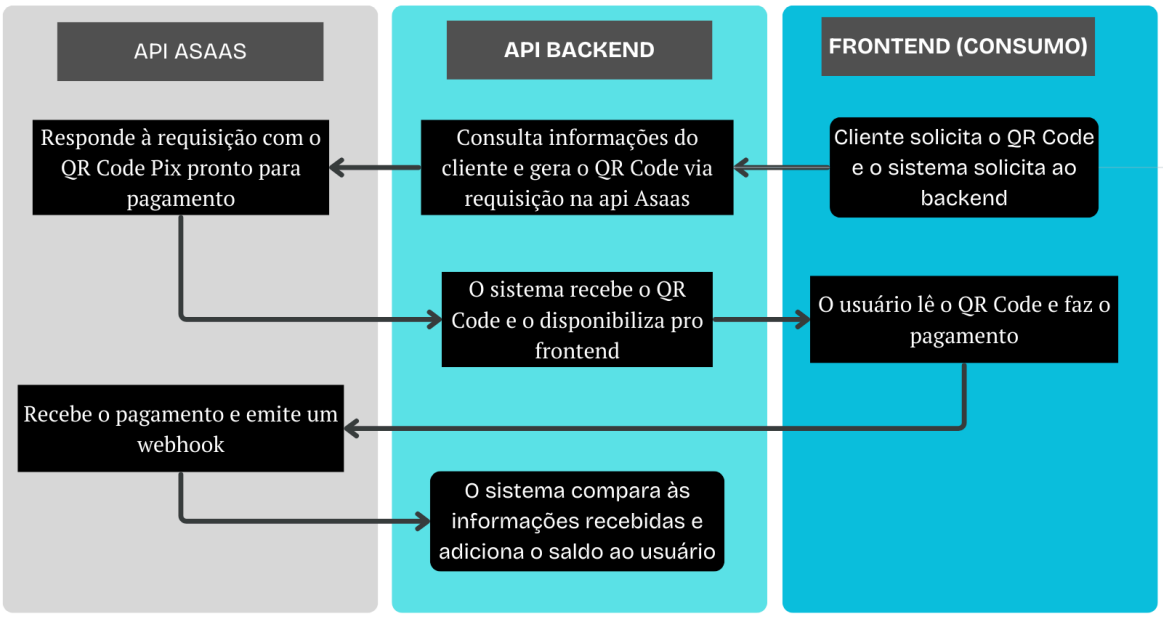
ideal a utilização do Node.js com o banco de dados MySQL para garantir a eficiência no desempenho do sistema, mesmo para cenários de altos volumes de dados.

3.3.5 Comunicação entre as APIs

A comunicação entre os módulos da API *WhatsApp* e o *back-end* foi possível por meio de endpoints RESTful e emissão de *webhooks*, proporcionando a troca de informações de forma consistente e segura. Isso ocorreu devido à modularização, ou seja, a divisão dos subsistemas atômicos que atuam de forma independente, isso garante posterior expansão e atualização do domínio sem violar a totalidade do sistema. O desenvolvimento em si foi realizado de acordo com todas as boas práticas de programação: versionamento, variáveis de ambiente, chaves para comunicação entre as APIs, garantindo a qualidade e confiabilidade do software desenvolvido.

Assim, o sistema desenvolvido consiste em duas APIs que se comunicam com APIs externas para fazer: transcrições de áudios e vídeos, processamento de mensagens com IA e adição de saldo via pix. Os fluxogramas a seguir destacam o fluxo de dados entre as APIs externas e internas, mostrando a tomada de decisões e o processo em cada etapa dos fluxos de mensagens e adição de saldo. Os esquemas foram construídos com o objetivo de mostrar a interação dos componentes entre si e com a lógica do sistema para fornecer a clareza do trabalho, o primeiro detalha o fluxo de adição de saldo via pix utilizando a api do Assas com *webhook* de pagamento configurado:

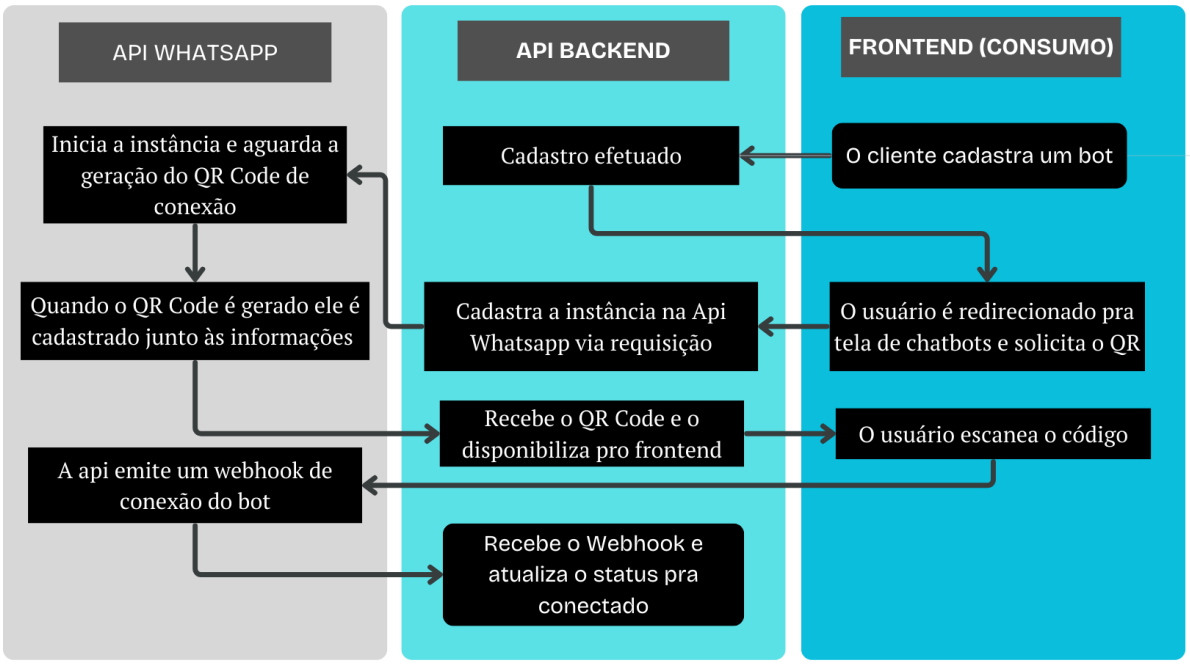
Figura 2: Fluxograma de adição de saldo via pix



Fonte: Arquivo Pessoal

O segundo fluxograma detalha o fluxo de criação e conexão de um bot utilizando QR Code. O processo se inicia no *front-end* com a criação do bot e vai até o envio do webhook pela api *Whatsapp* informando que o bot foi conectado:

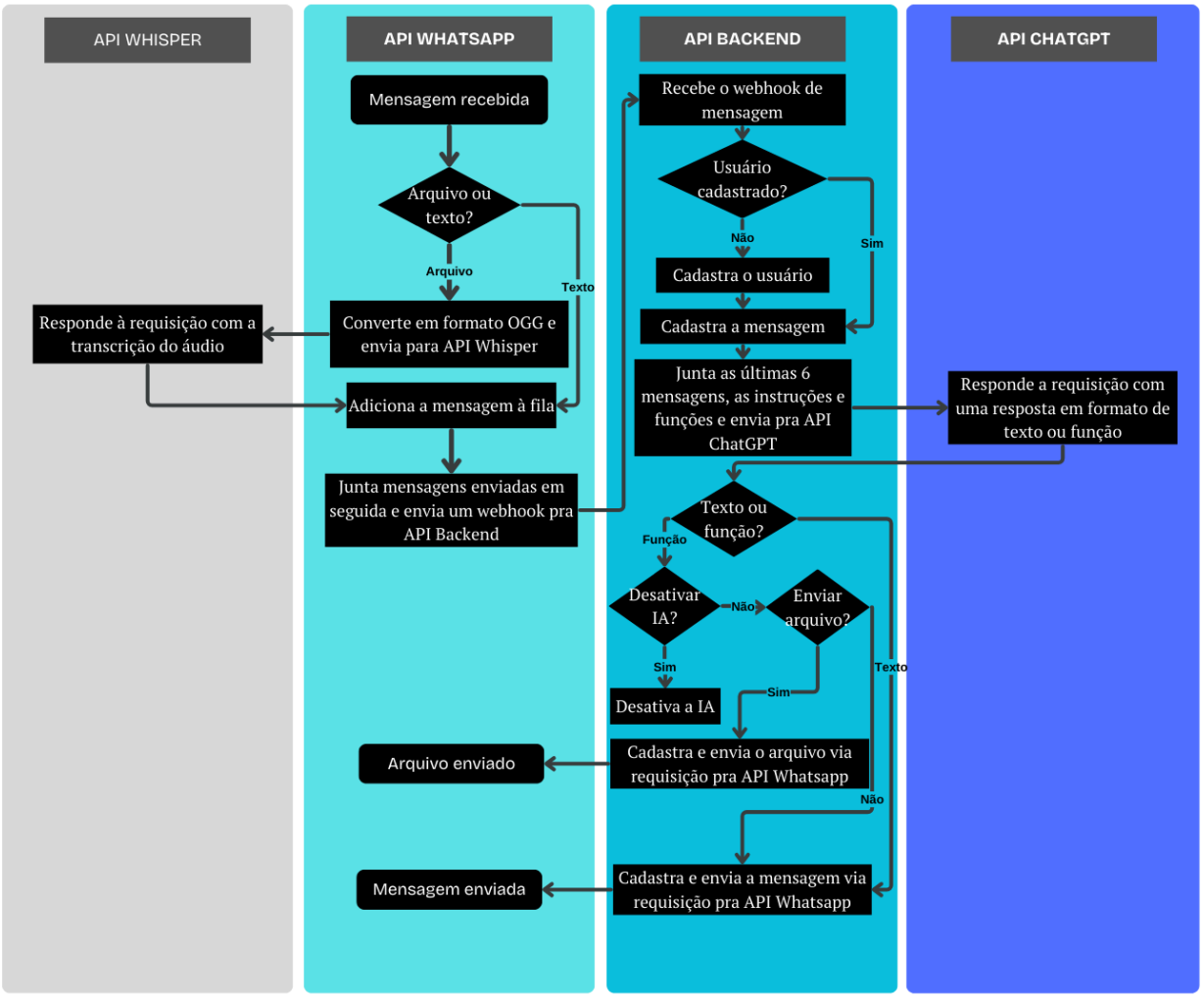
Figura 3: Fluxograma do cadastro e conexão dos bots



Fonte: Arquivo Pessoal

Já o terceiro fluxograma apresenta um fluxo mais complexo, composto de quatro APIs internas e externas demonstrando o processo de recebimento, processamento e resposta à uma mensagem, podendo ser essa resposta um texto ou arquivo, de acordo com as instruções e funções cadastradas na base de conhecimento, durante o processo, o sistema busca apenas as 6 últimas mensagens do usuário para não confundir o bot e manter o mesmo no assunto discutido no momento:

Figura 4: Fluxograma do recebimento e envio de mensagens



Fonte: Arquivo Pessoal

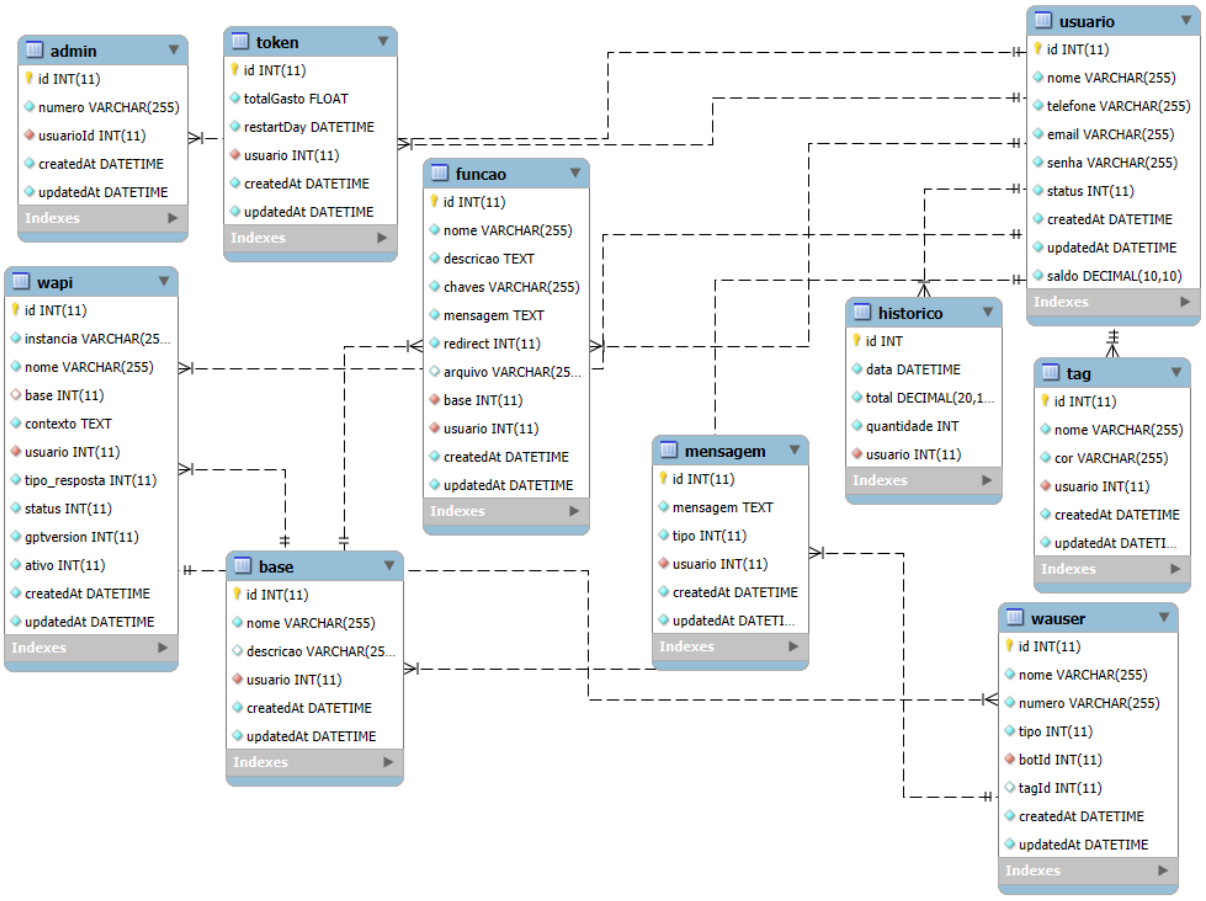
3.4 Diagrama de Entidade e Relacionamento

O Diagrama Entidade-Relacionamento (DER) é uma técnica de modelagem que utiliza uma representação visual para descrever entidades

dentro de um sistema, o que facilita tanto a compreensão do cliente quanto a do profissional que pretende desenvolver o banco de dados. O diagrama permite a representação clara das entidades e relacionamentos, assim como suas cardinalidades, o que acaba sendo essencial para a precisão no desenvolvimento de sistemas de informações complexos (Franck ; Pereira ; Filho, 2021).

A figura a seguir mostra a ilustração gráfica através do DER, que foi gerado utilizando o software *Mysql Workbench*, onde o modelo proposto está presente, utilizado para gerir o software em questão.

Figura 5: Diagrama de Entidade e Relacionamento (DER)



Fonte: Arquivo pessoal.

3.5 Softwares Utilizados

Os seguintes softwares foram utilizados , no desenvolvimento do projeto:

- Documentação TCC: Google Docs <https://docs.google.com>;
- Modelagem de Fluxogramas: Canva <https://www.canva.com>;

- Codificação: Visual Studio Code;
- Banco de Dados: Mariadb e MongoDB;
- Exportação do DER: MySql Workbench.

4 RESULTADOS

Com a implantação da API consumida pelo *front-end* Doumi, foi desenvolvido um sistema completo de atendimento automatizado, melhorando o ritmo e qualidade no atendimento ao público e possivelmente evitando a desistência do cliente devido a demoras no atendimento ou dúvidas não respondidas.

Satisfazendo a necessidade dos patrocinadores no décimo mês da parceria, o sistema completo foi atualizado e disponibilizado em 3 domínios diferentes, o valor total investido nas APIs *back-end* foi de vinte e sete mil e quinhentos reais pagos divididos nos dez meses. Os investidores disponibilizaram o sistema para seus clientes com diferentes nomes, sendo eles a SmartF5 com 14 usuários, a Doumi com 21 usuários e o AtendeADV com 63 usuários na data de entrega deste relato, desde o encerramento da parceria para desenvolvimento o sistema continuou evoluindo para prevenir erros e possibilitar mais funcionalidades que não foram solicitadas pelos patrocinadores, apenas o AtendeADV recebe as atualizações no momento.

Pela natureza do sistema, não puderam ser realizados testes automatizados, já que necessitaria de diversos dispositivos com *Whatsapp* conectado, mas os testes realizados em produção demonstraram grande eficácia nos três servidores onde o sistema foi instalado, um deles se destacou chegando a manter 41 bots conectados, o AtendeADV e possibilitou a observação do funcionamento da aplicação com alta demanda, chegando a processar mais de 500 respostas por dia sem erros no processamento, com pequenos atrasos na entrega de mensagens em horários de grande demanda.

A implementação da API do *WhatsApp* utilizando whatsapp-web.js permitiu o gerenciamento de mensagens em tempo real, enquanto o *back-end* desenvolvido com Node.js, Express e o ORM Sequelize proporcionou uma interface eficiente para manipulação e organização dos dados recebidos via webhook. Essa integração resultou em um sistema modular, organizado e capaz de atender a demandas tanto atuais quanto futuras.

Como perspectivas futuras, é possível aprimorar o sistema com a implementação de novas funcionalidades, como a utilização de parâmetros nas funções GPT para colher dados e fazer o envio para um número cadastrado ou um *endpoint* escolhido pelo cliente. Por fim, a integração com outras plataformas de comunicação poderia ampliar ainda mais o alcance e a usabilidade do sistema, consolidando ainda mais sua relevância em diferentes contextos de aplicação.

5 CONCLUSÃO

Este estudo descreveu o progresso na criação de APIs REST utilizando frameworks Javascript com a finalidade de criação e gerenciamento de bots automatizados no Whatsapp. A utilização da API do *WhatsApp* com whatsapp-web.js mostrou ser eficaz na gestão de mensagens, enquanto o *back-end* através do Sequelize assegurou uma organização estruturada e ampla na manipulação dos dados. Práticas de desenvolvimento eficientes incluíam o uso de módulos independentes, variáveis de ambiente, versionamento com Git e testes detalhados para garantir qualidade e facilitar futuras melhorias no sistema.

Mesmo que tenha alcançado os objetivos estabelecidos inicialmente em termos gerais, o projeto enfrentou obstáculos como depender de tecnologias específicas como a biblioteca whatsapp-web.js e precisar de uma otimização mais eficiente em situações de alta demanda, o que sugere espaço para melhorias no futuro, com possibilidade de integração com outras plataformas e adoção de novas tecnologias, contudo, o sistema criado fornece uma base robusta para projetos vindouros e contribui para o avanço das soluções tecnológicas focadas em comunicação e processamento de dados em tempo real.

Atualmente, o AtendeADV está em processo de atualização e irá receber novas funcionalidades em breve. Entre as melhorias previstas estão a adição de coleta e envio de dados às funções GPT da base de conhecimento, a implementação de uma frase de iniciação do atendimento, a introdução de comandos de ativação e desativação por meio de mensagens no WhatsApp, e a disponibilização de modelos de bots acessíveis a todos os usuários.

6 REFERÊNCIAS

ABREU, Matheus Silva de. **WhatsDine: solução para o delivery integrada ao WhatsApp.** Universidade Estadual Paulista (Unesp). 2023.

BARSOTI, Nathan; GIBERTONI, Daniela. **Impacto que o sequelize traz para o desenvolvimento de uma api construída em node. js com express. js**. Revista Interface Tecnológica, v. 17, n. 2, p. 231-243, 2020.

BUMANN, Natal. **Automated Chatbot Using Speech-to-Text and Text-to-Speech with Mobile App Integration.** HES-SO. 2023.

CAETANO, Caio Alexandre Troti. **Loolabae: aplicativo de leitura de texto e reprodução de efeitos sonoros por inteligência artificial**. Instituto de Ciência e Tecnologia, Universidade Estadual Paulista, Sorocaba, 2024.

CARMO, Klayver Ximenes. **Um estudo comparativo entre tecnologias de back-end: Node. js, Django REST Framework e ASP. NET Core.** Campus de Sobral, Universidade Federal do Ceará. 2023.

CUNHA, Bruna Carolina Rodrigues da. **Desenvolvimento full-stack com Javascript: uma visão geral e prática.** XXIX Simpósio Brasileiro de Sistemas Multimídia e Web: Minicursos. 2024.

DE ALMEIDA AMBRIOSO, Carlota Duarte Ferreira. **A Inteligência ARTIFICIAL (IA) NO MARKETING.** 2024. Tese de Doutorado. Universidade de Coimbra.

SOUZA, Kleiton Carlos De. **Implementação de um Chatbot com GPT-3.5 para Melhoria do Atendimento ao Cliente em Telecomunicações.** Instituto Federal De Santa Catarina. 2023.

FRANCK, Kewry Mariobo; PEREIRA, Robson Fernandes; FILHO, Jerônimo Vieira Dantas. **Diagrama Entidade-Relacionamento: uma ferramenta para modelagem de dados conceituais em Engenharia de Software.** Research, Society and Development, v. 10, n. 8, p. e49510817776-e49510817776, 2021.

FUJIWARA, Bruno Keichi. **Análise de metodologias e estratégias de testes de API rest: um estudo de caso.** Universidade Federal De São Carlos. 2023.

JUNIOR, Edemilton Alcides Galindo; ROCHA, Romeu Dias; DE SOUZA MACIEL, Ronierison. **Desenvolvimento de api rest com spring boot.** RIOS - Revista Científica do Centro Universitário do Rio São Francisco. 2021.

LANZ, Anna Elly; BRILINGER, Caroline Orlandi. **Automatização do atendimento ao cliente via aplicativo de mensagens em uma operadora de planos privados de saúde.** Brazilian Journal of Development, v. 8, n. 3, p. 221801-22189, 2022.

MONTANHEIRO, Lucas Souza; CARVALHO, Ana Maria Martins; RODRIGUES, Jackson Alves. **Utilização de json web token na autenticação de usuários em apis rest.** XIII Encontro Anual de Computação. Anais do XIII Encontro Anual de Computação EnAComp, p. 186-193, 2017.

NASCIMENTO, Diego Felipe Gonçalves do et al. **Projeto de Aplicação Chatbot Web para o gerenciamento de chamados de atendimento: um estudo de caso na UFPB.** Universidade Federal da Paraíba. 2024.

NICOLESCU, Luminița; TUDORACHE, Monica Teodora. **Human-computer interaction in customer service: the experience with AI chatbots—a systematic literature review.** Electronics, v. 11, n. 10, p. 1579, 2022.

O GLOBO. **Uso de WhatsApp impacta vendas e atendimento das empresas.** Disponível em: <https://oglobo.globo.com/patrocinado/dino/noticia/2024/09/04/uso-de-whatsapp-impacta-vendas-e-atendimento-das-empresas.ghtml>. Acesso em: 3 dez. 2024.

OLIVEIRA, Ana Karolina Santos; LIMA, Edilson Carlos Silva; DE SENA CARIDADE, Elda Regina. **O USO DOS FRAMEWORK FLUTTER E NEST. JS PARA DESENVOLVIMENTO DE UM FRONT-END E UM BACK-END DE UMA APLICAÇÃO DE HELP DESK.** Revista Ibero-Americana de Humanidades, Ciências e Educação, v. 8, n. 11, p. 1766-1786, 2022.

QUEIROZ, Lucas Grisi Oliveira De. **Funções de Hash e Gerenciamento de Senhas.** Monografia. Universidade Federal de Pernambuco. 2022.

REIS, Julio Cesar Dos; DELBUONO, Enrico. **Transcrição Automática de Vídeos para Textos em Linguagem Natural.** Universidade Estadual De Campinas - Instituto De Computação. 2023.

REGIN, R.; RAJEST, S. Suman. **A Chatbot that uses Artificial Intelligence to Automatically Respond to Whatsapp Messages.** AMERICAN Journal of Engineering, Mechanics and Architecture. 2024.

ROCHA, Gustavo Clemente Colombo da. **Uma Análise comparativa entre frameworks de persistência de dados em javascript.** Universidade Federal do Ceará, Campus de Quixadá. 2022.

SPINEI, Alexia. **Artificial friends: A ChatGPT implementation in AlphaMini companionship robots.** Tese de Doutorado. Universidade de Groningen. 2024.