

Douglas de Souza Carvalho

Desenvolvimento de um Chatbot para Marcação de Consultas em Clínicas

Natal – RN

Dezembro de 2022

Douglas de Souza Carvalho

Desenvolvimento de um Chatbot para Marcação de Consultas em Clínicas

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Diogo Pinheiro Fernandes Pedrosa

Universidade Federal do Rio Grande do Norte – UFRN
Departamento de Engenharia de Computação e Automação – DCA
Curso de Engenharia de Computação

Natal – RN
Dezembro de 2022

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Carvalho, Douglas de Souza.

Desenvolvimento de um chatbot para marcação de consultas em clínicas / Douglas de Souza Carvalho. - 2022.

50 f.: il.

Monografia (graduação) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Curso de Engenharia de Computação. Natal, RN, 2022.

Orientador: Prof. Dr. Diogo Pinheiro Fernandes Pedrosa.

1. Processamento de linguagem natural - Monografia. 2. Chatbots - Monografia. 3. ApacheNLP - Monografia. I. Pedrosa, Diogo Pinheiro Fernandes. II. Título.

RN/UF/BCZM

CDU 004.934.1

Douglas de Souza Carvalho

Desenvolvimento de um Chatbot para Marcação de Consultas em Clínicas

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Diogo Pinheiro Fernandes Pedrosa

Trabalho aprovado. Natal – RN, 21 de Dezembro de 2022:

Prof. Dr. Diogo Pinheiro Fernandes Pedrosa - Orientador
UFRN

Prof. Dr. Carlos Manuel Dias Viegas - Banca
UFRN

Prof. Dr. Anderson Luiz De Oliveira Cavalcanti - Banca
UFRN

Natal – RN
Dezembro de 2022

Dedico este trabalho aos meus pais, sem seu amparo e exemplo eu não seria nada. Muito obrigado por tudo.

AGRADECIMENTOS

Aos meus pais Leda Cristina e Wilson que são meus maiores incentivadores, obrigado por todo carinho, amor e investimento em minha educação, os senhores tornaram este caminho muito mais fácil. E também ao meu irmão William por todo o companheirismo.

Ao meu professor orientador Diogo, que com muita paciência e grande empatia me orientou nesta monografia, obrigado por todos os ensinamentos.

A todos os meus colegas de graduação, com quem convivi tantos anos e com quem compartilhei vários momentos de angústia e felicidade.

Ao Gleydson por me permitir conhecer o projeto do Quarkclinic e sempre disposto a ajudar e orientar, e a todos os meus colegas da Quark em especial a minha equipe do QuarkClinic que são excelentes profissionais e amigos.

“Se vi mais longe, foi por estar de pé sobre ombros de gigantes.”
(Isaac Newton)

RESUMO

Este trabalho apresenta um chatbot capaz de ajudar o público principal de clínicas de saúde particulares, os pacientes, à realizar agendamentos de consultas médicas em integração ao QuarkClinic, sistema de gerenciamento para essas instituições. A aplicação consiste em uma interface virtual no formato de chat, localizado em um portal de agendamentos online, em que os pacientes podem interagir textualmente com o chatbot. Poderão ser feitas várias interações, tais como: Marcação de consultas em determinado horário, cancelamento de consultas e listagem do histórico de marcações. O intuito é aumentar a praticidade e velocidade no agendamento de consultas sem a necessidade de atendimento humano dedicado. Para este trabalho foram utilizadas linguagens de programação como Javascript, Java e seus frameworks Vue e Spring Boot, além do ApacheNLP, biblioteca baseado em aprendizado de máquina para processamento de texto baseado em linguagem natural.

Palavras-chaves: Chatbots. Web. Vue. Spring-boot. ApacheNLP. Processamento de linguagem natural.

ABSTRACT

This work presents a chatbot capable of helping the main public of private health clinics, the patients, to schedule medical appointments in integration with QuarkClinic, a management system for these institutions. The application consists of a virtual interface in chat format, located in an online scheduling portal, where patients can interact textually with the chatbot. Several interactions can be made, such as: Scheduling consultations at a certain time, canceling consultations and listing the history of appointments. The aim is to increase the practicality and speed in scheduling appointments without the need for dedicated human service. For this work, programming languages such as Javascript, Java and its Vue and Spring Boot frameworks were used, in addition to Apache NLP, a library based on machine learning for text processing based on natural language.

Keywords: Chatbots. Web. Vue. Spring-Boot. ApacheNLP. Natural Language Processing.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura do sistema proposto	24
Figura 2 – Diagrama de classes	25
Figura 3 – Arquivos do microsserviço de conversação	25
Figura 4 – Arquivos do microsserviço de agendamento	26
Figura 5 – Arquivo com dados de treinamento para o modelo	27
Figura 6 – Arquivos e estrutura do FrontEnd Vue	27
Figura 7 – Fluxograma de conversação para marcar consulta	28
Figura 8 – Fluxograma de conversação para listar histórico de consulta	29
Figura 9 – Fluxograma de conversação para cancelar consulta	29
Figura 10 – Função 'contextLoads'	30
Figura 11 – Código da função de tokenização	31
Figura 12 – Resultado da tokenização para a frase: oi, bom dia	31
Figura 13 – Código da função de POS Tagger	31
Figura 14 – Código da função de lematização	32
Figura 15 – Resultado da função de lematização	32
Figura 16 – Código para treinamento do modelo	32
Figura 17 – Código da função de categorização	33
Figura 18 – Resultado com as probabilidades da função de categorizar	33
Figura 19 – Interface inicial do Portal de Agendamento Online com o ChatBot	34
Figura 20 – Fluxo de saudação	35
Figura 21 – Resultado da requisição do fluxo de saudação	35
Figura 22 – Escolha de convênio no fluxo de marcação de consulta	36
Figura 23 – Resultado da requisição da escolha do convênio	37
Figura 24 – Escolha de procedimento no fluxo de marcação de consulta	38
Figura 25 – Resultado da requisição da escolha do procedimento	38
Figura 26 – Escolha de clínica no fluxo de marcação de consulta	39
Figura 27 – Resultado da requisição da escolha da clínica	39
Figura 28 – Escolha do profissional no fluxo de marcação de consulta	40
Figura 29 – Resultado da requisição da escolha do profissional	40
Figura 30 – Escolha da agenda no fluxo de marcação de consulta	41
Figura 31 – Resultado da requisição da escolha da agenda	41
Figura 32 – Escolha do horário no fluxo de marcação de consulta	42
Figura 33 – Resultado da requisição da escolha do horário	43
Figura 34 – Resultado da requisição da confirmação da consulta	44
Figura 35 – Agendamento criado no sistema QuarkClinic	44

Figura 36 – Fluxo para histórico de consultas	45
Figura 37 – Fluxo para cancelamento de consultas	46
Figura 38 – Agendamento cancelado no sistema QuarkClinic	47
Figura 39 – Fluxo de mensagem não identificada	47

LISTA DE TABELAS

Tabela 1 – Verbos HTTP	21
Tabela 2 – Status HTTP mais comuns	21
Tabela 3 – Rotas e verbos HTTP da API	28

LISTA DE ABREVIATURAS E SIGLAS

HTML	<i>HyperText Markup Language</i>
CSS	<i>Cascading Style Sheets</i>
DOM	<i>Document Object Model</i>
JSON	<i>JavaScript Object Notation</i>
API	<i>Application Programming Interface</i>
REST	<i>Representational State Transfer</i>
HTTP	<i>Hypertext Transfer Protocol</i>
SPA	<i>Single-Page Application</i>
XML	<i>Extensible Markup Language</i>
NLP	<i>Natural Language Processing</i>
DTO	<i>Data Transfer Object</i>
POO	<i>Programação Orientada a Objetos</i>
SGBD	<i>Sistema de Gerenciamento de Banco de Dados</i>

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Motivação	14
1.2	Objetivos	14
1.2.1	Objetivo Geral	15
1.2.2	Objetivos Específicos	15
1.3	Metodologia	15
1.4	Estrutura do Trabalho	15
2	REVISÃO BIBLIOGRÁFICA	17
2.1	<i>Linguagens de programação</i>	17
2.1.1	Javascript	17
2.1.2	Java	17
2.2	<i>Frameworks</i>	18
2.2.1	Vue	18
2.2.2	Spring Boot	18
2.3	<i>Banco de dados</i>	18
2.3.1	PostgreSQL	18
2.4	<i>Versionamento de código</i>	19
2.4.1	Git	19
2.4.2	Github	19
2.5	<i>Arquitetura de software</i>	20
2.5.1	FrontEnd	20
2.5.2	Backend	20
2.5.3	Microserviços	20
2.6	<i>Protocolo HTTP</i>	20
2.7	<i>Rest API</i>	21
2.8	<i>Processamento de Linguagem Natural</i>	22
2.9	<i>OpenNLP</i>	22
2.10	<i>QuarkClinic</i>	23
2.10.1	Agendamento Online	23
3	METODOLOGIA	24
3.1	Modelagem de dados	24
3.2	Estruturação da aplicação	25
3.3	Comunicação entre os microserviços	27

3.4	Rotas da API	28
3.5	Fluxos de conversação	28
4	RESULTADOS	30
4.1	Categorização	30
4.1.1	Tokenização	30
4.1.2	POS tagger	31
4.1.3	Lematização	31
4.1.4	Treinando modelo	32
4.1.5	Categorizando sentença	32
4.2	Interface do Chat	33
4.3	Fluxos de conversação via Chat	34
4.3.1	Saudação	34
4.3.2	Marcar consulta	35
4.3.3	Histórico de consultas	45
4.3.4	Cancelar consulta	45
4.3.5	Fallback	47
4.4	Endpoint para PLN	48
5	CONCLUSÃO	49
5.1	Trabalhos futuros	49
	REFERÊNCIAS	50

1 INTRODUÇÃO

A segunda metade do século vinte foi marcada como o início da era dos computadores e grandes avanços tecnológicos que perduram até os tempos atuais. Em paralelo a isso houve grandes avanços da medicina e no acesso à saúde. Clínicas que servem em diferentes áreas da saúde foram se espalhando cada vez mais e, junto a isso, tendo que lidar com a grande demanda por pacientes.

Assistentes virtuais ou Chatbots se destacaram ao automatizar o atendimento ao público interessado em ter sua demanda atendida. Como se sabe, em empresas, incluindo clínicas, há uma área totalmente dedicada ao atendimento a clientes, que é a razão da existência desses estabelecimentos.

Segundo (CRUZ L.; ALENCAR, 2018) assistentes virtuais inteligentes são programas de computador projetados para interagir com clientes de uma empresa em linguagem natural. Assim, é possível estabelecer uma relação menos mecanizada e de forma mais simples. Esse tipo de abordagem é replicado em várias áreas: direito, finanças, comércio eletrônico, ensino, turismo, saúde e bem estar, em instituições governamentais, etc. A expectativa, com o avanço da tecnologia e da necessidade de atendimento de clientes, é que a adoção deles fique cada vez maior em vários setores.

1.1 Motivação

Com os diversos benefícios apontados anteriormente, em sistemas de informática que provêm serviços para clínicas de saúde, é notado uma grande falta deste tipo de solução, mesmo sendo um recurso requisitado pelos contratantes. Pensando nisso, é proposto o desenvolvimento de uma solução em formato de Chatbot que possa atender essa demanda, usando como plataforma o sistema de gestão de clínicas QuarkClinic e seu ambiente de interação direta com pacientes, o Agendamento Online.

1.2 Objetivos

Neste trabalho, os objetivos foram divididos e subdivididos em objetivo geral e específicos, como apresentado a seguir.

1.2.1 Objetivo Geral

- Desenvolver um chatbot que possa realizar consultas, cancelamentos e listar o histórico de consultas diretamente por uma interface de conversação em linguagem natural (PLN - Processamento de Linguagem Natural).

1.2.2 Objetivos Específicos

- Utilização de linguagens e frameworks modernos para desenvolvimento de um frontend e um backend de baixa latência, simples e eficaz;
- Modelagem eficiente da arquitetura da solução, de forma que seja escalável e coeso;
- Cumprir requisitos não funcionais para melhorar eficiência, velocidade e tempo de resposta da aplicação;
- Contribuir com administradores de clínicas na captação de pacientes de forma menos trabalhosa e eficiente;
- Contribuir com os pacientes que estão em busca de clínicas para atendimento e tem que lidar com burocracias desnecessárias;

1.3 Metodologia

O trabalho inicia-se com a concepção da arquitetura da solução e escolha das ferramentas de construção de código que melhor suportem essa solução e, que integram bem com o ecossistema já existente do QuarkClinic. Além disso, busca-se o desenvolvimento de um backend que possa lidar com vários fluxos de conversa a fim de poder suprir a necessidade do paciente. Por fim, fazer com que a ferramenta desenvolvida possa ser suficientemente versátil e, assim, ser reutilizada para outros propósitos.

1.4 Estrutura do Trabalho

Este trabalho está estruturado da seguinte forma:

- O capítulo 2 apresenta uma revisão bibliográfica sobre sistemas web, linguagens de programação, processamento de linguagem natural, frameworks utilizados e sobre o QuarkClinic como sistema de gerenciamento de clínicas;
- O capítulo 3 é dedicado a mostrar a concepção e desenvolvimento da solução proposta;

- O capítulo 4 mostra os resultados e discussões obtidos em um caso real de aplicação da solução;
- E, por fim, o capítulo 5 aborda as conclusões deste trabalho e possíveis trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, são apresentados os conhecimentos necessários para o entendimento da solução proposta.

2.1 *Linguagens de programação*

A forma com a qual os engenheiros e cientistas do passado criaram para se comunicar com as máquinas é chamada de linguagem de programação. Arquivos criados com a codificação pertinente à linguagem de programação escolhida são compilados e, assim, tem-se arquivos binários que são executados pelo processador do computador. A programação em computadores da primeira geração era complexa e não-produtiva pois necessitava-se de codificação em linguagem de máquina (inicialmente) e em linguagem de montagem (posteriormente). A partir da segunda geração de computadores houve a criação de linguagens de programação de alto nível e o desenvolvimento de compiladores (TENENBAUM, 2015).

2.1.1 Javascript

Javascript, segundo (FLANAGAN, 2013) é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e funcionais. É modelada inicialmente para ser executada pelos browser (navegadores), ou seja, do lado do cliente, porém com o advento do NodeJS também é possível utilizá-lo do lado do servidor, como linguagem para construção de backends. No contexto de desenvolvimento Web, ela possibilita dar funcionalidades à páginas, inserindo comportamento dinâmico a elas, principalmente manipulando dados e elementos HTML e CSS na DOM.

2.1.2 Java

Java é uma linguagem de programação muito difundida em ambientes empresariais, órgãos de governo e na comunidade em si de programadores (DEITEL P.; DEITEL, 2017). É uma linguagem multiplataforma orientada a objetos, que tem seu binário final interpretado por uma máquina virtual, a JVM. Ela tem sido uma escolha popular entre os desenvolvedores por mais de duas décadas, então é uma linguagem testada pelo tempo e de grande prestígio. Ela é rápida, segura e confiável, utilizada tanto em grandes aplicações empresariais, quanto móveis até para processamento de grande conjuntos de dados (DEITEL P.; DEITEL, 2017).

2.2 Frameworks

Segundo (BARRO, 2022) a principal função de um framework é facilitar o processo de desenvolvimento de um software ou aplicação. Oferecendo uma estrutura básica sobre a qual o sistema pode ser programado, os frameworks representam uma vantagem em termos de tempo e segurança. Esse conjunto de estruturas permite a construção de aplicação com menor esforço e maior rapidez.

2.2.1 Vue

Segundo (VUE, 2022), Vue é um framework progressivo para a construção de interfaces de usuário. Ao contrário de outros frameworks monolíticos, Vue foi projetado desde sua concepção para ser adotável incrementalmente. A biblioteca principal é focada exclusivamente na camada visual, sendo simples a integração com demais bibliotecas no projeto. Vue também é capaz de dar suporte ao desenvolvimento de aplicações do tipo *single page* (VUE, 2022) quando usado em conjunto com ferramentas modernas e bibliotecas de apoio.

2.2.2 Spring Boot

O Spring Boot, segundo (BOAGLIO, 2017) é um framework de código aberto desenvolvido para o Java como plataforma, baseado nos padrões de projetos, inversão de controle e injeção de dependência. Com ele é possível construir backends robustos e seguros.

2.3 Banco de dados

Um banco de dados se caracteriza por ser uma coleção organizada de informações, normalmente armazenadas eletronicamente em um servidor. Segundo (CARVALHO, 2017) a linguagem padrão de acesso a dados é o SQL, que reúne abordagens das linguagens de definição, manipulação e controle dos dados. Há também bancos de dados não relacionais, como o caso do MongoDB, que tem uma base dados flexível que permite guardar dados não estruturados, provendo um suporte completo por indexação (MONGODB, 2022).

2.3.1 PostgreSQL

Uma das principais tecnologias no quesito banco de dados é o postgres. Segundo (CARVALHO, 2017), postgresQL é um sistema gerenciador de banco de dados relacional de código aberto. Ele foi desenvolvido há bastante tempo, sendo amplamente testado pela comunidade, esse SGBD é um dos mais utilizados do mundo pois garante: uma grande

quantidade de recursos e operações possíveis para lidar com os dados, confiabilidade e uma baixa curva de aprendizado. Ele também se destaca, segundo (CARVALHO, 2017):

- Pela segurança: um sistema transacional, evitando assim que qualquer ação destrutiva ou de modificações em estruturas possam ser revertidas sem qualquer prejuízo;
- Pelo poderio: O postgres conta com uma diversidade de extensões como busca gerral de texto, *slow query*, criptografia de senha. Além de manipular diversos tipos de dados como JSON, XML, matrizes etc;
- Pela rapidez: O postgresQL faz uso de estratégias de indexação que otimizam buscas e diminuem o esforço.

2.4 *Versionamento de código*

O versionamento de código, de acordo com (CHACON, 2014), consiste em registrar as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que se possa recuperar versões específicas. Há diversas estratégias para gerenciamento de versões de um código de um sistema, sendo que elas tem por fim a garantia de mais segurança na transição de uma versão para outra.

2.4.1 Git

O Git é um sistema de controle de versão que, segundo (CHACON, 2014), teve sua concepção em 2005 por Linus Torvalds, o criador do Linux, não estando satisfeito com o sistema de controle de versão utilizado no desenvolvimento do kernel do Linux. A utilização do git como ferramenta de versionamento é feita em diversos projetos de código aberto e por empresas de desenvolvimento de softwares. A maior parte das operações do git, segundo (CHACON, 2014), são locais, ou seja, não precisa estar conectado em rede. Caso surja a necessidade de desenvolver coletivamente em um time ou projeto em que vários desenvolvedores tenham que alterar a mesma base de código, será necessário buscar por ferramentas que disponibilizem um servidor git, como o Github.

2.4.2 Github

Para projetos em que é necessária a colaboração no Git, é necessária a utilização de repositórios remotos, em servidores. O Github, segundo (CHACON, 2014) é o maior site de código aberto de hospedagem Git, uma aplicação Web que possibilita a hospedagem de repositórios públicos e privados. Como é centrado no usuários, o Github também serve de rede social para desenvolvedores.

2.5 *Arquitetura de software*

A arquitetura de software, segundo (VALENTE, 2020) pode ter várias definições, uma das usadas é que a arquitetura se preocupa em organizar componentes de mais alto nível, sem se preocupar tanto com interfaces de classes individuais, porém prezando pelos conjuntos de classes relacionadas.

2.5.1 FrontEnd

Segundo (SOUTO, 2022), o front-end classifica-se como a parte visual de um site, aquilo que possibilita a interação com o usuário. O trabalho com Front End tem a responsabilidade por desenvolver, por meio de um código, uma interface gráfica. Normalmente, são usadas as tecnologias base da web (HTML, CSS e JavaScript) assim como vários frameworks (Vue, Angular, React, entre outros).

2.5.2 Backend

O Backend é a parte do sistema que fica do lado do servidor e concentra a maioria das regras de negócio de uma aplicação e se comunica com o banco de dados. Segundo (SOUTO, 2022), o Backend trabalha normalmente relacionando os dados que vêm do navegador para um banco de dados, e vice-versa. O backend aplica as devidas regras de negócios, validações e garantias em um ambiente onde o usuário final não tenha acesso e possa manipular algo. Há uma grande quantidade de linguagens de programação que podem ser utilizadas no desenvolvimento de backends, tais como: Java, Ruby, python, etc.

2.5.3 Microserviços

Microserviços são pequenas fragmentações de software que são executadas de maneira independente. De acordo com (DUARTE, 2018), eles são focados em um único e pequeno conjunto bem definido de atividades dentro de um conjunto maior de serviços, formando assim uma grande arquitetura de micro serviços.

2.6 *Protocolo HTTP*

Segundo (DUARTE, 2018) o HTTP é um protocolo de comunicação utilizado para transferência de hipertextos, que é um texto estruturado que utiliza links. Ele é a base para comunicação na *World Wide Web*. Para cada tipo de requisição segundo (DUARTE, 2018), existem os chamados verbos do HTTP, que são métodos e cada um tem sua função. Os mais comuns são listados na tabela 1.

Tabela 1 – Verbos HTTP

Verbo	Propósito comum
GET	Utilizado para receber informações
POST	Utilizado quando é necessário enviar informações novas para serem adicionadas
PUT	Utilizado para atualizar recursos existentes
DELETE	Utilizado para excluir determinado recurso

Há também as classes de status retornadas pelo protocolo. Segundo (DUARTE, 2018) existem várias e cada uma possui uma classe de números com significados diferentes. As mais comuns são listadas na tabela 2.

Tabela 2 – Status HTTP mais comuns

Código	Explicação
1xx	Informativo, a requisição foi recebida e o processo continua
2xx	Ocorreu tudo certo com a requisição
200	<i>OK</i> , usado quando a requisição foi processada e entregue com sucesso
201	<i>Created</i> , criação foi executada com sucesso
3xx	Redirecionamento, alguma ação precisará ser tomada para completar a requisição
4xx	Erro no cliente, possivelmente a requisição foi mal feita
403	<i>Forbidden</i> , quando o usuário não tem permissão para acessar o recurso
404	<i>Not Found</i> , recurso requisitado não foi encontrado
5xx	Erro no servidor, a requisição parece válida, mas o servidor não consegue processar
500	<i>Internal Server Error</i> , servidor não conseguiu processar a requisição por algum erro inesperado
502	<i>Bad Gateway</i> , usado pelo servidor web que está sendo utilizado como proxy, ou seja, que está servindo as requisições para um ou mais servidores abaixo dele. Caso ele não receba uma resposta válida do servidor que está abaixo, ele retorna o status 502

2.7 Rest API

Uma API, segundo (HAT, 2022) é um conjunto de definições e protocolos usado no desenvolvimento e na integração de aplicações. Em outras palavras, ao interagir com um computador ou sistema para recuperar informações ou executar uma função, a API ajudará a comunicar o que se quer ao sistema para que ele entenda e realize o que foi solicitado, funcionando como um mediador entre os usuários ou clientes e os recursos ou serviços web que eles querem obter. As APIs também servem, segundo (HAT, 2022) para que organizações compartilhem recursos e informações e, ao mesmo tempo, mantenham a segurança, o controle e a obrigatoriedade de autenticação, pois permitem determinar quem tem acesso e o que pode ser acessado.

REST (Representational State Transfer) ou transferência de estado representativo, segundo (SAUDATE, 2013) é um estilo de desenvolvimento de web services que teve origem na tese de doutorado de Roy Fielding, co-criador do protocolo HTTP, portanto é um protocolo guiado, dentre outros preceitos, pelo que seriam as boas práticas de uso do HTTP. Segundo (SAUDATE, 2013) em REST tudo é definido como recursos, sendo estes os conjuntos de dados que são trafegados pelo protocolo.

API REST, também chamada de API RESTful, segundo (HAT, 2020) é uma interface de programação de aplicações (API ou API web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços web RESTful.

2.8 *Processamento de Linguagem Natural*

O Processamento de Linguagem Natural, conhecido também pela sigla PLN, segundo (ALLES, 2018) usa os conhecimentos léxicos, sintáticos e semânticos de um idioma bem como outras informações reais necessárias para determinar os significados das palavras e seus sentidos no contexto aplicado. Como um ramo de aplicação da inteligência artificial a PLN propicia aos computadores a capacidade de entender a língua humana e habilitar uma interação entre o computador e a linguagem natural (ALLES, 2018);

Por meio de técnicas linguísticas, o PLN busca fornecer a capacidade de reconhecimento de padrões e contexto para computadores, extraindo informações, analisando sentimentos e interpretando possíveis sentidos, podendo até apresender com os textos fornecidos (ALLES, 2018).

2.9 *OpenNLP*

O Apache OpenNLP é um software de código aberto que visa disponibilizar recursos para a construção de uma pipeline para PLN. A principal característica da ferramenta OpenNLP (ALLES, 2018) é a disponibilização de diferentes recursos, tais como tokenização, segmentação de sentença, etiquetagem morfossintática, extração de entidade nomeada, extração de sintagmas, análise sintática e resolução de correferência, sendo que esta última ainda não se encontra muito bem desenvolvida na ferramenta.

A ferramenta OpenNLP contém um conjunto de componentes que podem ser utilizados na extração de informação para diferentes técnicas e para diferentes idiomas. Existem modelos disponíveis, pré-treinados, para alguns idiomas que ajudam na extração de informação para diferentes tipos de técnicas, tais como alemão, dinamarquês, holandês, inglês, sueco e português europeu. Além disso, a ferramenta OpenNLP também permite o treinamento de modelos para outros idiomas, por meio de uma coleção de dados que

consiste em um conjunto de textos anotados com tags. A interface de programação fornecida pela OpenNLP é composta por um conjunto de classes e/ou bibliotecas que podem ser utilizadas como linha de comando ou podem ser integradas utilizando-se chamadas das bibliotecas no desenvolvimento de uma aplicação que queira utilizar PLN (ALLES, 2018).

2.10 *QuarkClinic*

QuarkClinic é um sistema proprietário da empresa Quark Tecnologia que se propõe a servir como o ponto central de gestão de clínicas privadas, indo desde o setor financeiro, estoque de produtos, atendimento ao paciente, agendamento de consultas, etc. Seu foco são os funcionários das clínicas, não podendo ser acessado por pacientes. O portal é disponibilizado no seguinte link: <https://ng.quarkclinic.com.br/>.

2.10.1 Agendamento Online

O portal de Agendamento Online é um portal focado nos pacientes e gratuitamente disponibilizado para que o próprio paciente agende sua consulta, seguindo uma sequência de passos. O portal é disponibilizado no seguinte link: <https://agendamento.quarkclinic.com.br>.

3 METODOLOGIA

Foi proposto uma arquitetura de microsserviços, representada pela Figura 1, onde haverá um principal, escrito em java com uso do framework Spring boot que se comunicará com um backend do serviço de Agendamento Online. Para a interface de interação com a API principal será utilizado o javascript com o Vue.

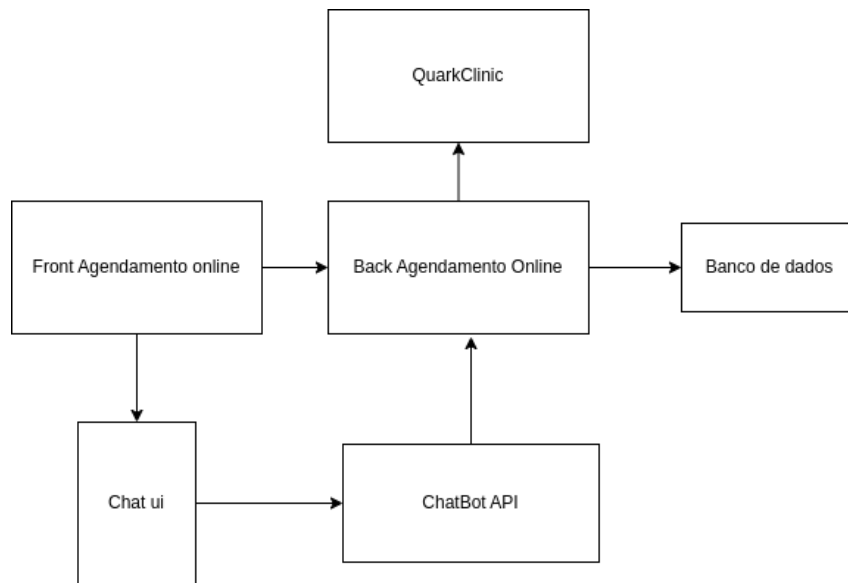


Figura 1 – Arquitetura do sistema proposto

Fonte: Elaborado pelo autor

3.1 Modelagem de dados

A aplicação contará com um relacionamento entre duas entidades, mostradas na Figura 2. A de mensagens para guardar todas as mensagens trocadas entre o chatbot e o usuário, que vai servir de ajuda para aperfeiçoar o modelo de treinamento com os dados. E há a entidade de sessão, que será responsável por guardar o fluxo em que a conversa está e guardar dados para as ações possíveis.

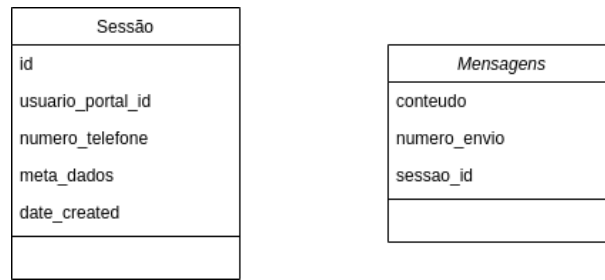


Figura 2 – Diagrama de classes

Fonte: Elaborado pelo autor

3.2 Estruturação da aplicação

A aplicação conta com 3 serviços principais: O módulo de agendamento Online, que já existia e opera com usuários e clientes reais, o backend que junta o fluxo de conversação com o processamento de linguagem, e a interface de comunicação para a conversação entre usuário e chatbot.

Sobre as duas últimas, que foram os principais focos deste trabalho, o microserviço de conversação conta com um endpoint principal, exposto em um RestController, em que permite ser passado, via parâmetros de URL, o texto fornecido pelo usuário que está interagindo com o chatbot, assim como um DTO (Data Transfer Object) que serve para transitar os dados necessários para realizar os principais fluxos da aplicação: Marcar uma consulta, cancelar uma consulta e listar o histórico de consultas do paciente. A disposição dos arquivos é mostrada na Figura 3.

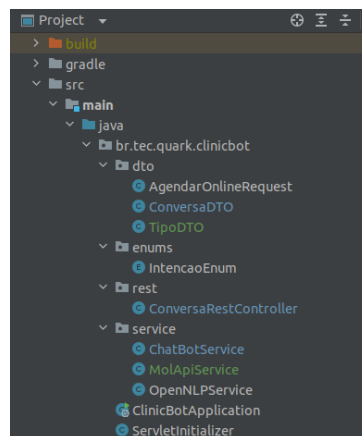


Figura 3 – Arquivos do microserviço de conversação

Fonte: Elaborado pelo autor

Ao acessar o RestController de conversação há a chamada para o método do Service do PLN, em que ocorrerá a categorização do texto fornecido e, dependendo de qual categoria se encaixar (marcar, cancelar ou listar consultas), será chamado o método específico de

cada uma. Estes métodos estão concentrados no Service do Chatbot (ChatBotService), e eles consistem em duas ações principais: garantir que o DTO para que a ação seja realizada esteja devidamente preenchido, e realizar a chamada para o microsserviço de Agendamento Online para que a ação seja de fato persistida no QuarkClinic.

O retorno deste endpoint de conversação é uma lista de DTOs que representam mensagens a serem mostradas na interface. Elas podem vir no formato de texto plano, no caso de uma mensagem comum, ou uma lista de json com chaves id e nome, para o caso em que seja necessário escolher entre entidades.

A comunicação com o microsserviço de Agendamento Online é feito via RestTemplate, que é uma classe utilitária para facilitar a comunicação com serviços RESTful, no service do mesmo (MolApiService). Ele é responsável tanto por listar entidades importantes para o preenchimento de informações para as ações, como lista de clínicas de uma organização, convênios, procedimentos e entre outros, quanto por realizar as ações do Service.

Neste microsserviço de Agendamento Online foi aberto um endpoint, para facilitar a comunicação com a API do chatbot, chamado de ChatBotRestController. Nele foi aberto endpoints para: buscar o histórico de consultas do usuário, cancelar consulta, salvar consulta e buscar horários de atendimento disponíveis para a agenda do profissional de saúde que o usuário escolheu. A disposição dos arquivos é mostrada na Figura 4.

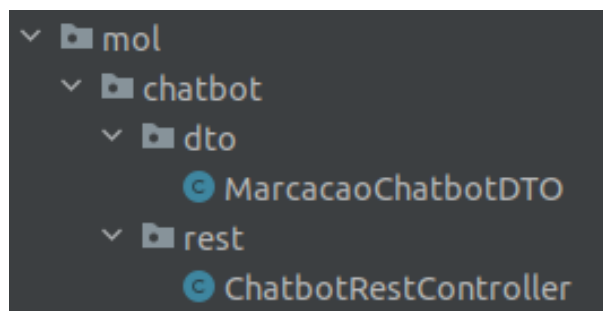


Figura 4 – Arquivos do microsserviço de agendamento

Fonte: Elaborado pelo autor

Há também outro RestController que expõe um endpoint responsável pelo processamento de linguagem natural em que ao fornecer um texto, como mostrada na Figura 5, é possível categorizá-lo em uma das entidades em que o modelo da PLN foi treinada. As entidades são: saudação, marcar consulta, cancelar consulta, finalizar consulta, histórico de consulta, afirmação, negação e finalização.

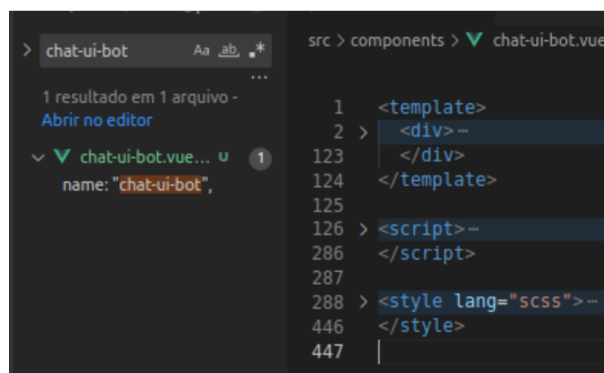
```
SAUDACAO oi. ola. olá. hello. bom dia. boa tarde. boa noite. oi, tudo bem. oi, bom dia. oi boa noite. oi, boa tarde. opa.
MARCAR_CONSULTA marcar. marcar consulta. marcar a consulta. posso marcar consulta. marcar uma consulta. realizar uma consulta. agendar.
CANCELAR_CONSULTA cancelar. cancelar uma consulta. cancelar o agendamento. cancelar agendamento.
FINALIZAR tchau. tchau, obrigado. até mais. até a próxima. finalizar. finaliza. acabe. acabar. bye. tchau, obrigado. até mais, valeu.
AFIRMACAO sim. si. s. yes. ye. y. concordo. afirmo. pode ser. ok. okay. tudo bem. tá certo. pode confirmar. pode fazer. pode sim.
NEGACAO não. n. no, not. discordo. não quero. não concordo. nego.
HISTORICO historico. historicos. historico de consultas. listar historico.
```

Figura 5 – Arquivo com dados de treinamento para o modelo

Fonte: Elaborado pelo autor

Cada categoria dessa foi preenchida com as principais frases que dão esse contexto, para que o modelo de PLN seja treinado com estes dados. Ao chamar o endpoint há um pré-processamento em que o texto fornecido pelo usuário passa antes da fase de categorização, que consiste em: fase de tokenização, *POS Tagging* e lematização.

A interface de interação com o usuário se localiza na interface do projeto do Agendamento Online. É apresentado na forma de um Chat em que as mensagens são divididas entre as digitadas pelo próprio usuário e as respostas do ChatBot. Seu desenvolvimento foi feito com o Framework Vue, fornecendo grande flexibilidade. Abaixo é apresentado o componente Vue criado, consiste de um único arquivo em que é concentrado tanto o HTML, CSS e Javascript, devido às características do próprio framework. A disposição dos arquivos é mostrada na Figura 6.



```
> chat-ui-bot Aa _ab_*
1 resultado em 1 arquivo -
Abrir no editor
v chat-ui-bot.vue... u 1
  name: "chat-ui-bot",
src > components > chat-ui-bot.vue
1 <template>
2 > <div>--
123 </div>
124 </template>
125
126 > <script>--
286 </script>
287
288 > <style lang="scss">--
446 </style>
447 |
```

Figura 6 – Arquivos e estrutura do FrontEnd Vue

Fonte: Elaborado pelo autor

3.3 Comunicação entre os microsserviços

Toda comunicação entre os microsserviços e também com o frontend é feito utilizando o protocolo HTTP, padrão utilizado largamente na internet. A interação do frontend com o microsserviço de conversação é feito através de um único endpoint utilizando biblioteca chamada Axios, e a comunicação entre backends foi feita com RestTemplate do próprio SpringFramework.

3.4 Rotas da API

As rotas HTTP desenvolvidas na api de conversação estão listadas na tabela 3. Há a rota padrão para conversação com o chatbot, a de detecção de intenção específica, fornecendo apenas um texto será possível categorizá-lo.

Tabela 3 – Rotas e verbos HTTP da API

Caminho	Verbo
/api/chat	POST
/api/nlp	GET

3.5 Fluxos de conversação

Ao interagir com o chatbot, o usuário seguirá um fluxo para que consiga atingir uma das ações em que o assistente suporta. O fluxo que representa a ação de marcar uma consulta está representado na Figura 7, o fluxo de listar o histórico de consultas aparece na Figura 8 e o fluxo de cancelamento na Figura 9.

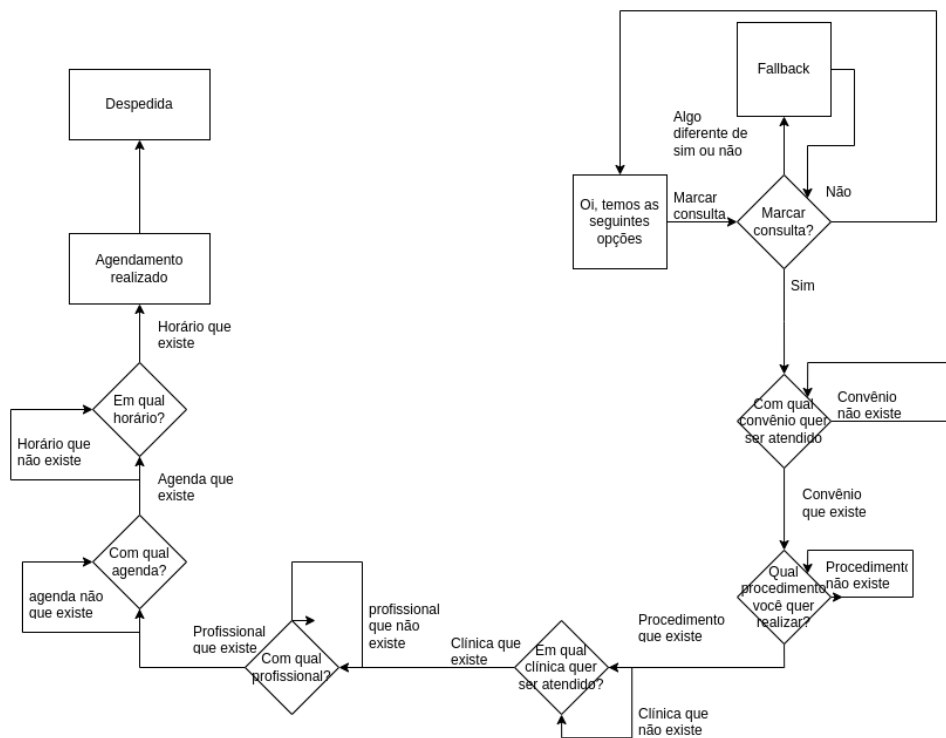


Figura 7 – Fluxograma de conversação para marcar consulta

Fonte: Elaborado pelo autor

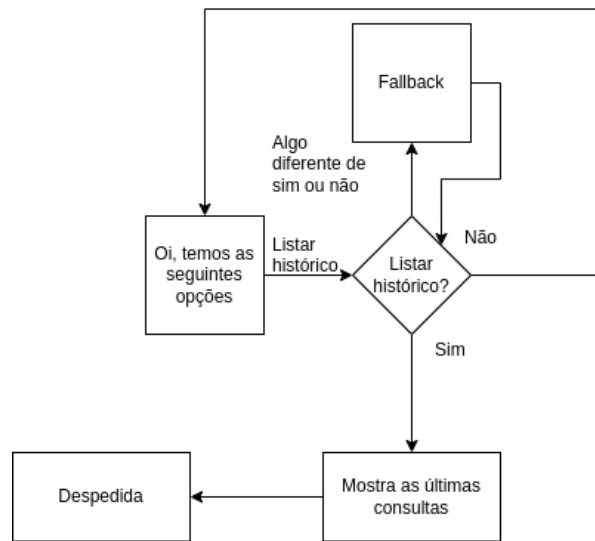


Figura 8 – Fluxograma de conversação para listar histórico de consulta

Fonte: Elaborado pelo autor

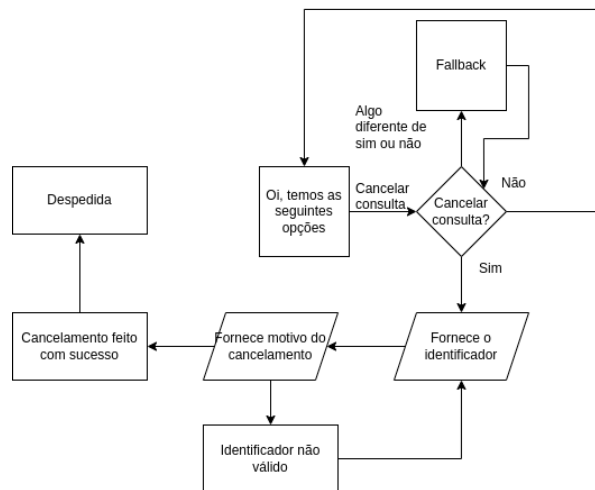


Figura 9 – Fluxograma de conversação para cancelar consulta

Fonte: Elaborado pelo autor

4 RESULTADOS

Há 3 fluxos principais que o assistente virtual se propõe. Marcação de consultas para uma determinada clínica escolhida pelo usuário, cancelamento de consultas fornecendo um identificador e a listagem das últimas consultas realizadas ou agendadas no sistema. Para isso, é necessário validar a categorização do modelo para o processamento de texto.

4.1 Categorização

A função principal e que norteia é chamada de “contextLoads”, mostrada na figura Figura 10. Ela é responsável por fazer um pré-processamento do texto e de treinar o modelo para categorização. O seu retorno é uma String informando em qual categoria, probabilisticamente, o texto fornecido se encaixou.

```
public String contextLoads(String text) throws IOException {
    // Tokenization
    String[] tokens = tokenize(text);

    // POS Tagging
    String[] posTags = posTag(tokens);

    // Lemmatization
    String[] lemmas = lemmatize(tokens, posTags);

    DoccatModel model = trainsCategorizerModel();

    //Categorizing/Chucking
    return categorize(model, lemmas);
}
```

Figura 10 – ‘Função contextLoads’

Fonte: Elaborado pelo autor

4.1.1 Tokenização

A função de tokenização, mostrada na figura Figura 11, é responsável por dividir a sentença em pequenas partes que são chamadas de tokens. Esses tokens podem ser, usualmente, palavras, números ou pontuação (vírgula, exclamação, interrogação, ponto final etc). O apacheNLP fornece uma forma de aplicar a tokenização com um modelo já treinado e disponível no seu site oficial. Foi utilizado um modelo para o português. O teste foi aplicado com a frase “oi, bom dia” e nota-se o resultado na Figura 12.

```
public static String[] tokenize(String text) throws IOException {  
  
    InputStream input = new FileInputStream( name: "/home/douglas/Área de Trabalho/tcc-clinic-bot/clinic-bot/pt-token.bin");  
  
    TokenizerModel model = new TokenizerModel(input);  
  
    TokenizerME tokenizer = new TokenizerME(model);  
  
    String[] tokens = tokenizer.tokenize(text);  
  
    return tokens;  
}
```

Figura 11 – Código da função de tokenização

Fonte: Elaborado pelo autor

```
> text = "oi, bom dia"  
> input  
> model  
> tokenizer  
> tokens = ["oi", ",", "bom", "dia"]
```

Figura 12 – Resultado da tokenização para a frase: oi, bom dia

Fonte: Elaborado pelo autor

4.1.2 POS tagger

A técnica de categorizar partes do texto (POS Tagger) busca marcar cada palavra (token) da sentença com seu tipo, como por exemplo: adjetivo, nome, número cardinal, etc. O código dessa função é mostrada na figura Figura 13.

```
public static String[] posTag(String[] tokens) throws IOException {  
  
    InputStream input = new FileInputStream( name: "/home/douglas/Área de Trabalho/tcc-clinic-bot/clinic-bot/pt-pos-maxent.bin");  
  
    POSModel model = new POSModel(input);  
  
    POSTaggerME tagger = new POSTaggerME(model);  
  
    String[] tags = tagger.tag(tokens);  
  
    return tags;  
}
```

Figura 13 – Código da função de POS Tagger

Fonte: Elaborado pelo autor

4.1.3 Lematização

Lematização é o processo de transformar ou mapear uma palavra que pode estar flexionada em algum tempo, modo, gênero ou outros tipo em uma forma básica da própria palavra, que também pode ser chamado de lema. Para que isso ocorra é necessário que a sentença passe pelo tokenizador e pelo processo de POS tagger. A função, mostrada na figura Figura 14, segue o mesmo princípio da tokenização, e foi utilizado um modelo disponível no site do OpenNLP. Não foram encontrados modelos em português para este caso no site oficial, e os disponíveis na internet não tiveram um desempenho satisfatório, portanto foi utilizado o inglês. O resultado é mostrado na figura Figura 15.

```

public static String[] lematize(String[] tokens, String[] posTags) throws IOException {
    InputStream input = new FileInputStream(new File("nome/douglas/Área de Trabalho/tcc-clinic-bot/clinic-bot/en-lemmatizer.bin"));
    LemmatizerModel model = new LemmatizerModel(input);
    LemmatizerME categorizer = new LemmatizerME(model);
    String[] lemmaTokens = categorizer.Lemmatize(tokens, posTags);
    return lemmaTokens;
}

```

Figura 14 – Código da função de lematização

Fonte: Elaborado pelo autor

```

> P tokens = ["oi", ",", "bom", "dia"]
> P posTags = ["num", "punc", "adj", "n"]
> ≡ input
> ≡ model
> ≡ categorizer
> 1 2 3 ≡ lemmaTokens = ["oi", ",", "bom", "dia"]

```

Figura 15 – Resultado da função de lematização

Fonte: Elaborado pelo autor

4.1.4 Treinando modelo

É preciso gerar um modelo que possa classificar a sentença em uma categoria específica. Para isso é necessário fornecer dados com vários exemplos e treinar o modelo. Quanto mais dados melhor. Na função de treinamento, mostrada na figura Figura 16, foi fornecido um arquivo de texto com as categorias e os exemplos relativos.

```

public static DoccatModel trainsCategorizerModel() throws IOException {
    InputStreamFactory inputStreamFactory = new MarkableFileInputStreamFactory(new File("path/name/categorias.txt"));
    ObjectStream<String> lineStream = new PlainTextByLineStream(inputStreamFactory, StandardCharsets.UTF_8);
    ObjectStream<DocumentSample> sampleStream = new DocumentSampleStream(lineStream);

    DoccatFactory factory = new DoccatFactory(new FeatureGenerator[] { new BagOfWordsFeatureGenerator() });

    TrainingParameters params = ModelUtil.createDefaultTrainingParameters();
    params.put(TrainingParameters.CUTOFF_PARAM, 1);

    DoccatModel model = DocumentCategorizerME.train("languageCode": "pt", sampleStream, params, factory);

    return model;
}

```

Figura 16 – Código para treinamento do modelo

Fonte: Elaborado pelo autor

4.1.5 Categorizando sentença

Na função de categorizar a sentença, mostrada na figura Figura 17, já com o modelo preparado basta fornecer os tokens da expressão. Ao categorizar será retornado um lista de

números que representam probabilidades para cada categoria, mostrada na figura Figura 18, e a função “getBestCategory” será responsável por recuperar a mais provável. É importante ressaltar que, caso o modelo não tenha encontrado uma categoria que se sobressaia, as probabilidades serão as mesmas, portanto seria retornado o primeiro encontrado.

```
public static String categorize(DoccatModel model, String[] tokens) {  
  
    DocumentCategorizerME myCategorizer = new DocumentCategorizerME(model);  
  
    double[] probabilitiesOfOutcomes = myCategorizer.categorize(tokens);  
  
    final var isAllEquals = Arrays.stream(probabilitiesOfOutcomes).distinct().count() == 1;  
  
    if (isAllEquals) {  
        return "NAO_IDENTIFICADO";  
    }  
  
    String category = myCategorizer.getBestCategory(probabilitiesOfOutcomes);  
    System.out.println("Category: " + category);  
  
    return category;  
}
```

Figura 17 – Código da função de categorização

Fonte: Elaborado pelo autor

```
> P model  
> P tokens = ["oi", ",", "bom", "dia"]  
> myCategorizer  
v probabilitiesOfOutcomes = [0.228  
  01 0 = 0.22840064900717222  
  01 1 = 0.12859989183213802  
  01 2 = 0.12859989183213802  
  01 3 = 0.12859989183213802  
  01 4 = 0.12859989183213802  
  01 5 = 0.12859989183213802  
  01 6 = 0.12859989183213802  
  01 isAllEquals = false  
> category = "SAUDACAO"
```

Figura 18 – Resultado com as probabilidades da função de categorizar

Fonte: Elaborado pelo autor

4.2 Interface do Chat

A interface de comunicação com o paciente, mostrada na figura Figura 19, tem uma área para a entrada do texto do usuário, uma interface para visualização das mensagens e um tipo de cabeçalho com opção de minimizar e maximizar o bloco de mensagens. Ao

enviar mensagens é chamado o endpoint para o processamento do texto e categorização. A medida que a conversação avança o usuário poderá escolher qual ação deseja para o momento e, para cada escolha, o backend fornecerá mensagens para serem somente lidas ou respondias pelo usuário. O objetivo é preencher o DTO que fica do lado do cliente com as informações necessárias para a realização da ação requerida.



Figura 19 – Interface inicial do Portal de Agendamento Online com o ChatBot

Fonte: Elaborado pelo autor

4.3 Fluxos de conversação via Chat

A conversação do usuário com o chatbot pode ser direcionada para vários caminhos. Os categorizados principais são: saudação, marcação de consulta, cancelamento de consulta, histórico de consultas e *fallback* para mensagens em que não foi possível uma identificação.

4.3.1 Saudação

O fluxo de saudação pode ser acionado com frases como: "olá, bom dia" e não carrega uma intenção que dispara uma das ações principais, mas sim serve para mostrar ao usuário quais opções são possíveis de escolher. A interação por interface é mostrada na figura Figura 20 e o resultado gerado pela requisição na figura Figura 21.

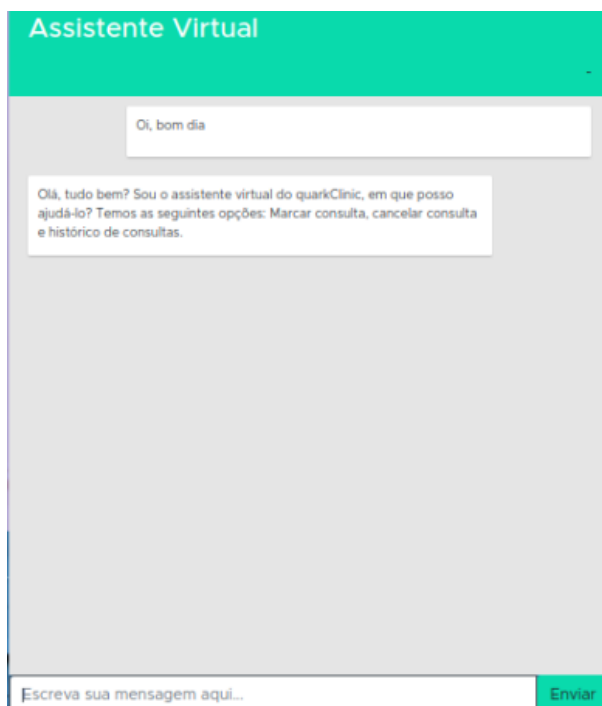


Figura 20 – Fluxo de saudação

Fonte: Elaborado pelo autor

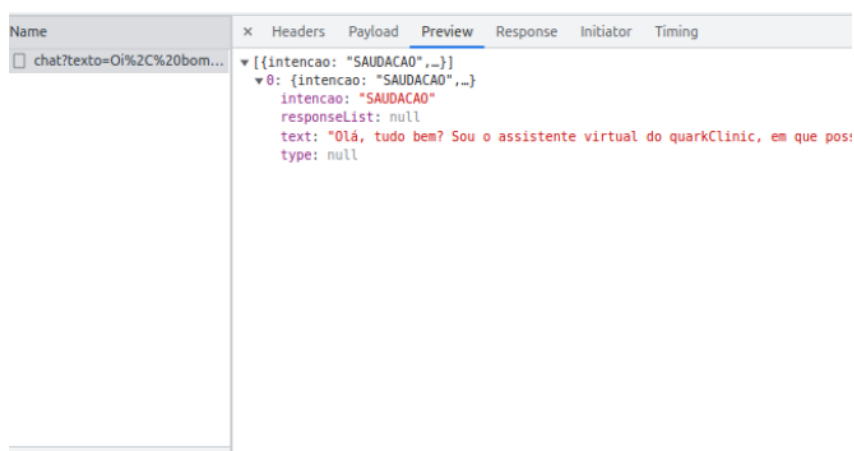


Figura 21 – Resultado da requisição do fluxo de saudação

Fonte: Elaborado pelo autor

4.3.2 Marcar consulta

O fluxo de marcação de consulta tem o objetivo de agendar, para o paciente, um horário específico em determinado dia na agenda de um profissional de saúde em determinada clínica. Para isso, será necessário escolher dentre algumas opções que o Agendamento Online requer, tais como: Qual convênio será escolhido pelo paciente, para qual procedimento será feita a consulta, em qual clínica, com qual profissional e qual

agenda e, finalmente, em qual horário será feito o atendimento. Com as informações confirmadas, o agendamento será marcado no sistema.

- A interação por interface para escolher o convênio é mostrada na figura Figura 22 e o resultado gerado pela requisição na figura Figura 23:

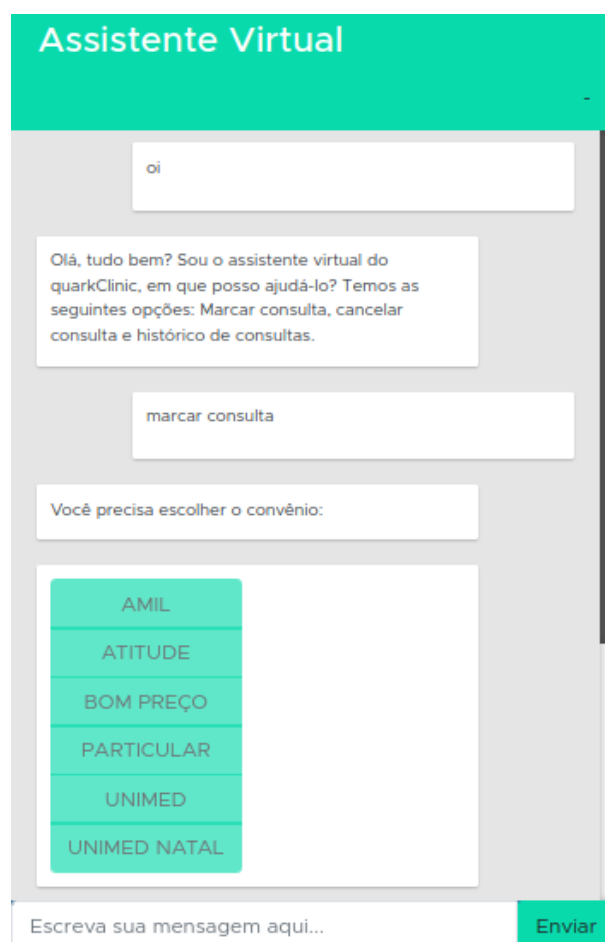
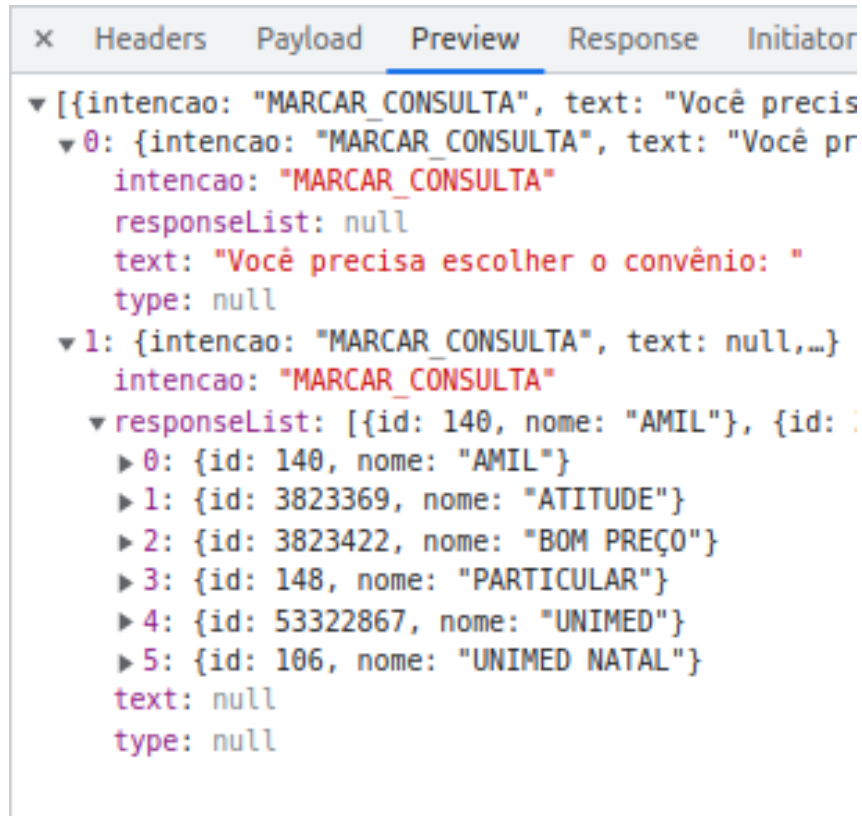


Figura 22 – Escolha de convênio no fluxo de marcação de consulta

Fonte: Elaborado pelo autor



```
× Headers Payload Preview Response Initiator
▼ [{intencao: "MARCAR_CONSULTA", text: "Você precis
  ▼ 0: {intencao: "MARCAR_CONSULTA", text: "Você pr
    intencao: "MARCAR_CONSULTA"
    responseList: null
    text: "Você precisa escolher o convênio: "
    type: null
  ▼ 1: {intencao: "MARCAR_CONSULTA", text: null,...}
    intencao: "MARCAR_CONSULTA"
    ▼ responseList: [{id: 140, nome: "AMIL"}, {id: :
      ▶ 0: {id: 140, nome: "AMIL"}
      ▶ 1: {id: 3823369, nome: "ATITUDE"}
      ▶ 2: {id: 3823422, nome: "BOM PREÇO"}
      ▶ 3: {id: 148, nome: "PARTICULAR"}
      ▶ 4: {id: 53322867, nome: "UNIMED"}
      ▶ 5: {id: 106, nome: "UNIMED NATAL"}
      text: null
      type: null
```

Figura 23 – Resultado da requisição da escolha do convênio

Fonte: Elaborado pelo autor

- A interação por interface para escolher o procedimento é mostrada na figura Figura 24 e o resultado gerado pela requisição na figura Figura 25:



Figura 24 – Escolha de procedimento no fluxo de marcação de consulta

Fonte: Elaborado pelo autor

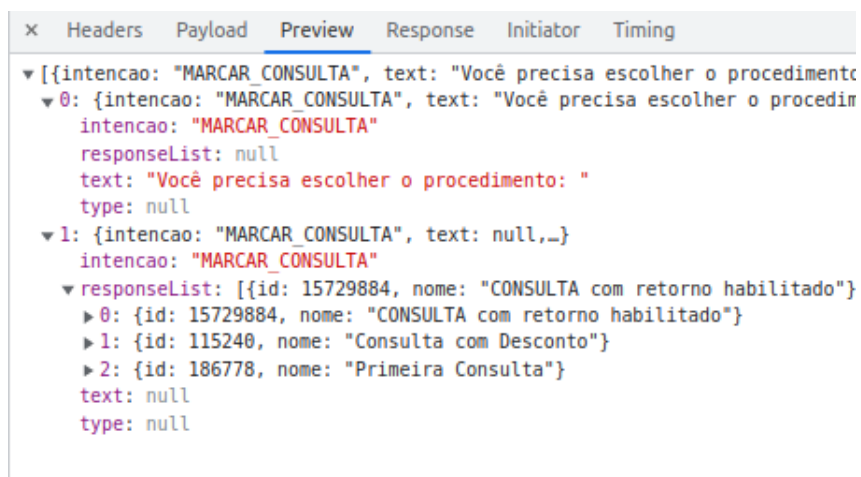


Figura 25 – Resultado da requisição da escolha do procedimento

Fonte: Elaborado pelo autor

- A interação por interface para escolher a clínica é mostrada na figura Figura 26 e o resultado gerado pela requisição na figura Figura 27:

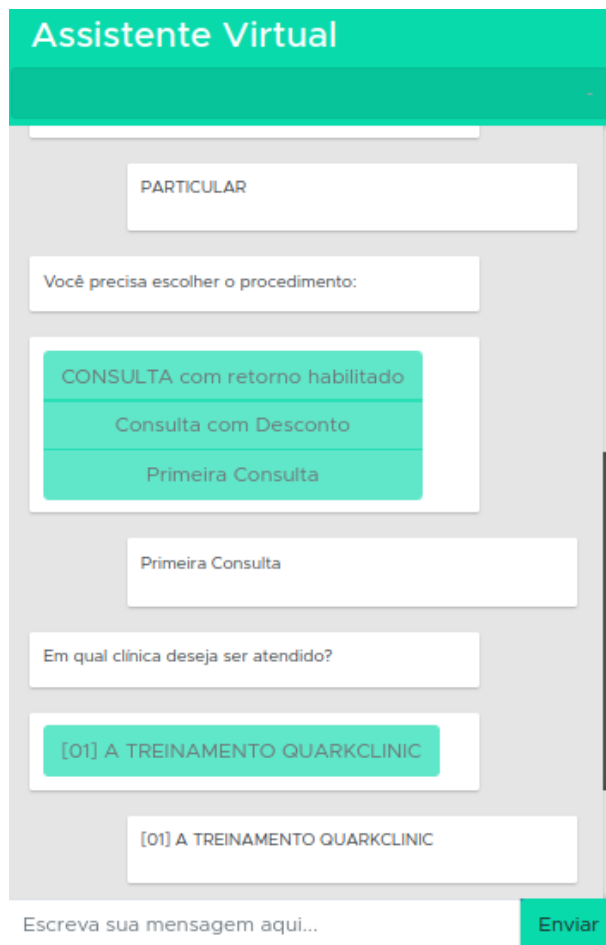


Figura 26 – Escolha de clínica no fluxo de marcação de consulta

Fonte: Elaborado pelo autor

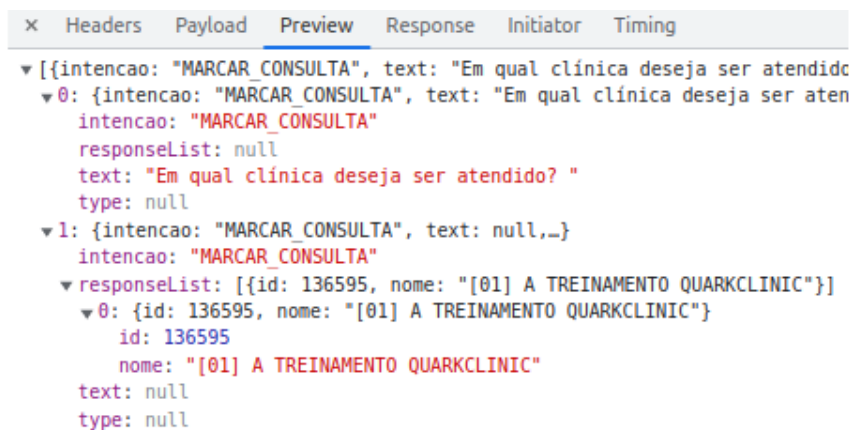


Figura 27 – Resultado da requisição da escolha da clínica

Fonte: Elaborado pelo autor

- A interação por interface para escolher o profissional é mostrada na figura Figura 28 e o resultado gerado pela requisição na figura Figura 29:

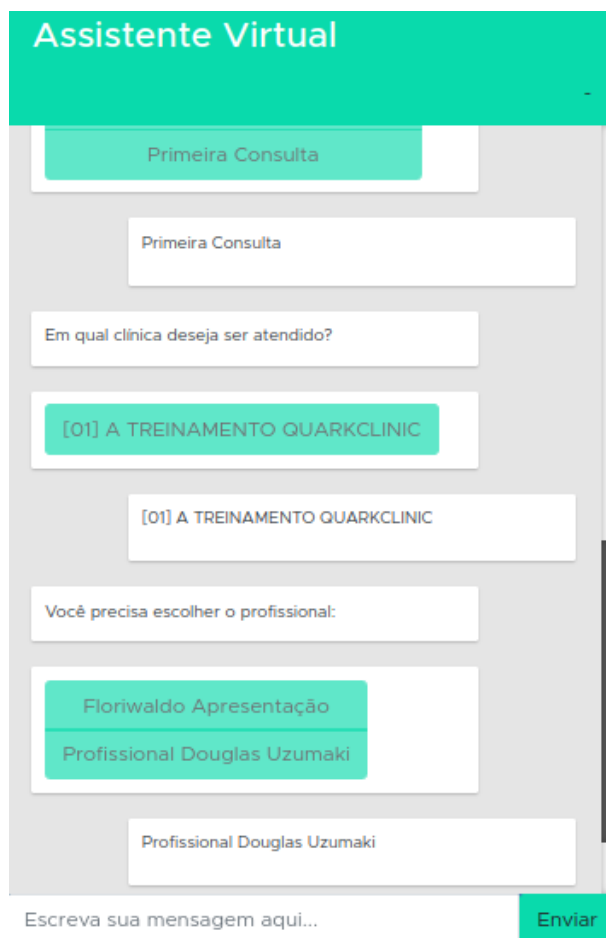


Figura 28 – Escolha do profissional no fluxo de marcação de consulta

Fonte: Elaborado pelo autor

```
x Headers Payload Preview Response Initiator Timing
▼ [{"intencao": "MARCAR_CONSULTA", text: "Você precisa escolher o
  ▼ 0: {"intencao": "MARCAR_CONSULTA", text: "Você precisa escolhe
    intencao": "MARCAR_CONSULTA"
    responseList": null
    text: "Você precisa escolher o profissional: "
    type": null
  ▼ 1: {"intencao": "MARCAR_CONSULTA", text: null,...}
    intencao": "MARCAR_CONSULTA"
    ▼ responseList": [{"id: 112682, nome: "Floriwaldo Apresentação"}
      ▼ 0: {"id: 112682, nome: "Floriwaldo Apresentação"}
        id: 112682
        nome: "Floriwaldo Apresentação"
      ▼ 1: {"id: 16062466, nome: "Profissional Douglas Uzumaki"}
        id: 16062466
        nome: "Profissional Douglas Uzumaki"
        text: null
        type": null
```

Figura 29 – Resultado da requisição da escolha do profissional

Fonte: Elaborado pelo autor

- A interação por interface para escolher a agenda do profissional é mostrada na figura Figura 30 e o resultado gerado pela requisição na figura Figura 31:



Figura 30 – Escolha da agenda no fluxo de marcação de consulta

Fonte: Elaborado pelo autor

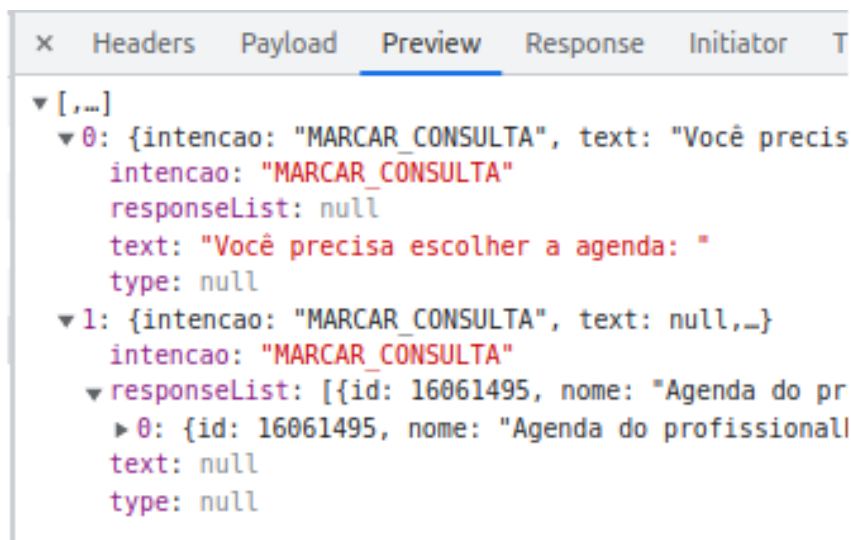


Figura 31 – Resultado da requisição da escolha da agenda

Fonte: Elaborado pelo autor

- A interação por interface para escolher o horário é mostrada na figura Figura 32 e o resultado gerado pela requisição na figura Figura 33:

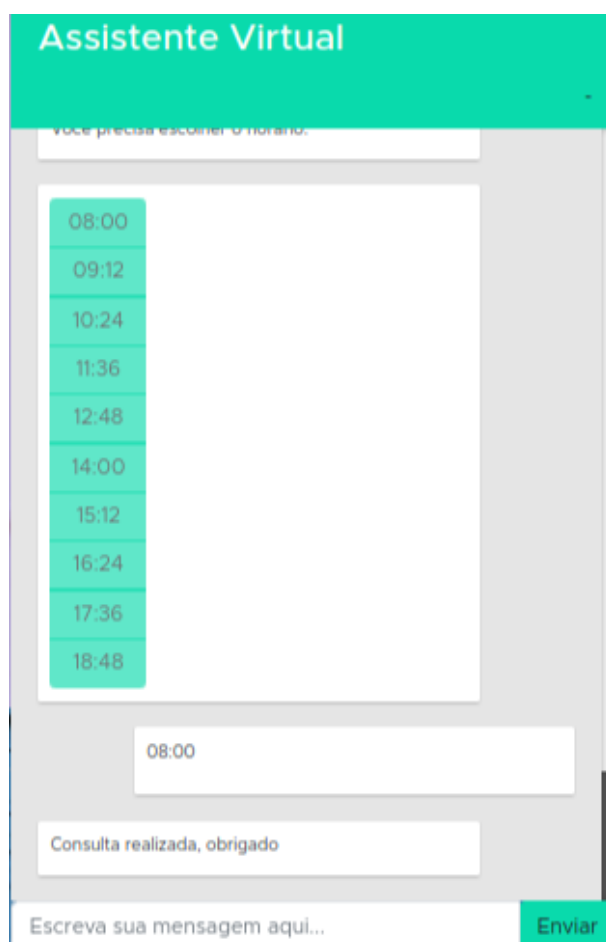
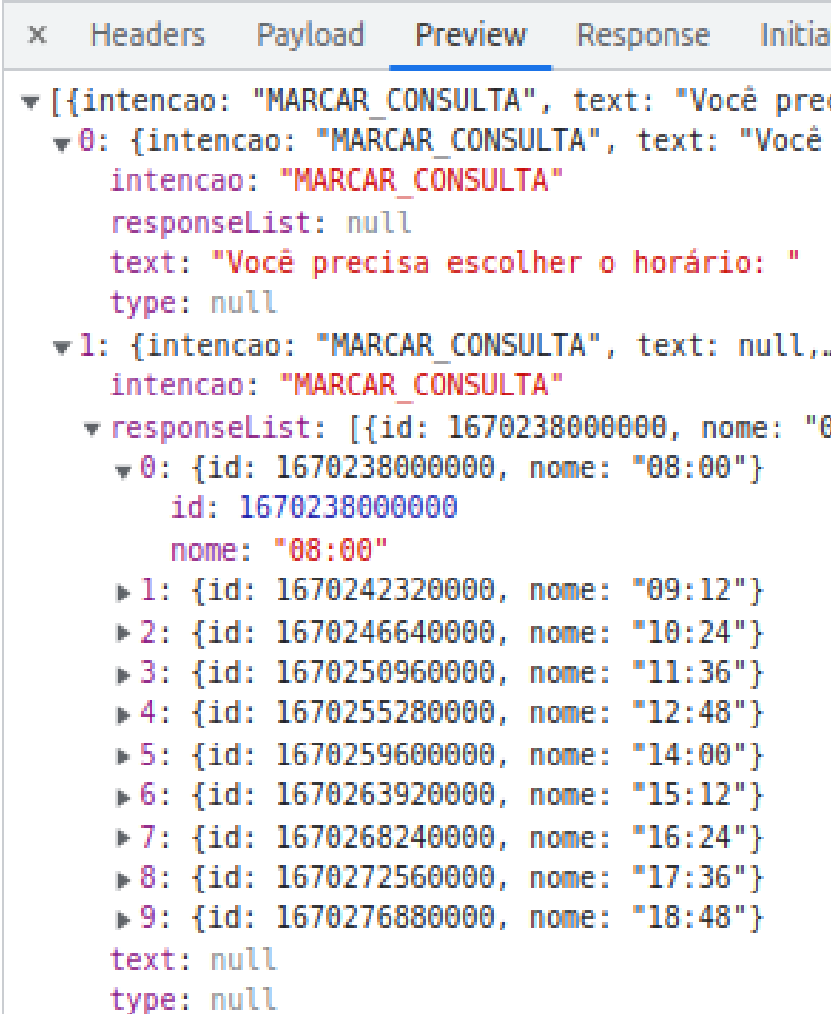


Figura 32 – Escolha do horário no fluxo de marcação de consulta

Fonte: Elaborado pelo autor



```
× Headers Payload Preview Response Initia
▼ [{intencao: "MARCAR_CONSULTA", text: "Você precisa
  ▼ 0: {intencao: "MARCAR_CONSULTA", text: "Você
      intencao: "MARCAR_CONSULTA"
      responseList: null
      text: "Você precisa escolher o horário: "
      type: null
  ▼ 1: {intencao: "MARCAR_CONSULTA", text: null,
      intencao: "MARCAR_CONSULTA"
      ▼ responseList: [{id: 1670238000000, nome: "08:00"}
          ▼ 0: {id: 1670238000000, nome: "08:00"}
              id: 1670238000000
              nome: "08:00"
          ▶ 1: {id: 1670242320000, nome: "09:12"}
          ▶ 2: {id: 1670246640000, nome: "10:24"}
          ▶ 3: {id: 1670250960000, nome: "11:36"}
          ▶ 4: {id: 1670255280000, nome: "12:48"}
          ▶ 5: {id: 1670259600000, nome: "14:00"}
          ▶ 6: {id: 1670263920000, nome: "15:12"}
          ▶ 7: {id: 1670268240000, nome: "16:24"}
          ▶ 8: {id: 1670272560000, nome: "17:36"}
          ▶ 9: {id: 1670276880000, nome: "18:48"}
          text: null
          type: null
```

Figura 33 – Resultado da requisição da escolha do horário

Fonte: Elaborado pelo autor

- O resultado da requisição para a confirmação da consulta é mostrada na figura Figura 34 e o resultado gerado no sistema quarkclinic é mostrado na figura Figura 35:

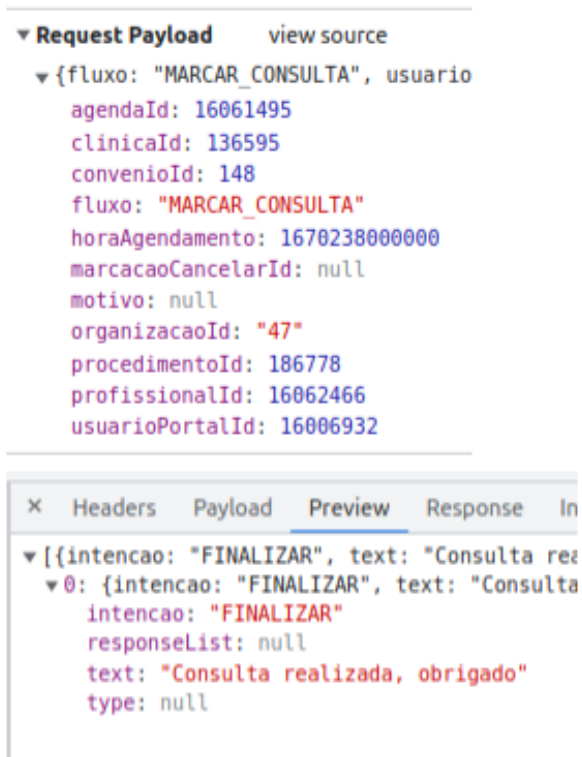


Figura 34 – Resultado da requisição da confirmação da consulta

Fonte: Elaborado pelo autor

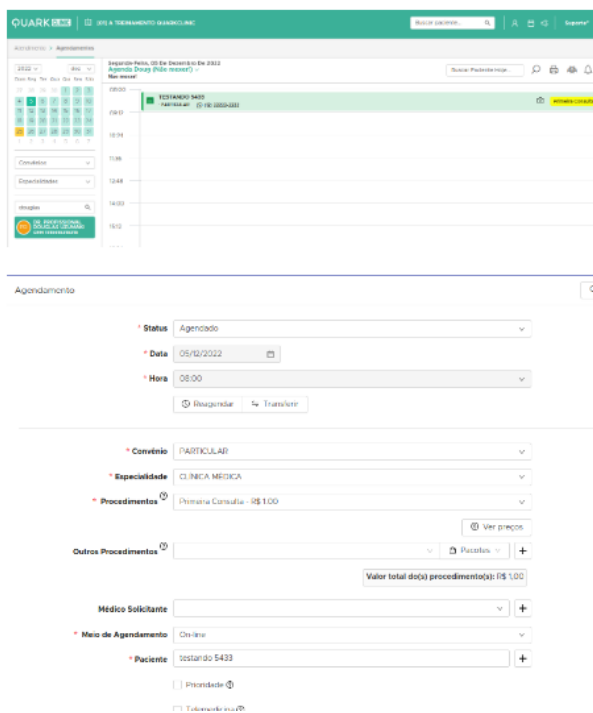


Figura 35 – Agendamento criado no sistema QuarkClinic

Fonte: Elaborado pelo autor

4.3.3 Histórico de consultas

O fluxo de listar o histórico de consultas é simples, basta pedir para o chatbot listar as últimas consultas. Como o usuário está logado basta apenas consultar seu identificador e retornar os últimos agendamentos, independente de já ter sido realizado ou ainda agendado para uma data futura ou até cancelados. A interação com a interface é mostrada na figura Figura 36.

O retorno conta como um identificador, que será necessário no fluxo de cancelamento caso o usuário queira cancelar o agendamento, e outras informações como: data do agendamento, clínica em que foi agendado, status em que o agendamento se encontra e o convênio.

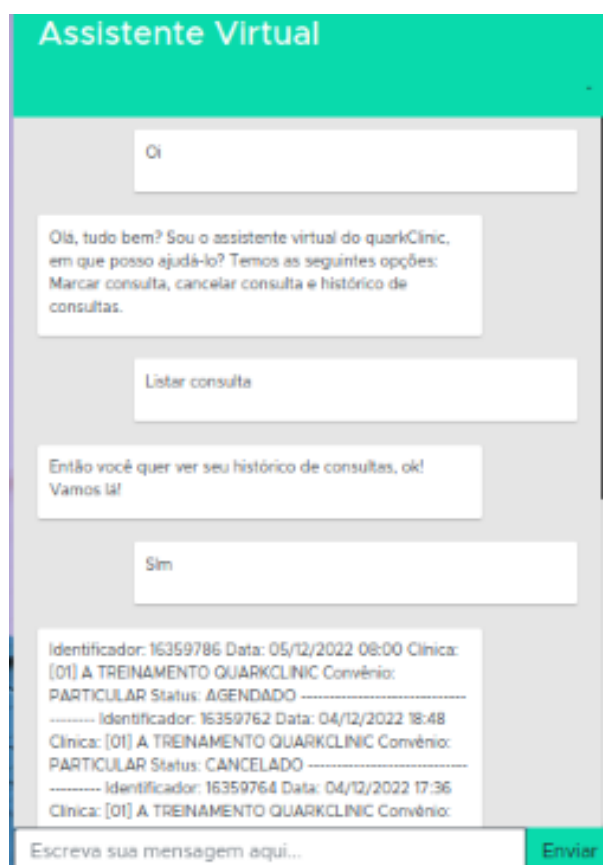


Figura 36 – Fluxo para histórico de consultas

Fonte: Elaborado pelo autor

4.3.4 Cancelar consulta

Ao cancelar uma consulta o horário reservado pelo paciente será novamente disponibilizado, para isso será necessário informar o identificador da marcação, que pode ser obtido no histórico, e um motivo para o cancelamento, informação essa que será exibida no sistema do Quarkclinic para análise da clínica. A interação com a interface é mostrada

na figura Figura 37, assim como o resultado da requisição. O resultado do cancelamento no sistema QuarkClinic é mostrada na figura Figura 38.

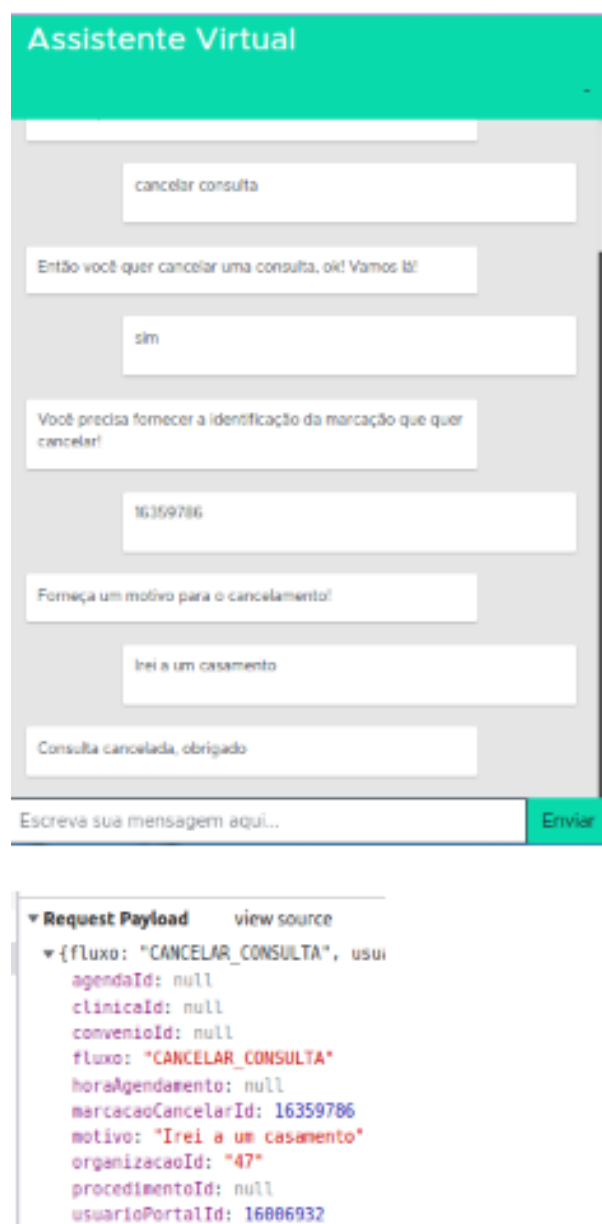


Figura 37 – Fluxo para cancelamento de consultas

Fonte: Elaborado pelo autor

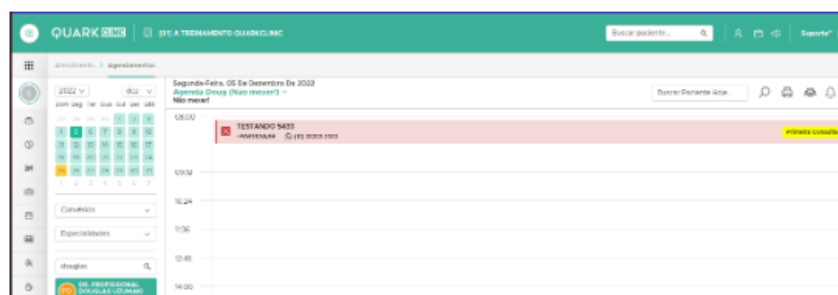


Figura 38 – Agendamento cancelado no sistema QuarkClinic

Fonte: Elaborado pelo autor

4.3.5 Fallback

O *fallback*, termo usado para representar um plano de contingência para caso não ocorra como esperado, é uma mensagem que é retornada quando o chatbot não consegue categorizar corretamente o que o usuário enviou para processamento. Neste momento o chatbot informa que não entendeu e sugere que o paciente tente reescrever a frase de uma forma mais clara. A interação com a interface é mostrada na figura Figura 39.



Figura 39 – Fluxo de mensagem não identificada

Fonte: Elaborado pelo autor

4.4 Endpoint para PLN

O endpoint para PLN é uma forma de acessar a função "contextLoad" de uma forma direta, servindo apenas como um endpoint de análise de texto e categorização, não participando do fluxo de conversação. Este tipo de endpoint pode ser útil para outras situações, como um dos casos do quarkclinic que é o de confirmação de consulta via Whatsapp, em que o paciente é perguntado sobre a presença ou não em uma consulta já marcada. Nestes casos o endpoint pode ser utilizado para categorizar sua resposta como positiva ou negativa para posteriormente decidir que fluxo seguir.

5 CONCLUSÃO

Este trabalho desenvolveu uma interface de comunicação inteligente para o público alvo das clínicas de saúde, os pacientes, capaz de atender aos fluxos envolvendo agendamento de consultas, que é a mais importante fonte de receita dessas instituições.

Para alcançar este objetivo foi realizado um estudo de ferramentas envolvendo levantamento bibliográfico para a construção dos microsserviços e do modelo para processamento de texto em linguagem natural, realizando a extração das intenções e processando sua requisição.

Utilizando-se de ferramentas como: apacheNLP, Vue, Spring-boot, postgresql, foi possível desenvolver uma solução simples e eficaz capaz de cumprir o objetivo proposto.

Este trabalho necessitou da aquisição de conhecimento de várias áreas, tais como: Linguagens de programação, Arquitetura e engenharia de software, redes de computadores, modelagem de banco de dados, aprendizado de máquina com processamento de linguagem natural, etc. Apesar da deficiência deste autor na área de inteligência artificial no início, com revisão e análise foi possível finalizar a proposta.

5.1 Trabalhos futuros

Este trabalho é um primeiro passo para o desenvolvimento de uma Api mais completa no campo de processamento de linguagem natural para a área de atuação do QuarkClinic. A expectativa seria abrir para interfaces de mensageria mais utilizadas por usuários, como: Whatsapp, Messenger e Telegram.

REFERÊNCIAS

ALLES, V. J. *Construção de um corpus para extrair entidades nomeadas do Diário Oficial da União utilizando aprendizado supervisionado*. 73 p. Dissertação (Mestrado em Engenharia Elétrica) — Universidade de Brasília, Brasília, 2018.

BARRO, B. B. O que são frameworks e quais os mais utilizados. 2022. Disponível em: <[https://www.hostinger.com.br/tutoriais-frameworks](https://www.hostinger.com.br/tutoriais/frameworks)><https://www.hostinger.com.br/tutoriais/frameworks>. Acesso em: 15 nov. 2022.

BOAGLIO, F. *Spring Boot: Acelere o desenvolvimento de microsserviços*. São Paulo, Brasil: Casa do Código, 2017.

CARVALHO, V. *PostgreSQL: Banco de dados para aplicações web modernas*. Brasil: Casa do Código, 2017.

CHACON, S. Pro-git. 2014. Disponível em: <[https://git-scm.com/book/pt-br-v2](https://git-scm.com/book/pt-br/v2)><https://git-scm.com/book/pt-br/v2>. Acesso em: 10 dez. 2022.

CRUZ L.; ALENCAR, A. *Assistentes Virtuais Inteligentes e Chatbots: Um guia prático e teórico sobre como criar experiências e recordações encantadoras para os clientes da sua empresa*. Rio de Janeiro, Brasil: BRASPORT Livros e Multimídia Ltda, 2018.

DEITEL P.; DEITEL, H. *Java: Como programar*. Brasil: Pearson Education do Brasil Ltda, 2017.

DUARTE, L. O que é um micro serviço ou microservice? 2018. Disponível em: <<https://www.luiztools.com.br/post/o-que-e-um-micro-servico-ou-microservice/>><https://www.luiztools.com.br/post/o-que-e-um-micro-servico-ou-microservice/>. Acesso em: 10 dez. 2022.

FLANAGAN, D. *JavaScript: o guia definitivo*. Porto Alegre, Brasil: Bookman Companhia Editora Ltda, 2013.

HAT, R. Api rest. 2020. Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>><https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>. Acesso em: 10 dez. 2022.

HAT, R. O que é uma api? 2022. Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>><https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>. Acesso em: 10 dez. 2022.

MONGODB. Api rest. 2022. Disponível em: <<https://www.mongodb.com/pt-br/what-is-mongodb>><https://www.mongodb.com/pt-br/what-is-mongodb>. Acesso em: 22 dez. 2022.

SAUDATE, A. *REST: Construa api's inteligentes de maneira simples*. Brasil: [s.n.], 2013.

SOUTO, M. Front-end, back-end e full stack. 2022. Disponível em: <<https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>><https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>. Acesso em: 04 dez. 2022.

TENENBAUM, A. H. B. *Sistemas Operacionais Modernos*. Brasil: Pearson Education do Brasil Ltda, 2015.

VALENTE, M. T. *Engenharia de Software Moderna: Princípios e práticas para desenvolvimento de software com produtividade*. Belo Horizonte, Brasil: [s.n.], 2020.

VUE. O que é vue.js? 2022. Disponível em: <<https://br.vuejs.org/v2/guide/index.html#O-que-e-Vue-js>><https://br.vuejs.org/v2/guide/index.html#O-que-e-Vue-js>. Acesso em: 02 dez. 2022.