INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PEDRO II
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MARCOS PACHECO UCHÔA

OPEN SCHEDULE: UMA PLATAFORMA *OPEN SOURCE* PARA REALIZAÇÃO DE AGENDAMENTOS DE CONSULTAS VIA *CHATBOT*

PEDRO II

2023

MARCOS PACHECO UCHÔA

OPEN SCHEDULE: UMA PLATAFORMA *OPEN SOURCE* PARA REALIZAÇÃO DE AGENDAMENTOS DE CONSULTAS VIA *CHATBOT*

Trabalho de Conclusão de Curso (artigo científico) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Pedro II.

Orientador: Prof. Me. Willame Oliveira Pereira.
Coorientadora: Dra. Ana Maria Carreiro de Melo Martins.

PEDRO II

2023

OPEN SCHEDULE: UMA PLATAFORMA *OPEN SOURCE* PARA REALIZAÇÃO DE AGENDAMENTOS DE CONSULTAS VIA *CHATBOT*

Marcos Pacheco Uchôa[1]
Willame Oliveira Pereira[2]

RESUMO

Os aplicativos de mensagens instantâneas são o ambiente propício para os *chatbots*, programas de computador que simulam uma conversa com um humano. Um dos domínios onde *chatbots* são utilizados é o da saúde, mais especificamente no atendimento ao paciente para a realização de agendamentos de consultas. Geralmente, o uso dessa tecnologia está quase limitado às grandes instituições de saúde privada encontradas principalmente em capitais. Nesse contexto, este artigo cria uma plataforma gratuita para melhoria do atendimento aos usuários nas instituições de saúde públicas e também nos estabelecimentos de saúde privados que não usam essas tecnologias devido aos custos das opções encontradas no mercado. A plataforma foi totalmente desenvolvida com tecnologias *open source*, que tornaram a implementação mais fácil e rápida. Foram criadas aplicações para os profissionais e pacientes das instituições de saúde. O *chatbot* foi criado com objetivo de fornecer funcionalidades, como agendar consultas, ver consultas marcadas e cancelar consultas de forma fácil e intuitiva para os usuários.

Palavras-chave: Chatbot. Código Aberto. Atendimento. Saúde.

ABSTRACT

Instant messaging applications are the ideal environment for chatbots, computer programs that simulate a conversation with a human. One of the domains where chatbots are used is health, more specifically in patient care for scheduling appointments. Generally, the use of this technology is almost limited to large private health institutions found mainly in capital cities. In this context, this article creates a free platform to improve service to users in public health institutions and also in private health establishments that do not use these technologies due to the cost of the options found on the market. The platform was fully developed with open source technologies, which made the implementation easier and faster. Applications were created for professionals and patients in health institutions. The chatbot was created with the aim of providing functionalities such as scheduling appointments, seeing scheduled appointments and canceling appointments in an easy and intuitive way for users.

Keywords: Chatbot. Open Source. Attendance. Health.

[1] Graduação em Análise e Desenvolvimento de Sistemas. IFPI. marcospacheco10111@gmail.com
[2] Mestrado em Ciência da Computação. UFPE. willame.oliveira@ifpi.edu.br

1 INTRODUÇÃO

Os aplicativos de mensagens instantâneas, como WhatsApp, Telegram e Line, se popularizaram junto aos *smartphones* como uma alternativa ao SMS, um meio de baixo custo para troca de mensagens individualmente ou em grupo em tempo real(CHURCH e DE OLIVEIRA, 2013). Nos blogs dos próprios aplicativos é mostrado o grande sucesso de tais serviços. A exemplo o WhatsApp notícia os dois bilhões de usuários alcançados em fevereiro de 2020 (WhatsApp, 2020) e o Telegram com seus 700 milhões de usuários em junho de 2022 (Telegram, 2022).

Essas plataformas que hoje facilitam como as pessoas se comunicam, são um ambiente quase perfeito para a inserção de *chatbots* (DALE, 2016). *Chatbots* são programas de computador que utilizam processamento de linguagem natural (PLN) para interagir com o usuário como se fosse um humano por meio de texto ou voz (SMUTNY e SCHREIBEROVA, 2020). A pesquisa de Shawar e Atwell (2017) apresenta implementações bem sucedidas de *chatbots* nos domínios da educação, recuperação de informações, negócios, *e-commerce* e entretenimento.

Não obstante, o desenvolvimento de *chatbots* e assistentes virtuais no âmbito da saúde têm obtido sucesso. Algumas dessas soluções são voltadas para auxiliar nos cuidados a pacientes com diabetes como em BALSA et al. (2019) e BUINHAS et al. (2019). Outras ajudam no tratamento de doenças mentais (ABD-ALRAZAQ et al., 2019), acompanhamento de gestantes no pré-natal (CARVALHO et al., 2019) e na obtenção de informações e realização de atendimentos (DJOELIANTO et al., 2022; SILVA e CAMPOS, 2021). Também é destacável o uso dessa tecnologia durante a pandemia de COVID-19 (AMIRI e KARAHANNA, 2022; ALMALKI e AZEEZ, 2020).

A pesquisa de Palanican et al. (2019), mostra uma alta aceitabilidade dos médicos em relação ao uso de *chatbots* para a realização de agendamentos e também para obtenção de informações como a localização dos estabelecimentos de saúde. O uso dessa ferramenta para o agendamento de consultas, pode trazer vários benefícios, principalmente em um país como o Brasil onde a locomoção é algo complicado e nem todos moram perto de algum estabelecimento de saúde. Um desses benefícios é a diminuição das filas na procura de serviços de saúde, além de uma contribuição positiva para o alto índice de absenteísmo (pacientes que não comparecem às consultas marcadas) no Brasil (DA SILVEIRA et al., 2018). Além disso, pode ser uma solução importante para um contexto de pandemia como a de COVID-19, evitando aglomerações nos estabelecimentos de saúde.

Porém, as soluções de chatbot na saúde costumam estar presentes apenas nas grandes instituições de saúde privadas, que em sua maioria se encontram nas capitais ou em cidades de grande população. Além disso, nas pesquisas realizadas, foram encontradas apenas alternativas pagas de *chatbots* para a realização de atendimentos na área da saúde, a exemplo, as soluções da Globalbot[3] e Take Blip[4]. Órgãos públicos de saúde e estabelecimentos de saúde privados e menores não costumam realizar seus atendimentos utilizando *chatbot.*

Nesse contexto, este trabalho apresenta uma solução open source de *chatbot*, voltada para o uso em órgãos públicos e instituições de saúde privadas menores. O objetivo é incentivar o uso de *chatbots* na saúde por conta dos benefícios citados anteriormente e promover uma melhoria na qualidade dos atendimentos, tanto em

[3] https://www.take.net/solucoes/saude

[4] https://globalbot.com.br/

capitais como em municípios menores onde as instituições de saúde não podem ou não querem pagar por esse tipo de solução.

O restante desse artigo está organizado da seguinte maneira: na Seção 2 são apresentados trabalhos com soluções semelhantes a deste artigo; na Seção 3 é apresentada a arquitetura, tecnologias e módulos da plataforma desenvolvida; na Seção 4 são apresentados os resultados da plataforma *web* e do *chatbot*; e por fim as considerações finais e trabalhos futuros são apresentados na Seção 5.

2 TRABALHOS RELACIONADOS

Durante a realização deste trabalho, foram realizadas pesquisas no Google Acadêmico no intuito de encontrar projetos similares à solução desenvolvida. Alguns projetos foram encontrados na literatura com soluções parecidas com a deste trabalho. É interessante notar que os trabalhos apresentados a seguir foram realizados a partir do ano de 2021, considerando o período de 2017 a 2022 definido nas pesquisas, mostra um possível crescimento no desenvolvimento desse tipo de solução nos últimos anos.

2.1 DOUTORA SARA

O primeiro trabalho encontrado foi da assistente virtual Doutora Sara de Silva e Campos (2021). Ela foi desenvolvida utilizando o Dialogflow, plataforma de processamento de linguagem natural (PLN) que facilita o design e a integração de agentes conversacionais em aplicativos para dispositivos móveis, sistemas web e até com assistentes residenciais como Google Home e Alexa.

Silva e Campos (2021) realizaram a integração do Dialogflow com o WhatsApp. Para tal, foi criado um módulo de comunicação, pois o WhatsApp não fornece uma API pública de comunicação gratuita. Esse módulo foi criado utilizando uma biblioteca de Node.js chamada Puppeteer que serve para automatização de tarefas em navegadores, sendo comumente utilizado na criação de teste automatizados e *Web Scraping*. O uso dessa biblioteca talvez não seja a maneira mais eficiente de realizar a tarefa de comunicação com WhatsApp, já que a biblioteca Puppeteer precisa de uma instância do navegador Chromium em segundo plano, algo que leva a um consumo considerável de memória. No presente artigo, o módulo Baileys, descrito na Seção 3.3, é uma forma mais eficiente de realizar a comunicação com o WhatsApp.

Além disso, no projeto de Silva e Campos (2021) também foi criada uma API utilizando a linguagem de programação JavaScript e Node.js, responsável pela persistência dos dados de usuários e consultas no banco de dados. Porém, não foi apresentada uma plataforma *web* onde os profissionais das instituições de saúde poderiam ver as consultas e gerenciar os horários de atendimento como apresentado na Seção 4.1. Não é possível afirmar com certeza se o assistente virtual Doutora Sara trata-se de um projeto *open source*, pois nenhum link para um repositório de código foi encontrado no artigo.

2.2 CONECTE SUS

Na busca por trabalhos com soluções semelhantes no sistema de saúde pública, o artigo de Celuppi et al. (2021) apresenta uma solução que não se trata de um *chatbot*, mas possui finalidade semelhante. Essa solução é o Sistema de Agendamento Online desenvolvido pelo Laboratório Bridge da Universidade Federal

de Santa Catarina que permite o agendamento de consultas por meio do aplicativo Conecte SUS Cidadão.

Como não se trata de um projeto *open source*, não foi possível descobrir muitas informações sobre a arquitetura e tecnologias utilizadas no projeto. Mas foi apresentado por Celuppi et al. (2021) que o gerenciamento das consultas e horários dos profissionais são realizados pelo sistema de Prontuário Eletrônico do Cidadão (PEC e-SUS APS) também desenvolvido pelo Laboratório Bridge.

Visto que a principal finalidade do PEC e-SUS APS não é a realização de agendamentos, ficaram visíveis as diferenças entre o sistema de agendamento do Laboratório Bridge e o deste trabalho.

Como a solução apresentada por Celuppi et al. (2021) não se trata de um *chatbot*, a instalação de um novo aplicativo pode não ser tão conveniente quanto utilizar um aplicativo já familiar, como o WhatsApp, que é o mais utilizado no Brasil de acordo com a Forbes (2022). Além disso, muitas pessoas, principalmente as de baixa renda, podem não ter espaço suficiente no celular para mais um aplicativo instalado.

2.3 SISTEMA DE AGENDAMENTOS DO CONSULTÓRIO ODONTOLÓGICO WORLD DENTAL

O trabalho encontrado mais próximo da solução deste artigo é o de Rivera Reyes e Román Amariles (2022). Os dois desenvolveram uma aplicação *web* com *chatbot* para a rede de consultórios odontológicos World Dental situada na cidade de Guayaquil no Equador. Com o objetivo de fornecer mais praticidade na administração dos agendamentos que até então eram realizados de forma manual pelo odontólogos utilizando ferramentas como Excel. Além disso, a aplicação também trouxe a facilidade dos clientes de poderem realizar seus agendamentos de forma online.

O sistema desenvolvido por Rivera Reyes e Román Amariles (2022), possui as funcionalidades de gerenciamento dos agendamentos, de realização de agendamentos de consultas por meio de *chatbot*, notificação de proximidade da consulta e notificação de cancelamento da consulta. Para o desenvolvimento do projeto Rivera Reyes e Román Amariles (2022), fizeram o uso de tecnologias Microsoft no *backend* com o framework ASP.NET Core e a linguagem de programação C#, no *frontend* foi utilizado a biblioteca JavaScript React.js junto ao framework Bootstrap para criação das interfaces usuários.

O *chatbot* de Rivera Reyes e Román Amariles (2022) foi criado utilizando o Azure Bot Service junto ao Bot Framework, tecnologias da Microsoft que possuem custos e foram assumidos pela proprietária do consultório (RIVERA REYES e ROMÁN AMARILES, 2022). Eles também fizeram o uso do WhatsApp por meio da plataforma Twilio, mas apenas para notificar os clientes da proximidade de sua consulta ou do cancelamento do agendamento.

Diferente de nossa solução, a aplicação desenvolvida por Rivera Reyes e Román Amariles (2022) não se trata de um projeto *open source*. Além disso, em nossa implementação preferimos utilizar o WhatsApp, por motivos já mencionados anteriormente, como o ambiente para o *chatbot* em vez de fazê-lo integrado ao módulo *web*. Outra diferença é que em nosso trabalho não foram utilizadas ferramentas externas no desenvolvimento do *chatbot* que tragam custos para o funcionamento do projeto.

3 ARQUITETURA E TECNOLOGIAS

Esta seção apresenta os componentes presentes na plataforma Open Schedule. Devido a finalidade do projeto foi imprescindível o uso de tecnologias códigos aberto com comunidades ativas de desenvolvedores.

O projeto foi dividido em três módulos. O primeiro apresentado é o módulo de negócio que consiste em uma API (*Application Programming Interface*) desenvolvida em python responsável pelas principais regras de negócios do sistema, sendo crucial para o funcionamento da plataforma. O segundo é o módulo web criado utilizando o framework Vue.js, essa será a parte da plataforma utilizada pelos profissionais das instituições de saúde no gerenciamento dos agendamentos e horários. O terceiro é o módulo de comunicação que realiza a integração com o WhatsApp e o processamento de linguagem natural, este é o módulo do *chatbot* que interage com os usuários para os agendamentos e cancelamentos de consultas.

A Figura 1 apresenta a arquitetura da Open Schedule, mostra como é realizada a comunicação entre os módulos.

A aplicação *web* e o *chatbot* são clientes da nossa REST (*Representational State Transfer*) API, que é a interface que proporciona uma comunicação mais amigável com o banco de dados utilizando o padrão JSON (*JavaScript Object Notation*).

Cada um dos três módulos possui seu repositório criado utilizando a ferramenta Git responsável pelo versionamento do código. Os repositórios estão hospedados no GitHub, o maior repositório de código aberto da internet, com os nomes openschedule-api[5], openschedule-admin[6] e openschedule-chatbot[7]. Os três repositórios estão sobre a licença de código aberto BSD de 3 cláusulas[8], que permite que qualquer pessoa use, modifique e distribua o software para qualquer finalidade, mas exige a inclusão um aviso de direitos autorais e uma cópia da licença quando distribuir o software e proíbe expressamente que uso do nome do autor ou do projeto para promover o software sem a permissão prévia do autor (ENGELFRIET, 2009).

3.1 MÓDULO DE NEGÓCIO

Visto que os módulos web e de comunicação acessam o mesmo banco de dados e também teriam as mesmas regras de negócio, a criação de uma API para ser a interface com o banco de dados e compartilhar as regras de negócio de forma consistente e padronizada, foi de suma importância para facilitar o desenvolvimento dos outros componentes da plataforma. Mais especificamente, foi utilizado o padrão de design REST (*Representational State Transfer*) no desenvolvimento da API, esse padrão se baseia no protocolo HTTP (*Hypertext Transfer Protocol*) utilizando os verbos GET, POST, PUT e DELETE para indicar a operação realizada e os dados são transmitidos no formato JSON (*JavaScript Object Notation*) (BATTLE e BENSON, 2008).

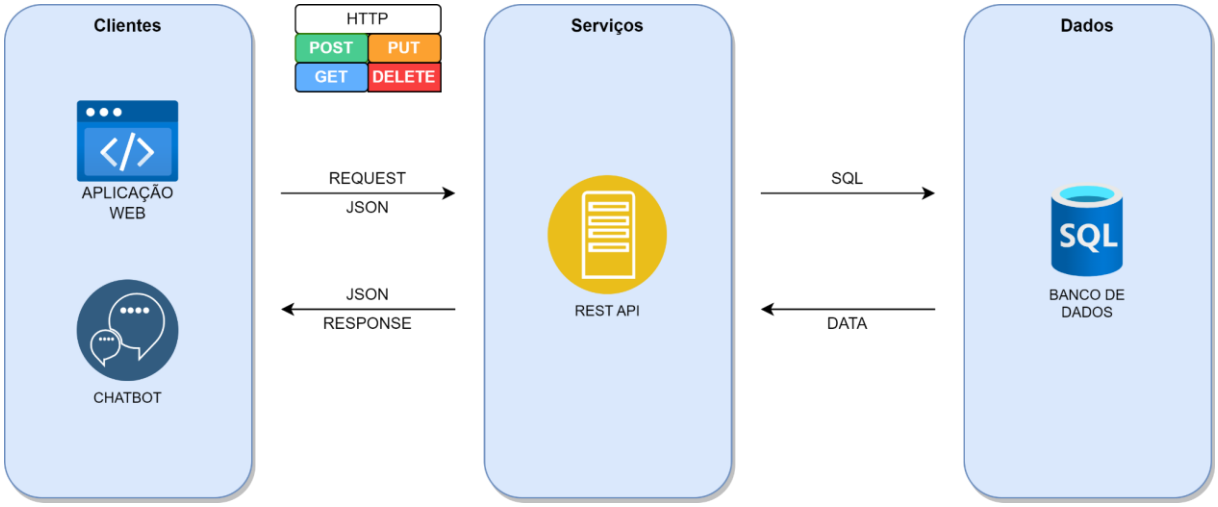
[5] https://github.com/uchoamp/openschedule-api

[6] https://github.com/uchoamp/openschedule-admin

[7] https://github.com/uchoamp/openschedule-chatbot

[8] https://opensource.org/licenses/BSD-3-Clause

Figura 1. Arquitetura da Open Schedule



A linguagem de programação Python na versão 3.10 com a biblioteca Flask foi utilizada no desenvolvimento da API REST. A escolha dessa linguagem se deu pelo fato de sua popularidade (STACK OVERFLOW, 2022) com uma grande comunidade ativa de desenvolvedores. Também é uma linguagem simples e rápida de aprender, possuindo uma sintaxe de fácil leitura e escrita possibilitando que programadores se concentrem na resolução do problema em vez de lidar com sintaxes complexas e verbosas de outras linguagens de programação. Python também possui uma ampla variedade de bibliotecas disponíveis que facilitam tarefas comuns no desenvolvimento de aplicações, como a manipulação de datas, além disso, é multiplataforma e pode ser executado nos principais sistemas operacionais Windows, MacOS e Linux (LUTZ, 2001).

Por mais que seja possível criar uma API REST em Python apenas com a biblioteca padrão da linguagem, não é algo recomendado e muito menos trivial. Para isso existem bibliotecas como Flask que fornecem todo o conjunto de interfaces de mais alto nível para criação de aplicações web. Flask foi escolhido como o framework para auxiliar no desenvolvimento de nossa API devido sua popularidade que assim como o Django são as bibliotecas mais populares no desenvolvimento de soluções para internet em Python (STACK OVERFLOW, 2022). Flask é um *microframework*, isso significa que ele é mais leve que o Django, porém possui menos recursos incluídos por padrão, mas isso tem seus lados positivos, pois torna o Flask mais fácil de aprender e implementar, também traz mais liberdade para os desenvolvedores sobre o design da aplicação. Devido sua extensibilidade, Flask pode ser facilmente incorporado às várias extensões disponíveis adicionando mais funcionalidades e tornando o *microframework* mais robusto (GHIMIRE, 2020). Além disso, ele possui uma ótima documentação e uma ampla comunidade de desenvolvedores para obter o suporte necessário. Criar soluções com Flask costuma ser mais rápido que com outros *Web frameworks*, devido sua simplicidade e flexibilidade, algo que o torna uma ótima opção para o desenvolvimento de protótipos e MVPs (*Minimum Viable Product*).

A forma como os dados são persistidos é algo muito importante para qualquer sistema. Alterações nessa camada além de serem mais difíceis, impactam em todas aplicações da plataforma. Sendo assim, a definição correta do esquema do banco e qual o SGBD (Sistema de Gerenciamento de Banco de Dados) relacional ou não relacional (NoSql), pode ser crucial para o futuro do projeto. Sabendo disso, optou-se pelo uso de um banco de dados relacional em vez NoSql (Not Only SQL), primeiro

pela maturidade dos bancos relacionais, pois as principais soluções de NoSql lançadas, MongoDB e Cassandra, ainda não possuem 15 anos desde de sua primeira versão. Alguns dos principais SGBDs relacionados como MySQL, PostgreSQL e Microsoft SQL Server foram lançados no século passado e já foram utilizados pelos mais diversos sistemas nesse período. Nosso projeto não faria proveito de um dos principais benefícios do NoSql que é a escalabilidade, pois não é nosso intuito criar uma plataforma que use tantos recursos ao ponto da necessidade de ser escalável. Outro ponto em prol dos bancos relacionais, são as visíveis relações entre as várias entidades do sistema. A exemplo de algumas, a entidade “horários” é relacionada com as entidades “clínicas” e “profissionais”, já a entidade “agendamentos” é relacionada com as entidades “profissionais” e “pacientes”.

A linguagem SQL (Structured Query Language) é utilizada para gerenciar os bancos de dados relacionais. Com ela é possível criar, alterar e consultar os registros do banco de dados. Ainda que SQL não seja uma linguagem difícil, não é recomendável inserir consultas de SQL dentro de um script de Python. Além de visualmente não ser agradável, pode trazer riscos à segurança, como ataque de *SQL injection* se os parâmetros das consultas não forem tratados corretamente.

Como meio de facilitar a comunicação com os SGBDs, existem os ORMs (Object-Relational Mapping), que realizam o mapeamento de objetos na linguagem de programação para dados relacionais e vice-versa. Isso permite que os programadores acessem e manipulem os dados do banco de dados de uma forma mais natural, usando os métodos e propriedades dos objetos em vez de escrever consultas SQL manualmente.

O ORM escolhido para o desenvolvimento da nossa API foi o SQLAlchemy que é o equivalente em Python ao Hibernate de Java e o Entity Framework da plataforma .NET. Ele foi facilmente integrado ao Flask com uso de uma extensão. Com o SQLAlchemy foi possível criar todas as tabelas do banco e realizar consultas no banco de dados sem a escrita de uma única instrução em SQL. Além disso, o SQLAlchemy possibilita facilmente a troca do SGBD entre os vários suportados, como SQLite, Postgresql, MySQL, Oracle e MS-SQL.

O SGBD escolhido para o desenvolvimento de nossa solução foi o MariaDB que se trata de uma bifurcação do MySQL criada devido às incertezas sobre sua continuidade como *software open source* e sua qualidade após ser adquirido pela Oracle em 2009 (RAZZOLI, 2014). Trabalhos como os de BARTHOLOMEW (2012) ainda mostram superioridade do MariaDB em relação ao MySQL, que vai desde recursos extras até melhorias de desempenho.

REST APIs costumam possuir várias aplicações clientes, que normalmente são desenvolvidas por programadores diferentes. Por isso, uma boa documentação é necessária para facilitar no desenvolvimento de soluções que vão consumir a API. Nesse sentido, utilizamos Swagger[9] que consiste no conjunto de ferramentas utilizadas para o *design* e documentação de APIs. Swagger segue a OpenAPI Specification (OAS) que é o padrão para descrever APIs REST. Com ele é possível descrever APIs de forma mais legível para humanos utilizando JSON ou YAML (KOREN e KLAMMA, 2018). Além disso, Swagger possui ferramentas que ajudam na realização de testes como a Swagger UI, que possibilita o envio de solicitações para a API.

3.2 MÓDULO *WEB*

[9]https://swagger.io/docs/

Para que os profissionais das instituições de saúde vejam os agendamentos e gerenciem os horários de atendimento, foi criada uma aplicação *web*, mais especificamente uma SPA (*Single-Page Application*). Diferente das aplicações *web* tradicionais que carregam uma nova página cada vez que o usuário navega para uma nova seção da aplicação, nas SPAs apenas uma página HTML com muito JavaScript é carregado no navegador, o JavaScript fica encarregado de alterar dinamicamente o conteúdo da página a cada ação do usuário (SCOTT JR, 2014). Isso torna a experiência do usuário mais suave e rápida, lembrando os aplicativos instalados no dispositivo.

A criação de uma SPA apenas com JavaScript sem o uso de bibliotecas é algo consideravelmente trabalhoso e lento. Para o desenvolvimento fácil e rápido dessas soluções existem *frameworks open source* disponíveis, sendo os três mais populares React, Vue e Angular (STACK OVERFLOW, 2022).

O *framework* Vue foi escolhido para o desenvolvimento deste módulo. Os motivos para sua escolha foram sua sintaxe simples e intuitiva, facilidade de aprendizado, possibilita o desenvolvimento rápido da aplicação e a leveza do *framework* o que torna mais rápido o carregamento da página. Além de ser facilmente integrado com outras bibliotecas e frameworks (NELSON, 2018). Além disso, Vue possui uma comunidade ativa com uma ampla gama de bibliotecas e ferramentas disponíveis para ajudar no desenvolvimento de aplicações.

Vue é compatível com TypeScript, linguagem que foi preferível ao JavaScript tanto neste como no módulo apresentado na próxima seção. TypeScript é a extensão do JavaScript que tem como objetivo resolver as deficiências da linguagem. Uma das principais características do TypeScript é a adição da tipagem estática (BIERMAN et al., 2014). O uso de TypeScript em vez do JavaScript traz vários benefícios como mais facilidade na manutenção e na depuração. Os erros são encontrados antes do código ser executado devido à sua integração aprimorada com IDEs (Integrated Development Environment.) e *linters* (BIERMAN et al., 2014). Essas características são importantes em projetos maiores, para manterem seu código organizado e funcionando corretamente de maneira mais fácil à medida que o projeto cresce.

3.3 MÓDULO DE COMUNICAÇÃO

No módulo de comunicação é onde se encontra o principal motivo para a idealização desse projeto. O *chatbot,* que realiza a integração com o aplicativo de mensagens WhatsApp e interage com o usuário fornecendo as funcionalidades de realização de agendamento, ou reagendamento em caso de uma consulta já marcada, e também o cancelamento dos agendamentos. Além disso, é possível o usuário ver seus agendamentos e também obter informações como a localização do estabelecimento de saúde.

Antes de continuarmos com as tecnologias utilizadas no desenvolvimento de nosso *chatbot*, é importante aprofundar um pouco mais sobre os *chatbots* além do que foi apresentado na Seção 1. Para Shawar e Atwell (2007), *chatbots* são programas de computador que interagem com humanos utilizando linguagem natural, por meio de texto ou voz. O trabalho de Adamopoulou e Moussiades (2020), classifica os *chatbots* usando diferentes parâmetros. De acordo com as classificações de Adamopoulou e Moussiades (2020) o *chatbot* deste trabalho seria de domínio fechado, interpessoal, baseado em tarefas.

Um dos parâmetros de classificação utilizado por Adamopoulou e Moussiades (2020), foi o de processamento da entrada do usuário e o método de geração de resposta. Nesse ponto, eles classificam os *chatbots* em três modelos, dois desses modelos têm diferenciais consideráveis que são o modelo baseado em regras e o modelo generativo. O modelo baseado em regras é programado com uma série de regras pré-definidas que determinam como ele deve responder a determinadas entradas do usuário. Esses chatbots específicos são geralmente fáceis de criar e implementar, no entanto, são limitados em sua capacidade de responder entradas mais complexas, ou até mesmo com apenas erros de ortografia. Já o modelo generativo usa algoritmos *machine learning* e técnicas de *deep learning* para gerar respostas autônomas ao invés de simplesmente selecionar respostas pré-definidas. Esses tipos de *chatbot* são mais parecidos com humanos e não possuem as limitações como o modelo baseado em regras, porém sua complexidade torna seu desenvolvimento consideravelmente mais difícil além de necessitar de mais recursos computacionais.

A criação de um *chatbot* baseado em regras, que fornece todas as funcionalidades já apresentadas no primeiro parágrafo desta seção, foi preferível a de um *chatbot* de inteligência artificial (IA) devido à complexidade na implementação dessa tecnologia. Ainda que existam plataformas como Dialogflow que facilitam o uso de IA no desenvolvimento *chatbots*, mas *os custos* que elas trazem não são bons para a finalidade de nosso projeto. Além disso, como apresentado na Seção 4, mesmo sendo mais simples, os *chatbots* baseado em regras podem oferecer formas intuitivas de interação com o usuário e ótima usabilidade.

Como já foi mencionado na seção anterior esse módulo também foi desenvolvido utilizando a linguagem de programação TypeScript. A pesquisa realizada anualmente pelo Stack Overflow (2022) mostra as linguagem JavaScript no top da lista de tecnologias mais utilizadas, isso claro devido várias de suas características, principalmente por ser uma linguagem de mais alto nível e mais fácil de aprender do que outras linguagens de mais baixo nível como C ou Assembly. TypeScript, considerado uma extensão de Javascript por Bierman et al. (2014), não fica muito atrás do JavaScript sendo a terceira linguagem de programação mais popular atrás de Python. A escolha de TypeScript foi devido trazer a parte boa do JavaScript como a facilidade de aprendizado, versatilidade, pode ser utilizado tanto no lado do servidor quanto do cliente e sua gigantesca comunidade, muito forte principalmente no *open source*, mas possui suas deficiências que em sua maioria são resolvidas com a tipagem estática introduzida no TypeScript (CHERNY, 2019).

A linguagem JavaScript foi inicialmente criada com objetivo de fornecer mais interatividade às páginas *web* que eram estáticas, sendo executado principalmente em navegadores de *internet*. Com a criação do Node.js foi então possível utilizar JavaScript também no desenvolvimento de aplicações no lado do servidor (CHERNY, 2010). Node.js foi criado usando o motor JavaScript V8, utilizado pelo navegador Google Chrome para executar JavaScript, reconhecido pelo seu ótimo desempenho que também é visto no Node.js (CHERNY, 2010). Hoje Node.js é extremamente popular como mostra Stack Overflow (2022), sendo utilizado no desenvolvimento em soluções nas mais diversas áreas, como aplicativos de servidor, aplicativos de *desktop* e aplicações de IoT (Internet das Coisas) (CHERNY, 2010).

A plataforma Node.js não consegue interpretar TypeScript, porém ele possui seu próprio compilador, que “transpila” o código em TypeScript para JavaScript (CHERNY, 2019). Isso é semelhante ao processo de compilação tradicional, mas com

diferença de que em vez da compilação ser para um código de mais baixo nível é para outra linguagem de alto nível (REISER e BLÄSER , 2017).

Para realizar a comunicação com o WhatsApp foi necessário a instalação da biblioteca Baileys[10]. Essa biblioteca faz uso dos *WebSocket* do WhatsApp Web para conseguir enviar e receber mensagens. O uso dessa biblioteca foi necessário pois diferente do Telegram, WhatsApp não fornece uma API pública e gratuita para comunicação. Além disso, o acesso para a API oficial do WhatsApp é fornecido apenas para empresas de grande e médio porte. Mesmo não sendo oficial, a biblioteca Baileys oferece recursos encontrados na API oficial como o envio de mensagens com botões.

O WhatsApp foi escolhido como o ambiente para nosso *chatbot* em vez do Telegram, pois é o aplicativo de mensagens mais utilizado no Brasil Forbes (2022), ficando muito à frente do Telegram na quantidade de usuários. Esse ponto foi mais importante para nossa decisão do que as facilidades trazidas pela API pública do Telegram na integração.

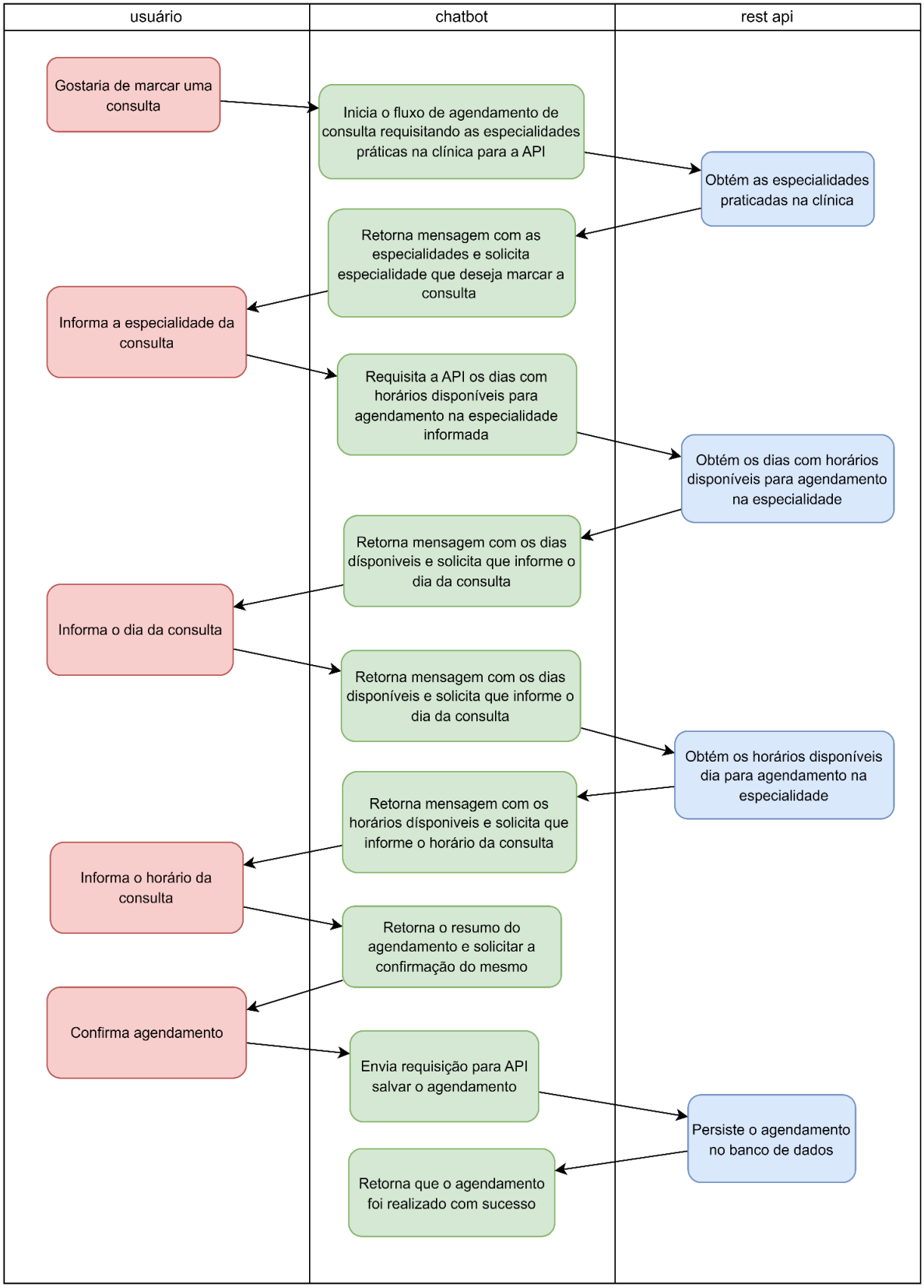
A biblioteca baileys foi crucial para escolha do Node.js e TypeScript no desenvolvimento de nosso módulo de comunicação, pois é escrito em TypeScript e executado na plataforma Node.js. Também utilizamos a biblioteca Axios[11] como nosso cliente HTTP para realizar as consultas em nossa REST API.

A Figura 2 apresenta o fluxo desenvolvido para o agendamento de uma consulta, mostrando como o usuário interage com o *chatbot* e as requisições que são realizadas para nossa REST API durante o processo de marcação da consulta.

[10] https://github.com/adiwajshing/Baileys

[11] https://github.com/axios/axios

Figura 2. Página gerada pelo Swagger com a documentação da Open Schedule API



4 OPEN SCHEDULE

Nesta seção são apresentadas as principais funcionalidades implementadas no projeto. Para o uso das funcionalidades temos duas plataformas com finalidades e usuários diferentes. A primeira aplicação apresentada será o painel de administração que faz parte do módulo *web* e tem como usuários os profissionais das instituições de saúde. A segunda aplicação apresentada é o *chatbot* do módulo de comunicação que tem como usuários os pacientes dos estabelecimentos de saúde.

4.1 PAINEL DE ADMINISTRAÇÃO

O painel de administração foi criado para que os profissionais dos estabelecimentos de saúde possam ver os agendamentos realizados, realizar o cadastro de profissionais com sua especialidade, assim podendo definir os horários de atendimento. Para acesso ao painel administrativo é necessário um usuário e senha.

Existem dois tipos de usuários na plataforma. Primeiro, o usuário administrador possui todas as permissões possíveis para alterar as entidades do sistema, a Figura 3 apresenta como é a página inicial quando logado com um usuário administrador. Segundo, o usuário profissional possui menos permissões que o usuário administrador, esse usuário pode acessar apenas seus agendamentos e realizar alteração apenas em seus horários de atendimento, a Figura 4 apresenta como é a página inicial quando logado um usuário profissional. A principal diferença entre esses usuários é a maior quantidade de funções presentes no menu à esquerda para o usuário administrador.

O primeiro acesso na plataforma sempre será com um usuário administrador, para que seja realizado os cadastros de estabelecimentos, especialidades, profissionais e horários. A Figura 5 mostra a página vista no primeiro acesso.

Após o primeiro acesso deve ser utilizado o menu lateral para cadastrar as informações do estabelecimento de saúde (Figura 6) e também as especialidades da instituição (Figura 7).

Com os dados do estabelecimento e especialidades devidamente cadastrados, é possível realizar os cadastros dos profissionais responsáveis pelos atendimentos, como mostra a Figura 8. Com o profissional cadastrado é possível definir seus horários de atendimento como na Figura 9. A Figura 10 mostra a listagem dos horários do profissional selecionado.

O profissional poderá utilizar o usuário e senha cadastrada para realizar seu acesso. Uma página semelhante à Figura 11 será apresentada com os horários e agendamentos da semana, os retângulos com cor azul mais escuro são os agendamentos e com azul mais claro são os horários. A navegação na agenda é realizada utilizando os botões no menu superior ou calendário que se encontra na lateral, assim é possível visualizar os agendamentos das semanas anteriores ou posteriores.

Por mais que nosso objetivo seja que os agendamentos sejam realizados pelo *chatbot*, a aplicação *web* também fornece a possibilidade de realizar agendamentos pelo painel de administração, Figura 12.

Figura 3. Painel de administração quando logado com um usuário administrador

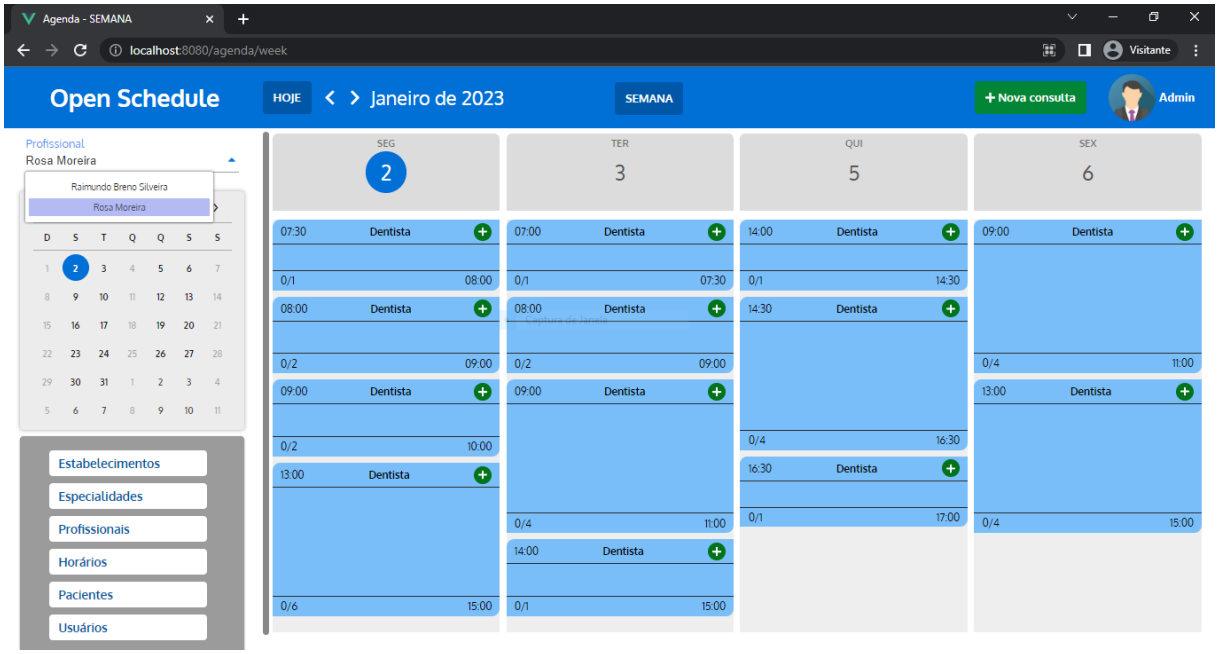


Figura 4. Painel de administração quando logado com um usuário profissional

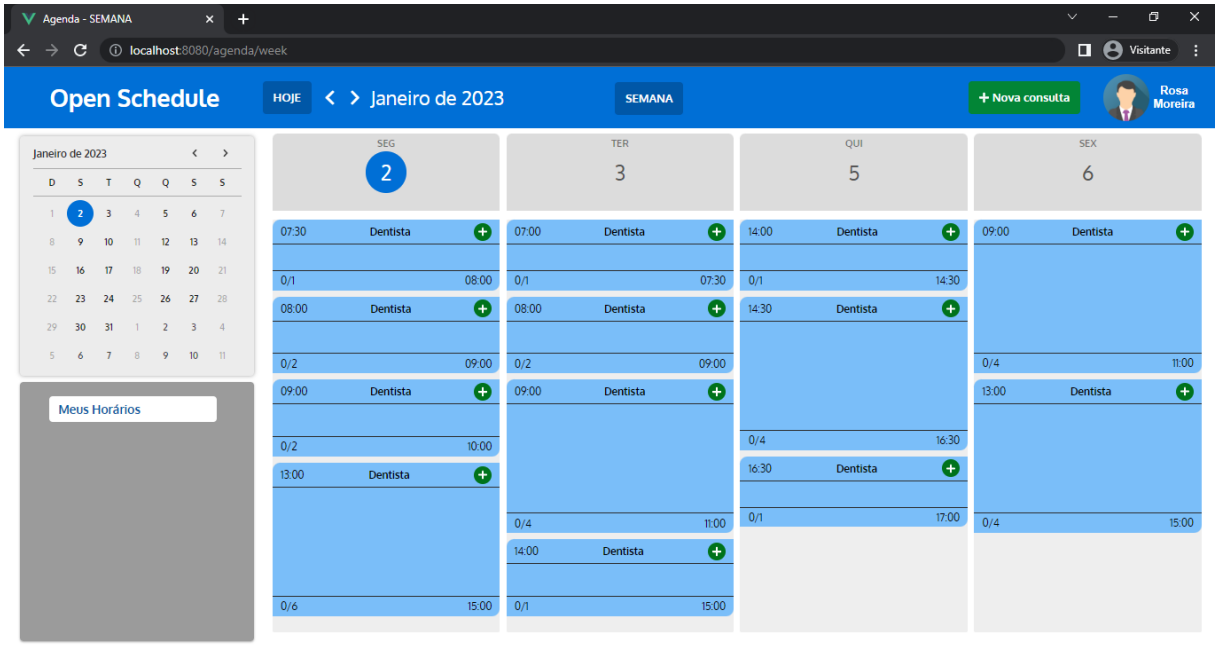


Figura 5. Primeiro acesso ao painel de administração

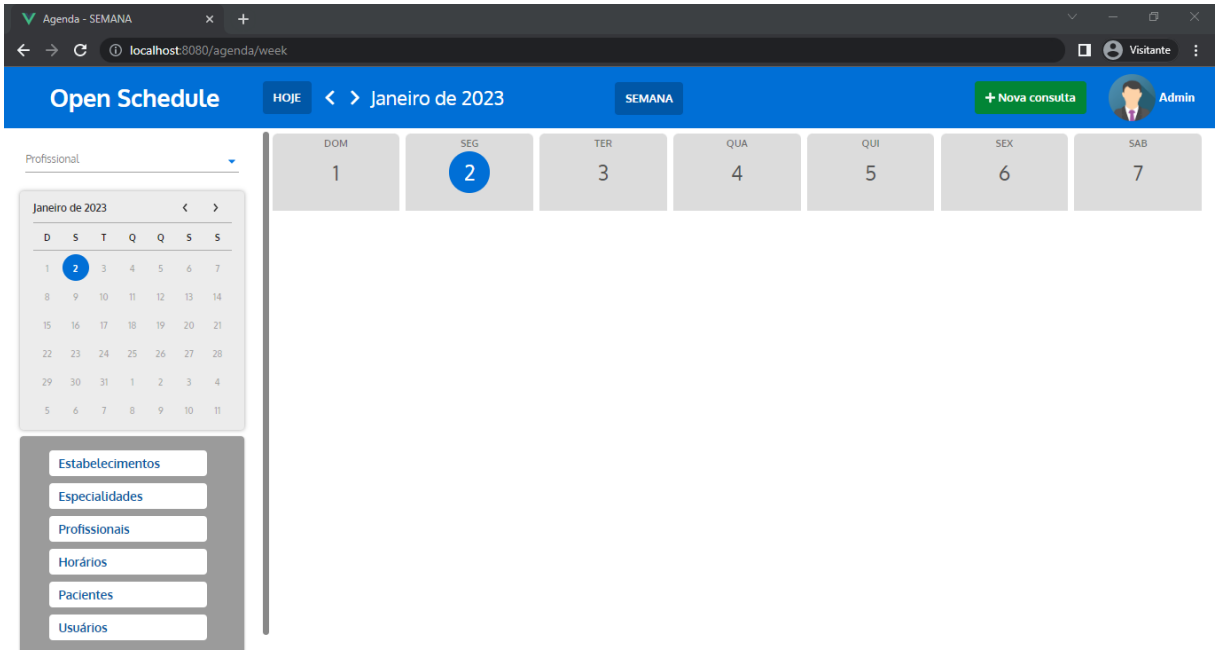


Figura 6. Cadastro dos dados do estabelecimento de saúde

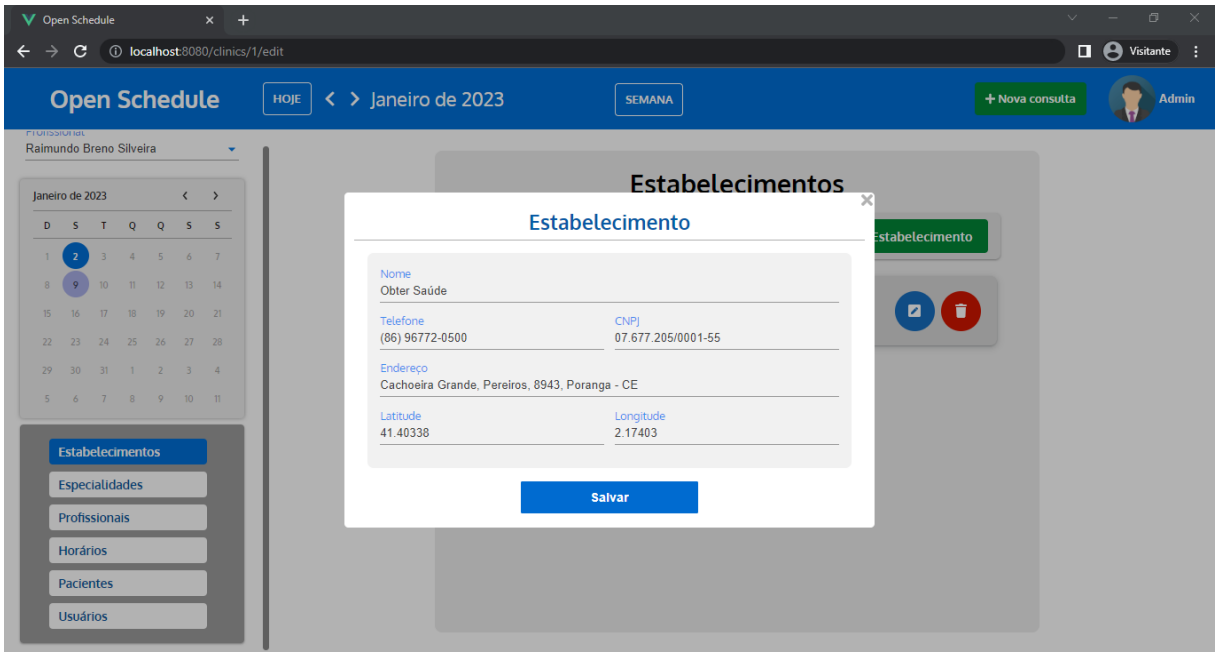


Figura 7. Cadastro das especialidades da instituição de saúde

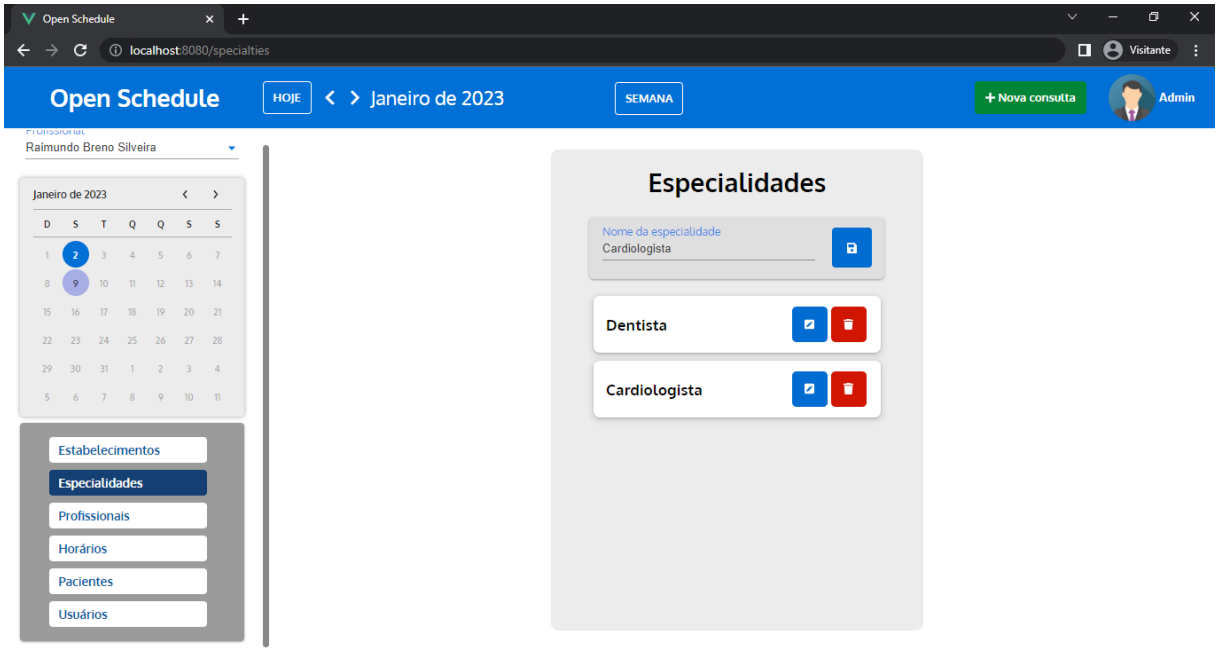


Figura 8. Cadastro do profissional para atendimento no estabelecimento de saúde cadastrado

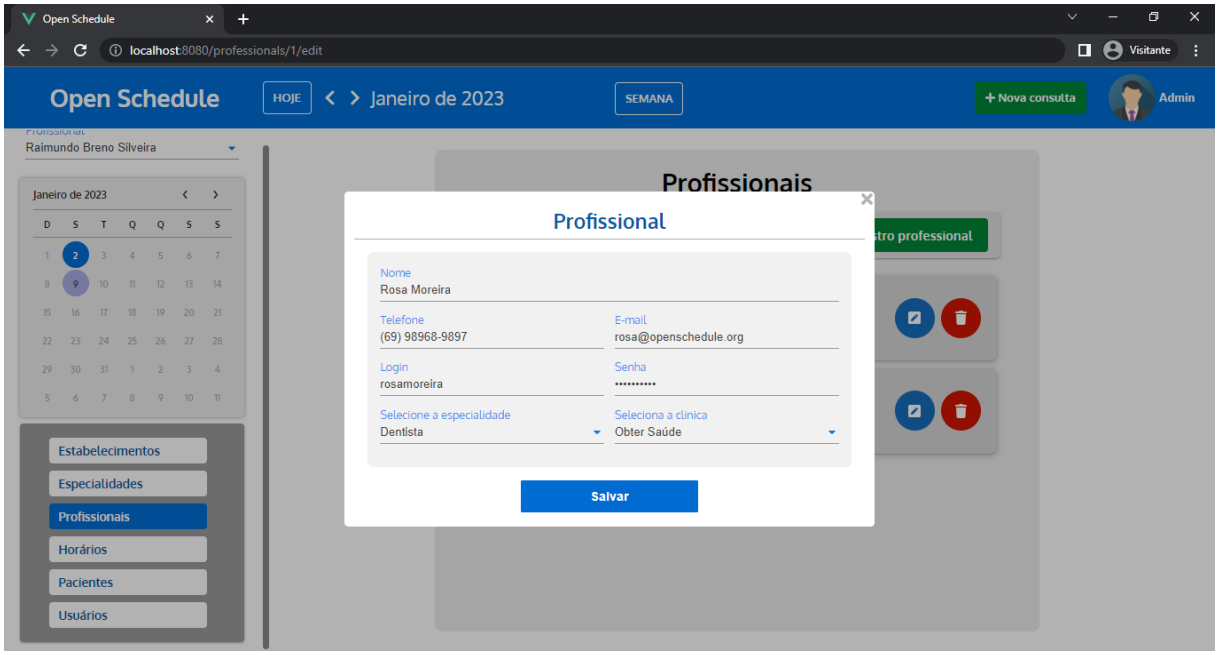


Figura 9. Cadastro do horário de atendimento do profissional

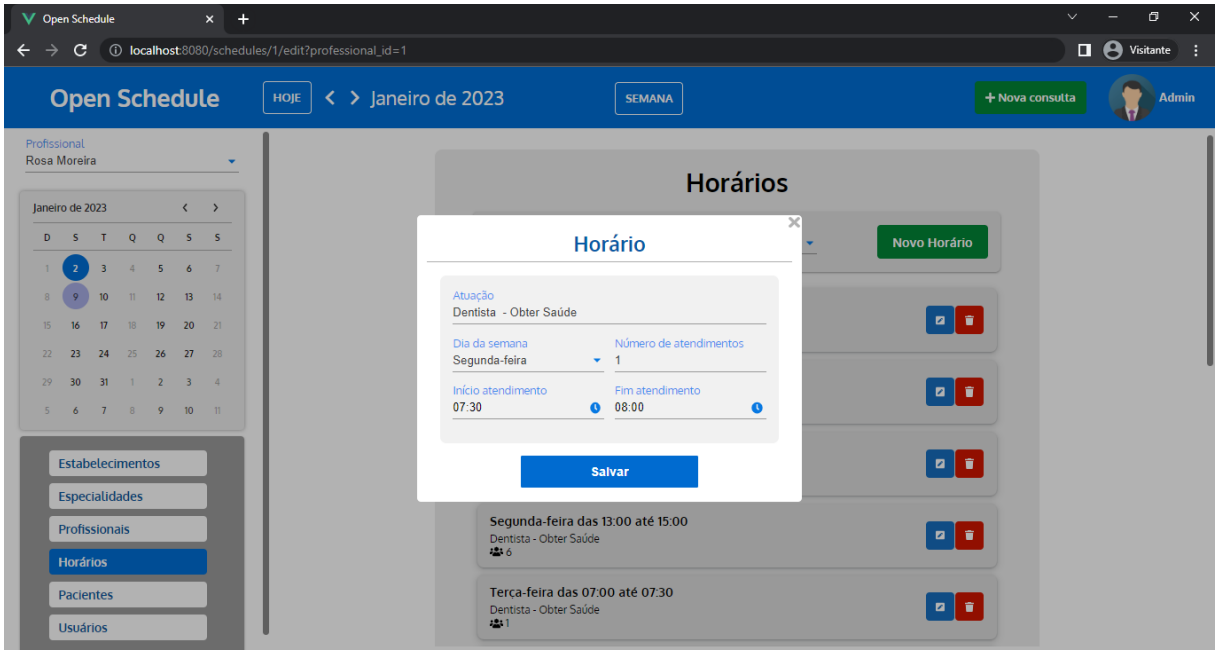


Figura 10. Listagem dos horários de atendimento do profissional

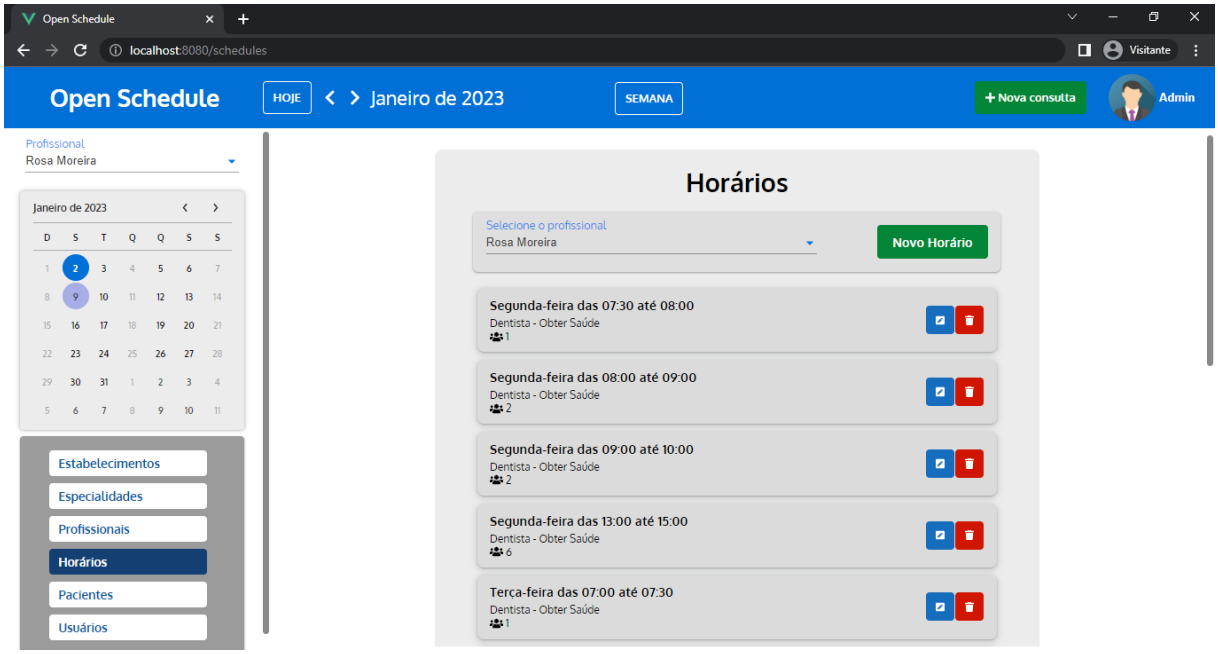


Figura 11. Agenda da semana do profissional logado

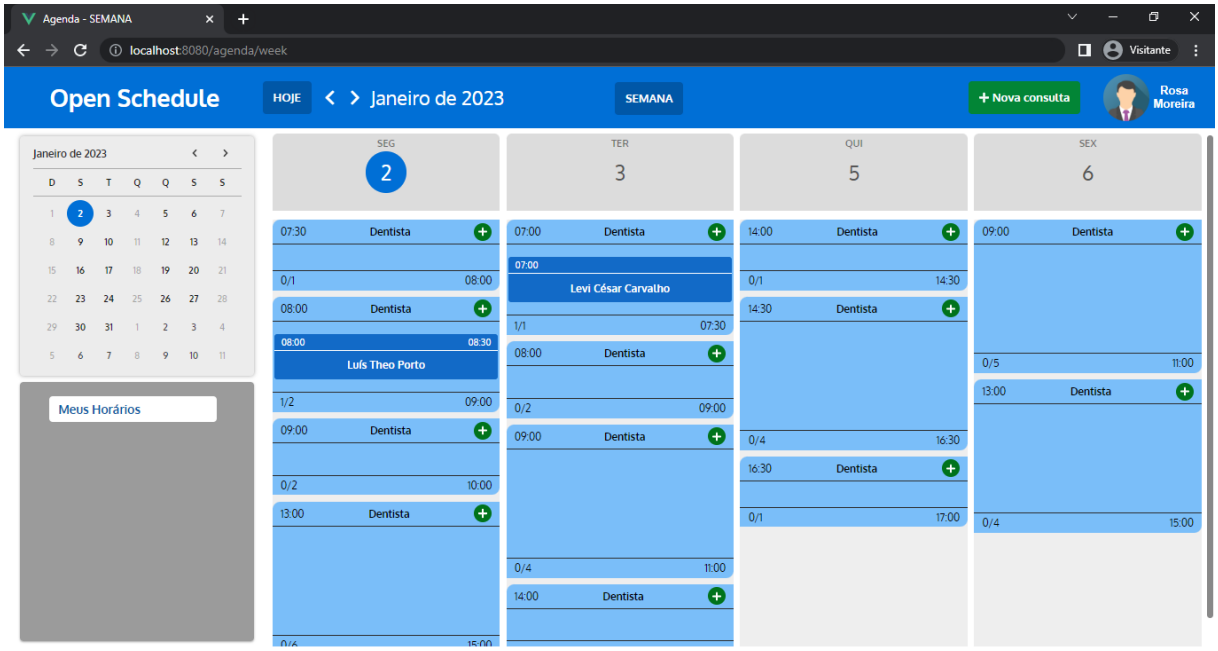
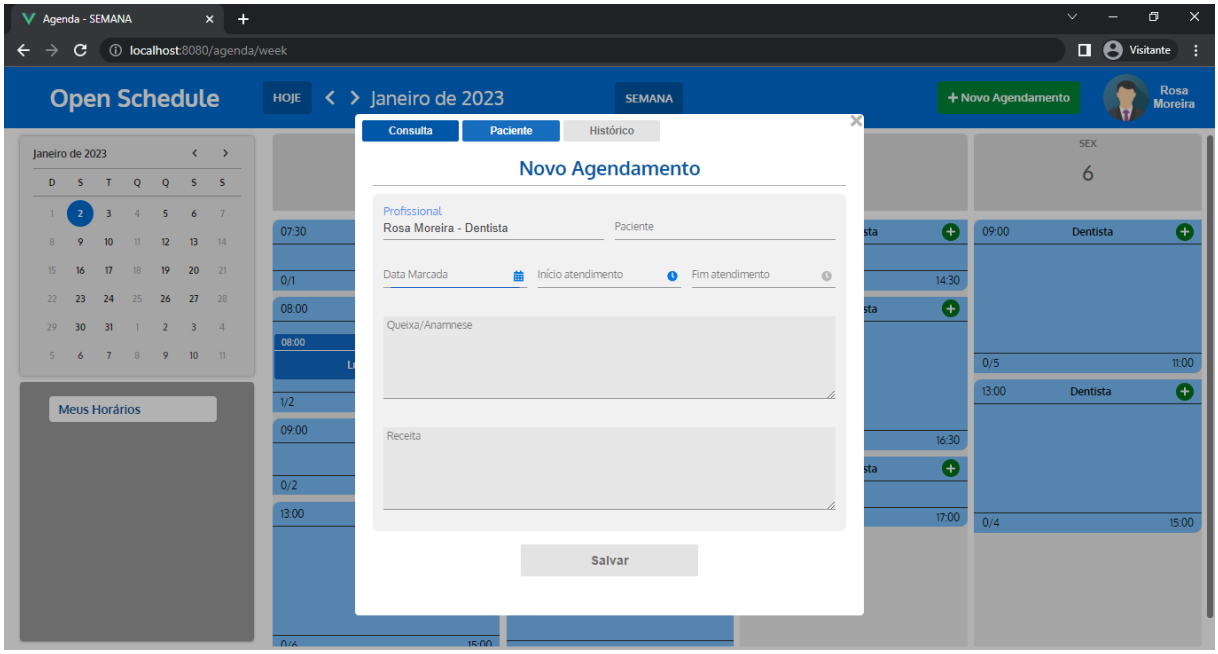


Figura 12. Formulário de agendamento pelo painel de administração



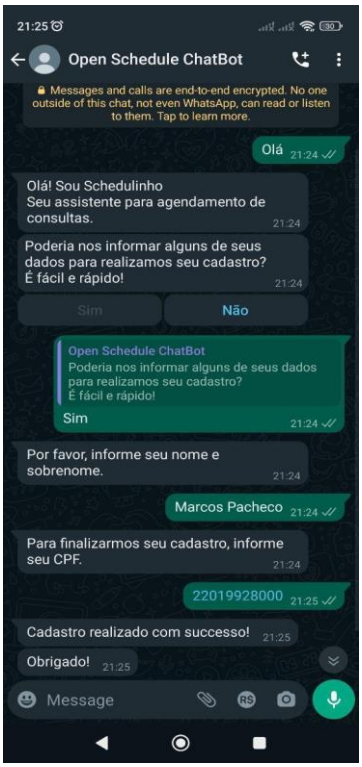
4.2 CHATBOT

O *chatbot* foi criado integrado com o aplicativo de mensagens WhatsApp de modo que seja fácil e intuitivo de ser utilizado, fornecendo funcionalidades de agendamento de consulta, cancelamento de consulta e também obter informações sobre o estabelecimento que será realizado o atendimento, além de poder ver as consultas marcadas a qualquer momento.

Na primeira interação do usuário (paciente da clínica) com chatbot, ele requisitará algumas informações do usuário para realizar seu cadastro na plataforma

(Figura 13). Nas iterações seguintes esse procedimento não é mais necessário, pois os dados já estão persistidos na base de dados e o chatbot reconhece com quem está falando.

Figura 13. Primeira interação com o chabot com a realização do cadastro do usuário



O *chatbot* fornece um menu de opções com botões tornando a interação com o usuário mais fácil e com menos erros. A possibilidade de enviar mensagens com botões e outros recursos do WhatsApp trazem facilidades aos usuários.

A partir do menu de opções é possível realizar um agendamento clicando na opção “Marcar Consulta” (Figura 14). Em seguida, inicia-se o fluxo de agendamento, com o *chatbot* fornecendo a lista de especialidade do estabelecimento de saúde (Figura 14). O usuário informa a especialidade e o *chatbot* retorna uma lista com os próximos dias que terão horários disponíveis para agendamento na especialidade (Figura 14). Com o dia informado pelo usuário (Figura 15), o *chatbot* retorna os horários disponíveis para agendamento no dia (Figura 17). O usuário seleciona o horário (Figura 16) e o *chatbot* retorna o resumo do agendamento com a data e hora marcada, a especialidade e o profissional que realizará o atendimento (Figura 17). Por fim, o *chatbot* fornece a opção de confirmar agendamento ou refazer (Figura 17), em caso do agendamento confirmado pelo usuário, o agendamento é persistido no banco de dados e o *chatbot* retorna a mensagem de agendamento realizado com sucesso (Figura 17).

Para ver suas consultas agendadas, o usuário pode utilizar o menu de opções ou escrever “Minhas consultas” como na Figura 18. O *chatbot* responderá com cada consulta em um botão, com informação de especialidade e horário e dia do agendamento (Figura 18). Ao clicar na consulta agendada é possível ver mais informações sobre o agendamento (Figura 19), também será apresentada a opção de reagendar a consulta que consiste em alterar o dia ou horário de atendimento, semelhante ao que já foi mostrado no fluxo de agendamento, além da opção de

cancelar consulta (Figura 19). Quando escolhida a opção de cancelar, o chatbot pedirá a confirmação do usuário para o cancelamento do agendamento como mostrado na Figura 19.

Além das funcionalidades relacionadas aos agendamentos, também é possível obter informações sobre o estabelecimento de saúde, como mostra a Figura 20.

Figura 14. Utiliza menu de opções para iniciar o agendamento de uma consulta

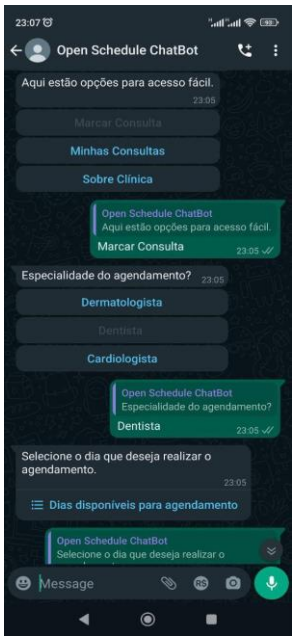


Figura 15. Seleciona dia do agendamento

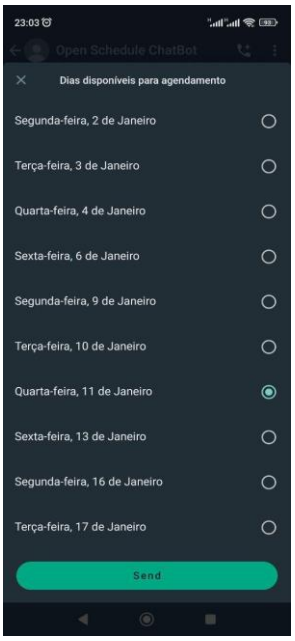


Figura 16. Seleciona horário do agendamento

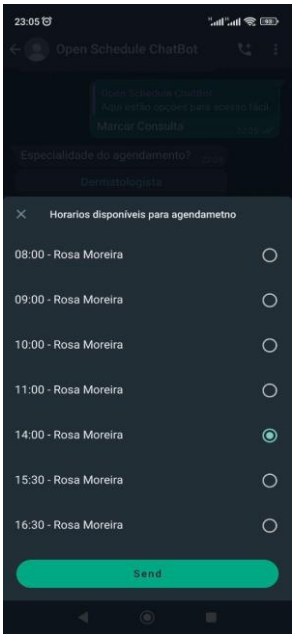


Figura 17. Confirma o agendamento

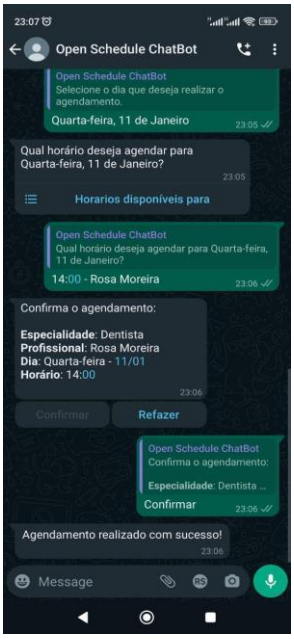


Figura 18. Visualiza consultas marcadas

Figura 19. Agendamento selecionada para cancelamento

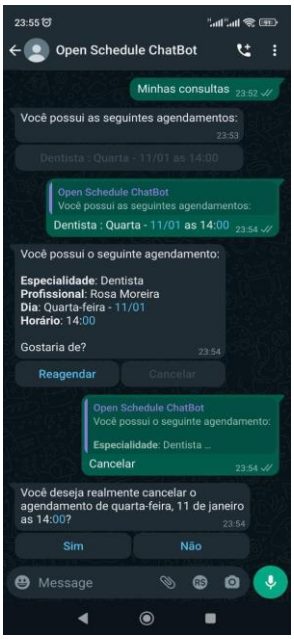


Figura 20. Obtendo informações sobre o estabelecimento de saúde

5 CONSIDERAÇÕES FINAIS

Open Schedule é um projeto de código aberto criado com intuito de democratizar o uso da tecnologia na melhoria do atendimento nas instituições de saúde, que não utilizam essas ferramentas por falta do conhecimento de suas existências ou pelos custos das opções disponíveis no mercado. O objetivo é que a plataforma seja utilizada no sistema de saúde pública e também em estabelecimentos de saúde privados que não utilizam as opções pagas devidos aos custos.

O desenvolvimento da plataforma teve o apoio de uma profissional da área de saúde, a Dra. Ana Maria, médica no IFPI (Instituto Federal do Piauí) - Campus Pedro II, que contribuiu para o levantamento dos requisitos e também com *feedbacks* das funcionalidades implementadas.

O projeto foi totalmente desenvolvido utilizando tecnologias *open source*, que possuem boas documentações, comunidades ativas que trazem recursos e ferramentas que facilitam no desenvolvimento e trazem rapidez no desenvolvimento da solução, além da facilidade de manutenção e da adição de novas funcionalidades aos módulos.

Neste trabalho foi implementado um *chatbot* que possibilita os pacientes das instituições de saúde realizarem os agendamentos de consultas através do aplicativo de mensagem WhatsApp. Também, foi criada uma aplicação *web* para uso dos profissionais do estabelecimento de saúde realizarem o acompanhamento dos agendamentos realizados pelo *chatbot*.

Como trabalhos futuros, pretende-se implantar a plataforma em instituições de saúde reais com o objetivo de entender a viabilidade do projeto e, com os feedbacks dos usuários, melhorias e novos recursos poderão ser implementados. Para tal, os autores deste trabalho têm como projeto a continuação do desenvolvimento da plataforma com a contribuição dos alunos das disciplinas de projeto integrador do curso superior de análise e desenvolvimento de sistemas e também do curso técnico de informática do IFPI - Campus Pedro II. Isso evitará que a plataforma caia no desuso e possibilitará sua evolução com a implementação de melhorias, adição de novas funcionalidades, documentação do projeto e fornecimento de suporte para as instituições de saúde que utilizarem a plataforma. Outro trabalho futuro, porém mais simples, é a integração do *chatbot* com outras plataformas de mensagem como o Telegram, trazendo mais opções para os usuários. Além disso, podem ser realizados estudos em relação ao uso da ferramenta em outras áreas, pois durante o desenvolvimento foi evitado atrelar a plataforma ao contexto da saúde, tornando-a mais facilmente adaptável a outros contextos onde existe prestação de serviço realizada através de agendamentos.

REFERÊNCIAS

ABD-ALRAZAQ, Alaa A. et al. An overview of the features of chatbots in mental health: A scoping review. **International Journal of Medical Informatics**, v. 132, p. 103978, 2019

ADAMOPOULOU, Eleni; MOUSSIADES, Lefteris. An overview of chatbot technology. In: **IFIP International Conference on Artificial Intelligence Applications and Innovations**. Springer, Cham, 2020. p. 373-383.

ALMALKI, Manal; AZEEZ, Fahad. Health chatbots for fighting COVID-19: a scoping review. **Acta Informatica Medica**, v. 28, n. 4, p. 241, 2020.

AMIRI, Parham; KARAHANNA, Elena. Chatbot use cases in the Covid-19 public health response. **Journal of the American Medical Informatics Association**, v. 29, n. 5, p. 1000-1010, 2022.

BALSA, João et al. Intelligent virtual assistant for promoting behaviour change in older people with T2D. In: **EPIA Conference on Artificial Intelligence**. Springer, Cham, 2019. p. 372-383.

BARTHOLOMEW, Daniel. Mariadb vs. mysql. **Dostopano**, v. 7, n. 10, p. 2014, 2012.

BATTLE, Robert; BENSON, Edward. Bridging the semantic Web and Web 2.0 with representational state transfer (REST). **Journal of Web Semantics**, v. 6, n. 1, p. 61-69, 2008.

BIERMAN, Gavin; ABADI, Martín; TORGERSEN, Mads. Understanding typescript. In: **European Conference on Object-Oriented Programming**. Springer, Berlin, Heidelberg, 2014. p. 257-281.

BUINHAS, Susana et al. Virtual assistant to improve self-care of older people with type 2 diabetes: First prototype. In: **International Workshop on Gerontechnology**. Springer, Cham, 2019. p. 236-248.

CARVALHO, Laryssa et al. DRA. LARA: ASSISTENTE VIRTUAL DE APOIO E ACOMPANHAMENTO AO PRÉ-NATAL. **WWW/INTERNET**, v. 2019, p. 257, 2019.

CELUPPI, Ianka Cristina et al. Sistema de agendamento online: uma ferramenta do PEC e-SUS APS para facilitar o acesso à Atenção Primária no Brasil. **Ciência & Saúde Coletiva**, v. 26, p. 2023-2034, 2021.

CHERNY, Stefan; VINOSKI, Steve. Node. js: Using JavaScript to build high-performance network programs. **IEEE Internet Computing**, v. 14, n. 6, p. 80-83, 2010.

CHERNY, Boris. **Programming TypeScript: making your JavaScript applications scale**. O'Reilly Media, 2019.

CHURCH, Karen; DE OLIVEIRA, Rodrigo. What's up with WhatsApp? Comparing mobile instant messaging behaviors with traditional SMS. In: **Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services**. 2013. p. 352-361.

DA SILVEIRA, Gabriela Silva et al. Prevalência de absenteísmo em consultas médicas em unidade básica de saúde do sul do Brasil. **Revista Brasileira de Medicina de Família e Comunidade**, v. 13, n. 40, p. 1-7, 2018.

DALE, Robert. The return of the chatbots. **Natural Language Engineering**, v. 22, n. 5, p. 811-817, 2016.

DJOELIANTO, Agoeng Dwi; KAUTSAR, Irwan Alnarus; ROSID, Mochamad Alfan. Development of Web Service and Telegram Bot for Location-Based Health Service Information System. **Procedia of Engineering and Science**, v. 2, n. 2, 2022.

ENGELFRIET, Arnoud. Choosing an open source license. **IEEE software**, v. 27, n. 1, p. 48-49, 2009.

FORBES. **6 apps mais usados pelos brasileiros.** 2022. Disponível em: https://forbes.com.br/forbes-tech/2022/08/6-apps-com-maior-frequencia-de-uso-pelos-brasileiros/. Acesso em: 26 dez. 2022.

GHIMIRE, Devndra. Comparative study on Python web frameworks: Flask and Django. 2020.

KOREN, István; KLAMMA, Ralf. The exploitation of openapi documentation for the generation of web frontends. In: **Companion Proceedings of the The Web Conference 2018**. 2018. p. 781-787.

LEAVITT, Neal. Will NoSQL databases live up to their promise?. **Computer**, v. 43, n. 2, p. 12-14, 2010.

LUTZ, Mark. Programming python. " O'Reilly Media, Inc.", 2001.

NELSON, Brett. Getting to Know Vue. js. **Getting to Know Vue. js**, 2018.

PALANICA, Adam et al. Physicians' perceptions of chatbots in health care: cross-sectional web-based survey. **Journal of medical Internet research**, v. 21, n. 4, p. e12887, 2019.

RAZZOLI, Federico. **Mastering MariaDB**. Packt Publishing Ltd, 2014.

REISER, Micha; BLÄSER, Luc. Accelerate JavaScript applications by cross-compiling to WebAssembly. In: **Proceedings of the 9th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages**. 2017. p. 10-17.

RIVERA REYES, Guillermo Oswaldo; ROMÁN AMARILES, David Sebastián. **Aplicación web con chatbot para la gestión de agendamiento, recordatorios y envío de mensajes de cancelación de citas para el consultorio odontológico World Dental**. 2022.

SCOTT JR, Emmit A. **SPA Design and Architecture: Understanding single-page web applications**. Simon and Schuster, 2015.

SHAWAR, Bayan Abu; ATWELL, Eric. Chatbots: are they really useful?. In: **Ldv forum**. 2007. p. 29-49.

SILVA, André O. CAMPOS, Leonardo B. Doutora Sara: um assistente virtual para marcação de consultas médicas. In: **Anais da XXI Escola Regional de Computação Bahia, Alagoas e Sergipe**. SBC, 2021. p. 20-27.

SMUTNY, Pavel; SCHREIBEROVA, Petra. Chatbots for learning: A review of educational chatbots for the Facebook Messenger. **Computers & Education**, v. 151, p. 103862, 2020.

STACK OVERFLOW. **2022 Developer Survey.** 2020. Disponível em: https://survey.stackoverflow.co/2022. Acesso em: 28 dez. 2022.

TELEGRAM. **700 Milhões de Usuários e Telegram Premium.** 2022. Disponível em: https://telegram.org/blog/700-million-and-premium?ln=pr-br. Acesso em: 17 dez. 2022.

WHATSAPP. **Dois bilhões de usuários: conectamos o mundo com privacidade.** 2020. Disponível em: https://blog.whatsapp.com/two-billion-users-connecting-the-world-privately. Acesso em: 17 dez. 2022.