UNIVERSIDADE DE TAUBATÉ

DANIELA DE OLIVEIRA ARRUDA
FRANCIANE CRISTINE VITORIANO PENINA

ESTUDO SOBRE CHATBOT E MÉTODOS DE DESENVOLVIMENTO

TAUBATÉ
2022

Daniela de Oliveira Arruda
Franciane Cristine Vitoriano Penina

ESTUDO SOBRE CHATBOT E MÉTODOS DE DESENVOLVIMENTO

Monografia apresentada como requisito parcial para obtenção do certificado de graduação do curso de Engenharia da Computação pela Universidade de Taubaté.

Área: Inteligência Artificial

Orientador: Prof. Dr. Luis Fernando de Almeida

TAUBATÉ
2022

Grupo Especial de Tratamento da Informação - GETI
Sistema Integrado de Bibliotecas – SIBi
Universidade de Taubaté - Unitau

P411e Penina, Franciane Cristine Vitoriano
Estudo sobre Chatbot e métodos de desenvolvimento / Franciane Cristine Vitoriano Penina , Daniela de Oliveira Arruda. -- 2022.
97 f. : il.

Monografia (graduação) – Universidade de Taubaté, Departamento de Informática, 2022.
Orientação: Prof. Dr. Luis Fernando de Almeida, Departamento de Informática.

1. Inteligência artificial. 2. Chatbot. 3. Framework. I. Arruda, Daniela de Oliveira. II. Universidade de Taubaté. Departamento de Informática. Graduação em Engenharia de Computação. III. Titulo.

CDD – 006.3

Ficha catalográfica elaborada **por Shirlei Righeti – CRB-8/6995**

AGRADECIMENTOS

Primeiramente, gostaríamos de agradecer a Deus, que fez com que nossos objetivos fossem atingidos, durante todos os anos de estudo. Agradecer por ter permitido que nós tivéssemos saúde e determinação para não desanimar durante a elaboração deste trabalho.

Aos nossos pais e familiares, que nos incentivaram nos momentos difíceis e compreenderam nossa ausência enquanto nos dedicávamos ao curso.

Aos amigos que sempre estiveram ao nosso lado, pela amizade e pelo apoio demonstrado ao longo de todo período de estudos.

E, por fim, ao professor Dr. Luis Fernando de Almeida, por ter sido nosso orientador e ter desempenhado tal função com dedicação, amizade e paciência pela qual guiou o nosso aprendizado.

“A persistência é o caminho do êxito”.

(Charles Chaplin)

RESUMO

Com a evolução da inteligência artificial e das tecnologias computacionais, tem se tornado cada vez mais comum os chatbots serem utilizados por grandes e até pequenas empresas para realizarem tarefas, exclusivamente humanas, como atendimentos rápidos, que significa uma iteração imediata com o cliente, sem a necessidade de baixar um aplicativo separadamente e sem perder a qualidade. A vantagem do uso destes chatbots facilita a redução de custos e padronizam informações. Fornece ao usuário a probabilidade de obter informações de maneira prática e com qualidade, explanar dúvidas e realizar pedidos, são algumas das aplicações disponíveis no atendimento através do uso de agentes virtuais. Porém, existem muitas ferramentas de construção de chatbot, dificultando, assim, a escolha. O presente trabalho propõe um estudo das tecnologias de construção de chatbot para auxiliar na escolha da melhor plataforma e biblioteca para essa necessidade. São citados alguns dos métodos disponíveis mais acessíveis para este estudo. O trabalho ainda aborda a história da inteligência artificial, explica o que é chatbot e como usar os frameworks escolhidos. Comenta-se comparações realizadas com cada ferramenta, através de parâmetros de avaliação, estudos de casos, tabelas, imagens e resultado final, que é a proposta deste trabalho.

Palavras-chave: Inteligência artificial. Chatbot. Framework. Agentes virtuais.

ABSTRACT

With the evolution of artificial intelligence and computer technologies, it has become increasingly common for chatbots to be used by large even small companies to perform tasks, exclusively human, such as quick service, which means an immediate iteration with the customer, without the need to download an application separately and without losing quality. The advantage of using these chatbots facilitates cost savings and standardizes information. Providing the user with the probability of obtaining information in a practical and quality way, explaining doubts and placing orders, are some of the applications available in the service through the use of virtual agents. However, there are many chatbot building tools, making it difficult to choose. The present work proposes a study of chatbot building technologies to assist in choosing the best framework for your need. Some of the most accessible methods available for this study are cited. The work also addresses the history of artificial intelligence, explains what chatbot is and how to use the chosen frameworks. Comparisons made with each tool are commented, using evaluation parameters, case studies, tables, images and final result, which is the purpose of this work.

Key-words: Artificial intelligence. Chatbot. Framework. Virtual agents.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

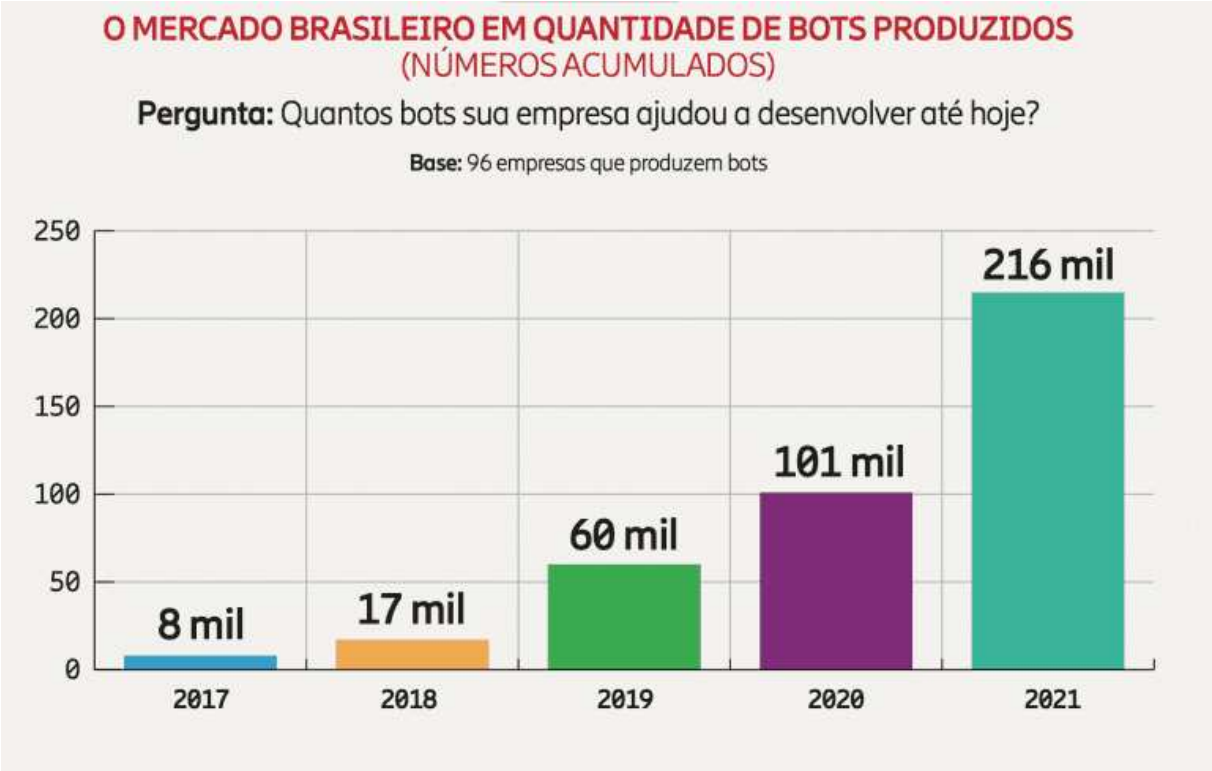
1. INTRODUÇÃO

Chatbot pode ser definido como uma ferramenta usada nos canais de comunicação de empresas, para facilitar a interação com clientes, simulando a interação com atendentes, nas diversas demandas (B2B STACK, 2019).

O conceito da palavra Chatbot é a junção das palavras “chat”, que significa conversa, em inglês e bot que significa robô, também em inglês. Baseado em plataformas de troca de mensagens, esses agentes virtuais têm o objetivo de automatizar os processos interação de empresas com cliente final, podendo ser usados no Facebook, Whatsapp, Telegram e diversos outros serviços relevantes para a empresa.

De acordo com a 5ª edição do Mapa Brasileiro do Ecossistema Brasileiro de Bots, relatório anual produzido por Mobile Time, a partir de dados coletados com 96 empresas que atuam nesse mercado no país, a quantidade de chatbots em atividade no Brasil praticamente dobrou em um ano, passando de 24 mil para 47 mil. Cada um desses bots conversam, em média, com 5,5 mil pessoas diferentes por mês e registram um tráfego mensal de 58 mil mensagens. O volume total de mensagens trocadas pelos robôs em atividade no Brasil, por mês aumentou 27% no mesmo período, subindo de 2,2 bilhões para 2,8 bilhões (MOBILE TIME, 2021). A Figura 1 apresenta o mapeamento do mercado brasileiro quanto à utilização de chatbots.

Figura 1: Mapeamento do uso de chatbots no Brasil.



Fonte: https://www.mobiletime.com.br/noticias/27/08/2021/brasil-dobra-a-quantidade-de-robos-de-conversacao-em-um-ano/ (2021)

A popularização dos chatbots, pode ser justificada pela pandemia e à necessidade de adaptação nos negócios das empresas. O Mobile Time explica que "82% das companhias entrevistadas no relatório declararam que a pandemia do vírus SARS-CoV-2 provocou um aumento na demanda por chatbots em seus negócios. No ano de 2020, 76% disseram o mesmo. Apenas 2% afirmaram que a procura, ao contrário, diminuiu durante a pandemia – mesma proporção registrada em 2020 (MOBILE TIME, 2021).

Durante a pandemia, para realizar a maioria das demandas, muitas empresas se adaptaram ao uso dos chatbots para atender os clientes online para realizar compras.

De acordo com a Super Bots Experience, em 2020 as principais mudanças no Brasil que foram impactantes nos negócios são (COMUNIQUE-SE PORTAL, 2021):

- O número de robôs para chat passou de 60mil para 101mil até setembro de 2020, aumentando assim 68%.
- Destes 101 mil robôs, estavam em atividades 24 mil, mostrando um fluxo de conversação com humanos entre 2,2 bilhões por mês. Com isso, esses dados mostram que os atendimentos por chats aumentaram 120% no Brasil.

1.1. TRABALHOS CORRELATOS

A importância do chatbots pode ser evidenciada em face de muitas pesquisas nesta área.

Santos (2018) realizou um trabalho de implementação de um chatbot para acompanhamento de pessoas com doenças crônicas e não transmissíveis. O *framework* escolhido para o desenvolvimento foi o Chatfuel, na plataforma Facebook.

Hlupic, Irani e Paul (1999) relatam um aumento no uso de ferramentas de simulação e *softwares* no mercado, por conta disso é evidente a importância de uma abordagem adequada para avaliação e seleção desses *softwares*. O artigo apresenta uma estrutura abrangente que pode ser usada para avaliação desses pacotes, aonde os critérios de avaliação são agrupados de acordo com sua natureza e podem ser de utilidade prática para qualquer pessoa envolvida na avaliação do *software*.

Naviani (2020) publicou um artigo mostrando a construção de um chatbot usando o ChatterBot e explicando suas funções. Ele também explica sobre bibliotecas disponíveis para construções de bots, avalia os pontos fortes e fracos de cada uma delas, passa pela construção do bot desde a instalação dos pacotes necessários, explica como funciona a criação da instancia do chatbot, seu treinamento, a criar uma base de dados para esse treinamento, chegando à execução do chatbot de forma simples, mas funcional.

1.2. OBJETIVOS

1.2.1. **Objetivo Geral**

Este trabalho tem como objetivo geral realizar um estudo com os *frameworks* escolhidos, para entender as técnicas envolvidas no desenvolvimento dos agentes de conversação e avaliar cada atividade realizada através de parâmetros pré-selecionados.

1.2.2. **Objetivos Específicos**

Para o desenvolvimento deste estudo, foram necessários os seguintes passos:

- Conhecer e estudar sobre chats.
- Fazer um levantamento de critérios para avaliar os chatbots, posteriormente criados.
- Pesquisar e conhecer frameworks e bibliotecas de criação de chatbots.
- Instituir-se sobre as redes sociais utilizadas e suas ferramentas.
- Implementar chats funcionais.
- Testar as ferramentas através dos critérios/parâmetros anteriormente selecionados.
- Avalizar os resultados obtidos nos testes.

1.3. ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado em seis capítulos: o presente capítulo apresenta a problemática abordada neste trabalho; o Capítulo 2 explica sobre chats e bots, bem como o seu uso nos dias atuais, demonstra e explica os mecanismos de criação que estão por trás do funcionamento de cada framework e biblioteca escolhida; o capítulo 3 expõe os critérios de avaliação, separados em 10 grupos, de acordo com a necessidade de cada ferramenta utilizada; no capítulo 4, foram apresentados os estudos de caso e os resultados obtidos a partir dos testes e simulações realizadas durante o desenvolvimento dos chatbots; o Capítulo 6 finaliza com as considerações finais sobre a pesquisa desenvolvida.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta-se dividido para entender melhor a explicação de cada tema abordado, iniciando com a teoria sobre o que são os chats, em seguida sobre os bots e métodos utilizados para o desenvolvimento do deste estudo. Por fim, mostra detalhes e imagens de cada framework estudada.

2.1. O QUE É UM CHAT

A palavra CHAT é originária da língua inglesa e pode ser traduzida como bate-papo ou conversa. Esse recurso nasceu na Finlândia com o objetivo e ser um meio de comunicação instantâneo na internet. Desde sua criação, é uma ferramenta muito utilizada para assuntos pessoas e profissionais, por empresas.

O primeiro chat utilizado no mundo, foi o IRC. Jarkko Oikarine, em agosto de 1988, criou o primeiro bate-papo chamado INTERNET RELAY CHAT, que significa “bate-papo de internet”. O objetivo deste chat era discutir sobre computadores, mas acabou virando febre entre os usuários e tendo a finalidade de trocar arquivos e informações (RDD, 2014).

Atualmente, o serviço de mensagens instantâneas mais usado no Brasil é o WhatsApp, que cota conta com mais de 460 milhões de usuários. Para as empresas, hoje o chat é o meio de comunicação mais viável e rápido, pois traz agilidade e praticidade aos funcionários e clientes.

2.2. O QUE É UM BOT

Os bots fornecem uma experiência que parece menos como usar um computador e mais como lidar com uma pessoa, ou pelo menos com um robô inteligente. Eles podem ser usados para deslocar tarefas simples e repetitivas, como fazer uma reserva de jantar ou coletar informações de perfil, para sistemas

automatizados que podem não exigir mais intervenção humana direta (MICROSOFT, 2022). Os usuários conversam com um bot usando texto, cartões interativos e fala. Uma interação de bot pode ser uma pergunta e resposta rápidas ou pode ser uma conversa sofisticada que fornece acesso inteligente aos serviços.

Um bot pode ser pensado como um aplicativo Web que tem uma interface de conversa. Um usuário se conecta a um bot por meio de um canal como Facebook, Slack ou Microsoft Teams.

Os bots são muito parecidos com aplicativos Web modernos, vivendo na Internet e usando APIs para enviar e receber mensagens. O que está em um robô varia muito dependendo do tipo de bot. O software de bot moderno depende de uma pilha de tecnologias e ferramentas para oferecer experiências cada vez mais complexas em uma ampla variedade de plataformas. No entanto, um robô simples pode apenas receber uma mensagem e ecoá-la de volta para o usuário com muito pouco código envolvido.

Os bots podem fazer o mesmo que outros tipos de software podem fazer: ler e gravar arquivos, usar bancos de dados e APIs e realizar as tarefas computacionais regulares. O que torna os bots exclusivos é o uso de mecanismos geralmente reservados para comunicação entre humanos. Geralmente ele funciona executando o reconhecimento na entrada do usuário para interpretar o que o usuário está pedindo ou dizendo, seguindo, ele gera respostas para enviar ao usuário e então dependendo de como o bot está configurado e de como ele é registrado com o canal, os usuários podem interagir com o bot por meio de texto ou fala, e a conversa pode incluir imagens e vídeos.

2.3. FRAMEWORKS

Frayad et al (1999) e Johnson & Foote (1988) definem o framework como um conjunto de códigos genéricos que compõe um projeto abstrato para a solução de

problemas. Ele funciona como um tablete ou modelo que oferece elementos estruturais básicos para o desenvolvimento de softwares ou aplicações.

Esta sessão apresenta o processo de criação de criação de chatbots pelos frameworks Chatfuel e ManyChat, explica sobre a aplicação, como eles funcionam e por fim ilustra a simulação de marcação de uma consulta médica com um paciente usuário.

2.3.1. Chatfuel

O Chatfuel é uma ferramenta que permite criar, de modo rápido, um chatbot mais popularmente usando com o Facebook Messenger. A criação do bot de alto desempenho é feita em minutos, não necessita de conhecimento em programação.

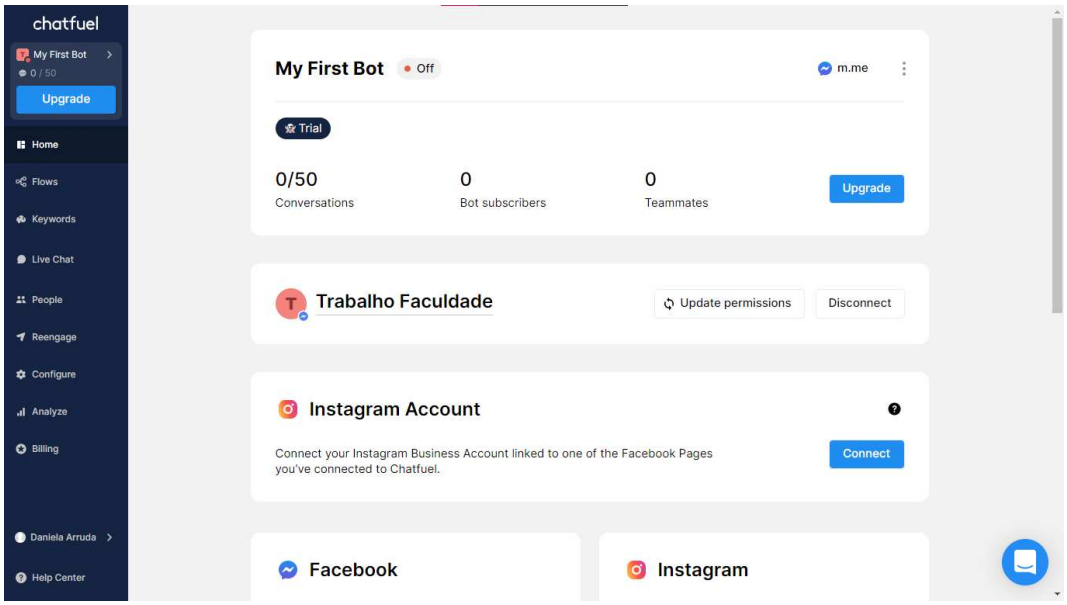
Com o objetivo de ser o local onde se é possível criar robôs de modo acessível, o Chatfuel nasceu em 2015. O primeiro propósito do framework era implementar a plataforma Telegram. Atualmente, após seu grande crescimento, o foco voltou-se para o Facebook Messenger.

Para a criação de chats e utilização da ferramenta, é necessário que o usuário tenha uma conta/ página no Facebook para logar ao site do Chatfuel.

2.3.1.1 Ciclo de construção do bot

Assim que o login foi realizado na plataforma, a tela inicial (home) da plataforma foi apresentada, como na Figura 2. Todos os bots criados são exibidos para gerenciamento do desenvolvedor.

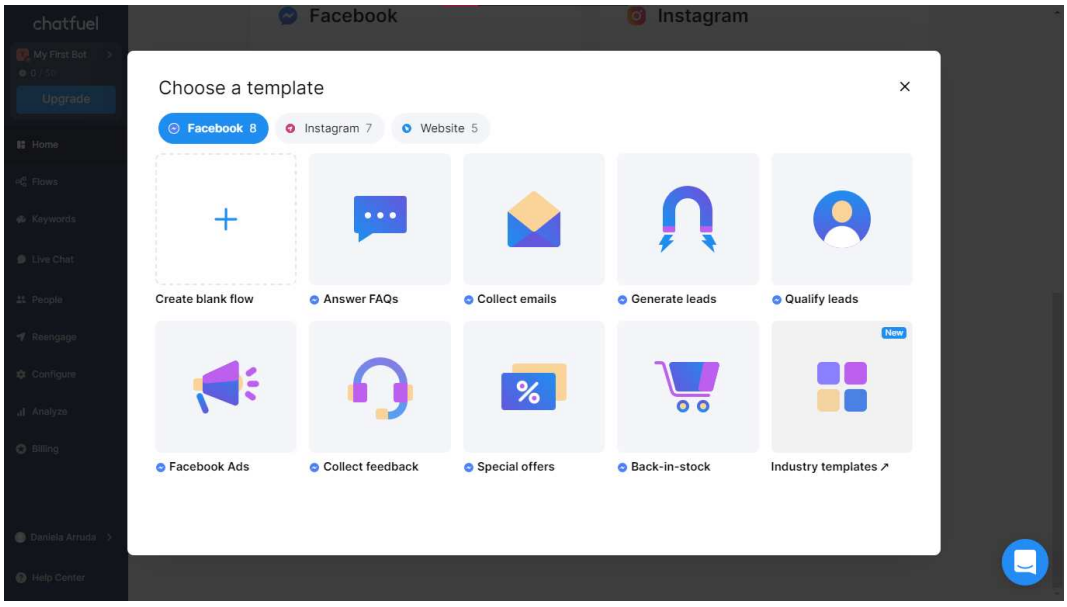
Figura 2: Tela inicial do Chatbot.



Fonte: As autoras.

Selecionando o My First Bot (Figura 2), logo abaixo da tela inicial apareceu as opções de modelos já montados para criação do fluxo de mensagens do chatbot, como na Figura 3.

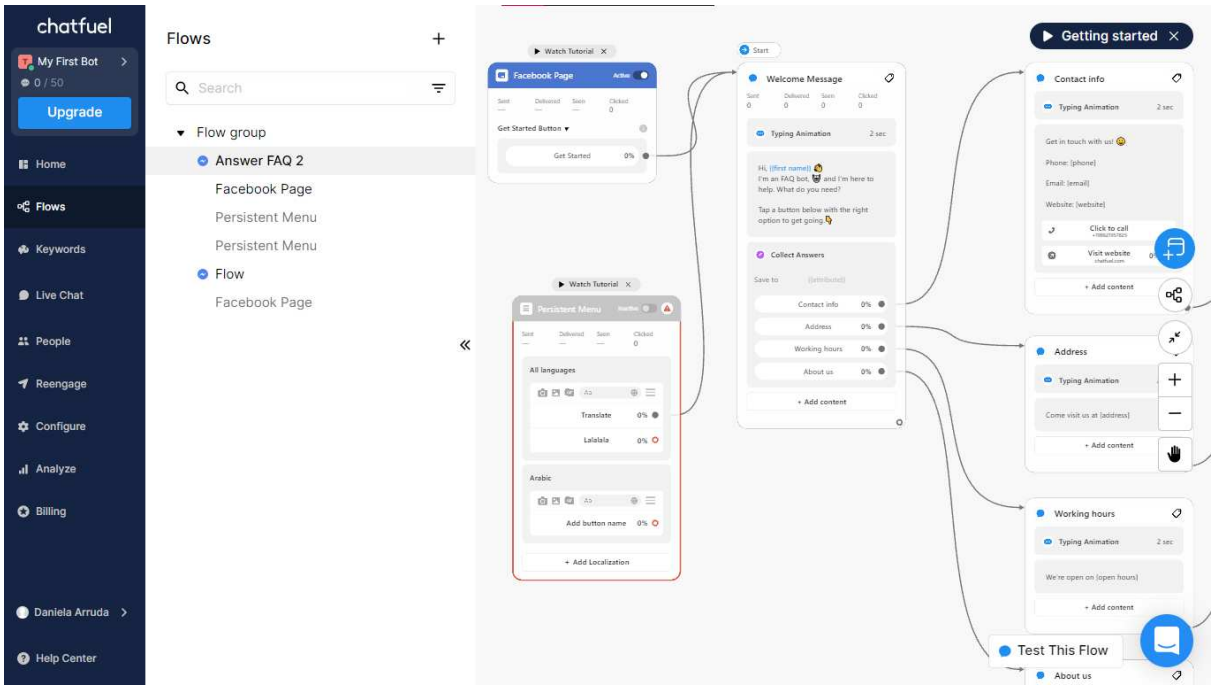
Figura 3: Tela de opções de modelo de chat.



Fonte: As autoras.

Para este trabalho, foi selecionado a opção de criar a sequência de mensagens através de blocos textuais, simulando uma conversa real com respostas rápidas. A Figura 4 exemplifica um modelo apresentado pela ferramenta.

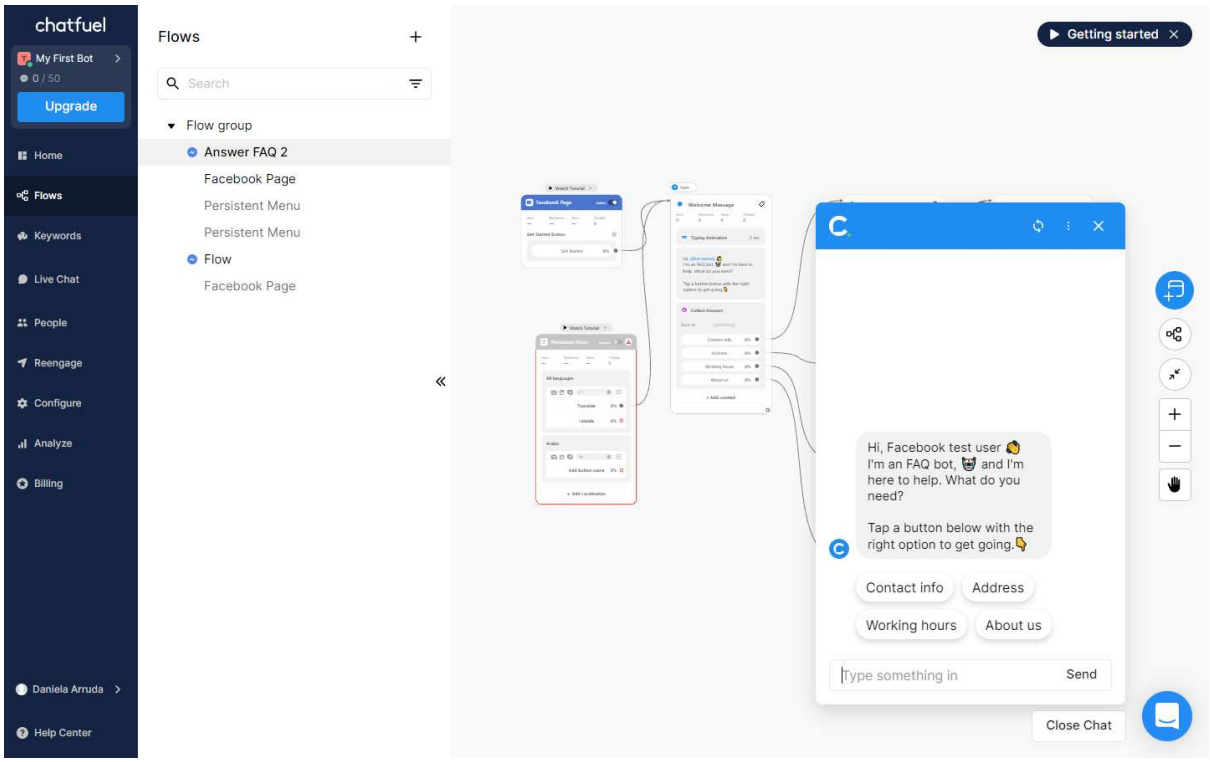
Figura 4: Modelo pronto de Chatbot.



Fonte: As autoras.

O Chatfuel possui um recurso chamado inbox, que simula a conversa entre o bot e o usuário, da mesma maneira como no Facebook Messenger, como apresentado na Figura 5.

Figura 5: Recurso Inbox.

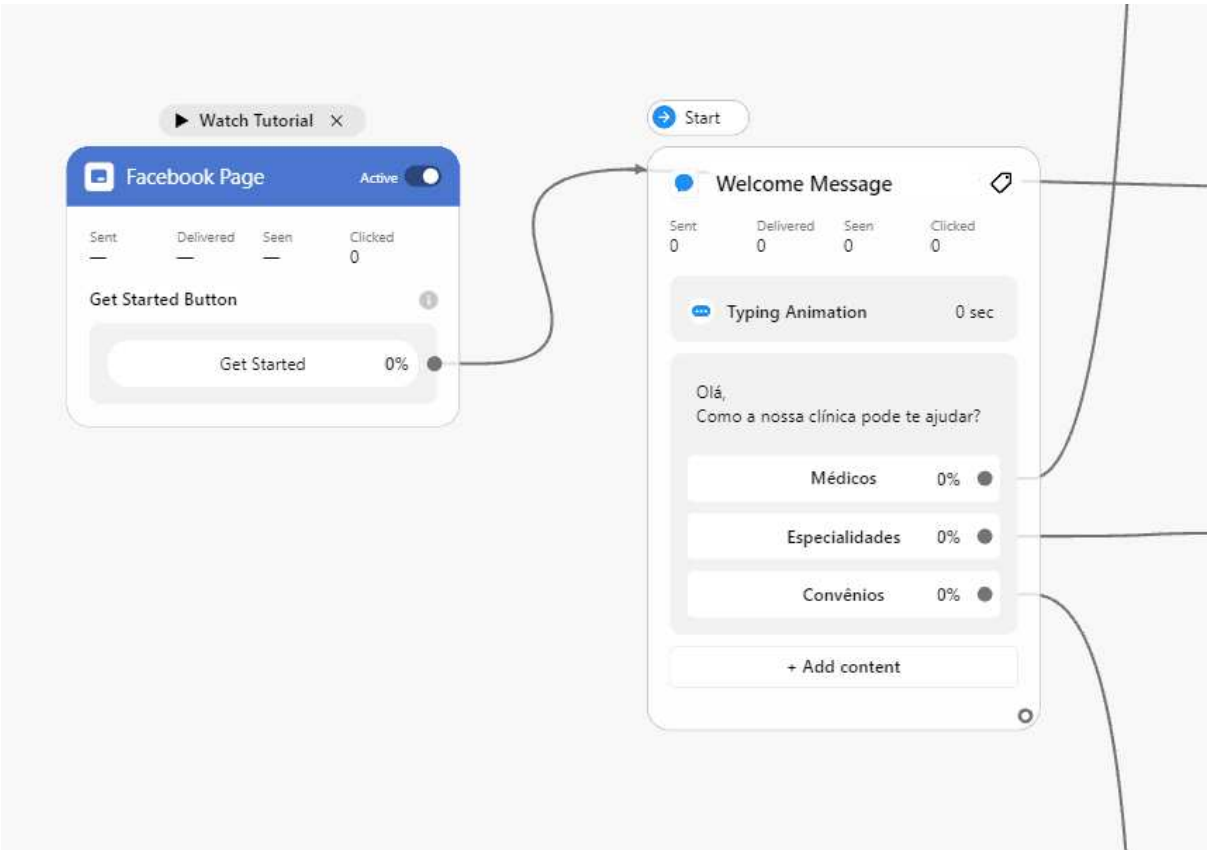


Fonte: As autoras.

Para o estudo, o chat foi separado em três partes: médicos, especialidades e convênios de uma clínica médica. O teste de conversação deste bot foi realizado no Facebook Messenger. Veja a seguir como funciona internamente o fluxograma do chatbot criado.

A Figura 6 representa o primeiro contato do usuário com o bot. Nela ele pode escolher o que deseja consultar primeiramente.

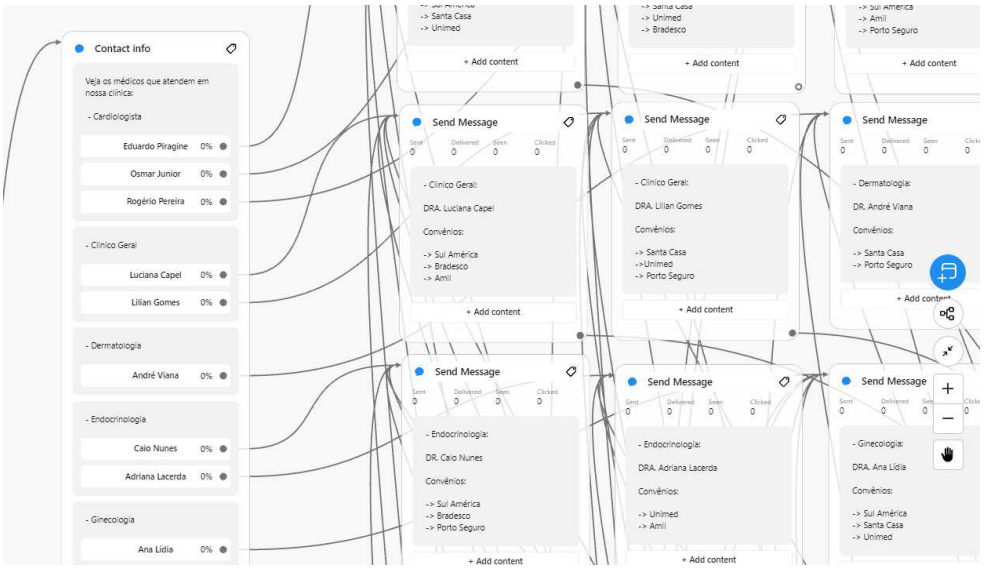
Figura 6: Primeiro contato do usuário com o bot.



Fonte: As autoras.

Quando o usuário selecionou o botão Médico, automaticamente ele foi respondido com uma lista de médicos e após escolher o desejado, apareceu os detalhes, como o nome completo do doutor/doutora, a especialidade e o convênio que o mesmo atende, como na Figura 7.

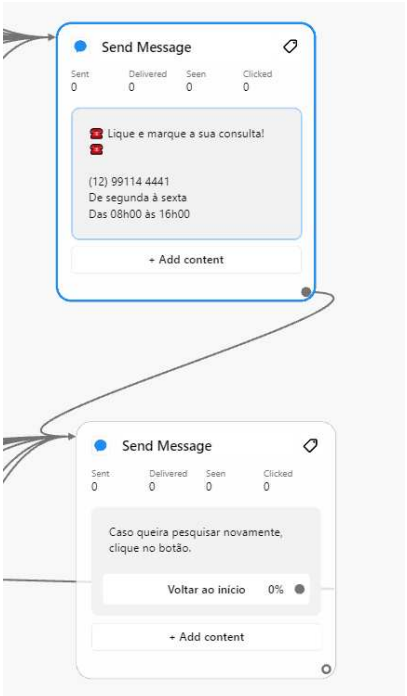
Figura 7: Fluxograma do bot com listas de médicos.



Fonte: As autoras.

Por fim, a informação de contato surge na conversa e o cliente pode voltar ao início para uma nova pesquisa de consulta, como na Figura 8.

Figura 8: Informações de contato com a clínica fictícia

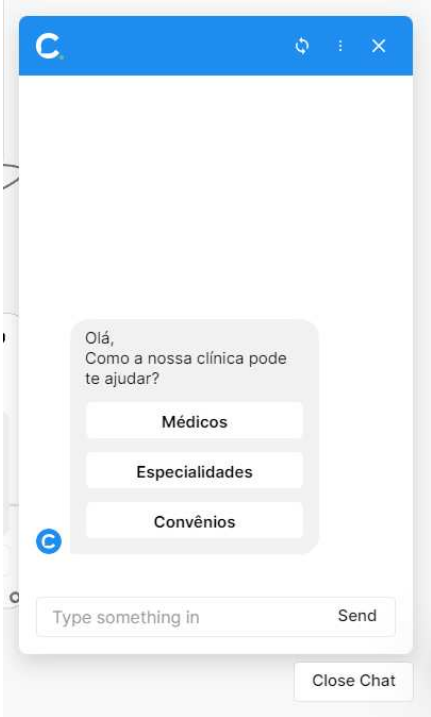


Fonte: As autoras.

As próximas figuras mostram uma breve conversa entre um paciente com o robô, realizando o recurso Inbox do próprio Chatfuel, onde o usuário deseja pesquisar os médicos da clínica.

A Figura 9 apresenta o recurso Inbox juntamente com o primeiro contato do usuário com o bot.

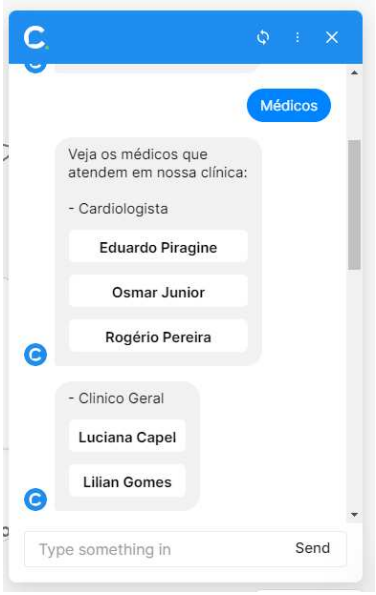
Figura 9: Recurso Inbox com o primeiro contato.



Fonte: As autoras.

A Figura 10 mostra a lista de médicos, após o usuário ter selecionado o botão.

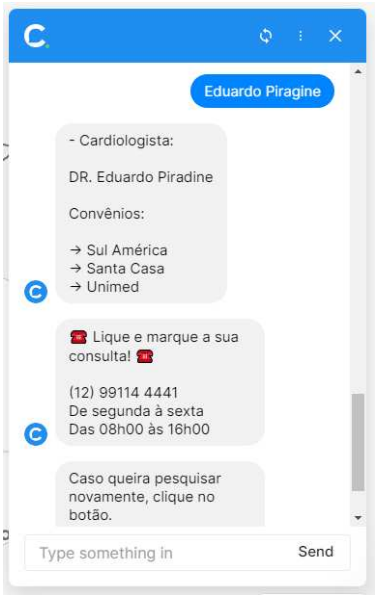
Figura 10: Lista de médicos apresentada pelo chat.



Fonte: As autoras.

Assim que o paciente escolheu o especialista, um conjunto de detalhes foi apresentado na tela juntamente com as informações de contato da clínica, como na Figura 11.

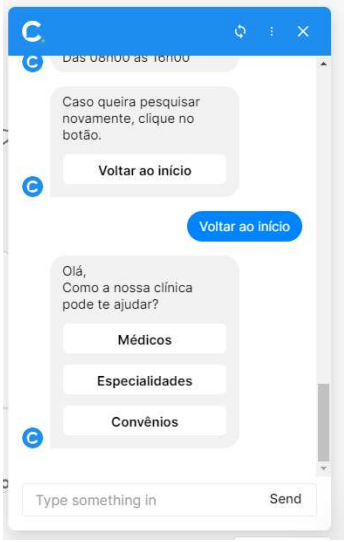
Figura 11: Detalhes do médico selecionado e informações de contato.



Fonte: As autoras.

A seguir, a simulação é realizada quando o paciente decide consultar as especialidades, após clicar em Voltar ao início, assim como na Figura 12.

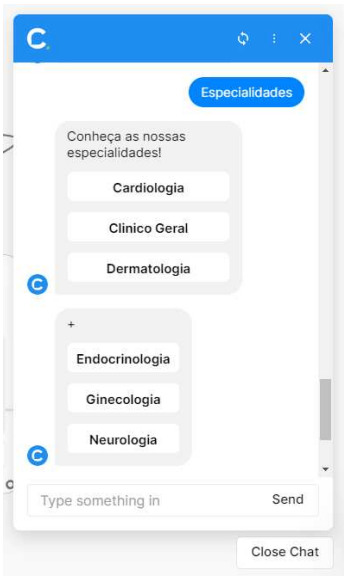
Figura 12: Representação do botão de início e nova pesquisa.



Fonte: As autoras.

A Figura 13 apresenta a lista de especialidades da clínica.

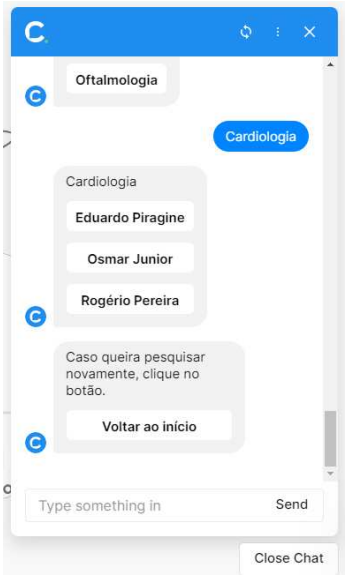
Figura 13: Listas de especialidades da clínica fictícia.



Fonte: As autoras.

Assim que o usuário escolheu a especialidade, uma lista de médicos foi apresentada na tela, como na Figura 14.

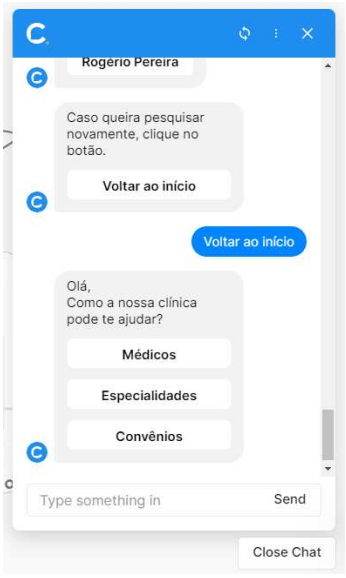
Figura 14: Escolha da especialidade e detalhes dos médicos.



Fonte: As autoras.

Por fim, o próximo teste é realizado no momento em que o usuário escolhe saber quais são os convênios atendidos na clínica, após clicar em voltar ao início, como na Figura 15.

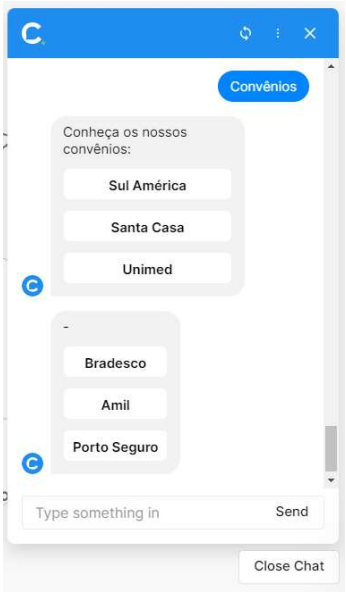
Figura 15: Representação do botão de início e nova pesquisa.



Fonte: As autoras.

A Figura 16 apresenta a lista de convênios da clínica, para o usuário realizasse a pesquisa.

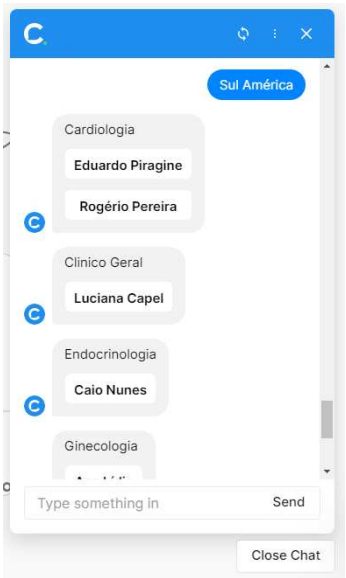
Figura 16: Lista de convênios da clínica fictícia.



Fonte: As autoras.

A Figura 17 mostra a lista de especialidades e médicos atendidos pelo convenio selecionado.

Figura 17: Escolha do convênio e detalhes das especialidades e médicos.

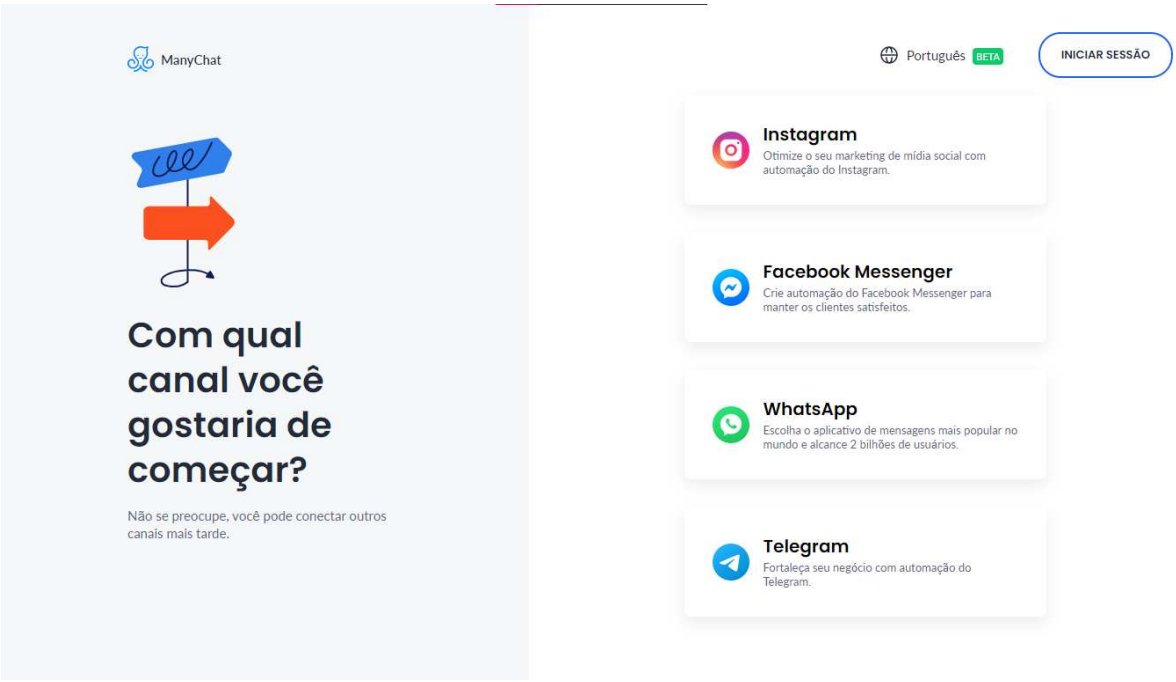


Fonte: As autoras.

2.3.2. **ManyChat**

O ManyChat, como na Figura 18, é uma ferramenta de criação de bots de mensagens de texto para o Facebook Messenger, WhatsApp, Instagram e Telegram, tendo como finalidade automatizar o engajamento de marcas ou empresas, o aumento de vendas e geração de leads.

Figura 18: Tela inicial do ManyChat.



Fonte: As autoras.

A empresa ManyChat nasceu em São Francisco na Califórnia, pelas mãos de Mikael Yang.

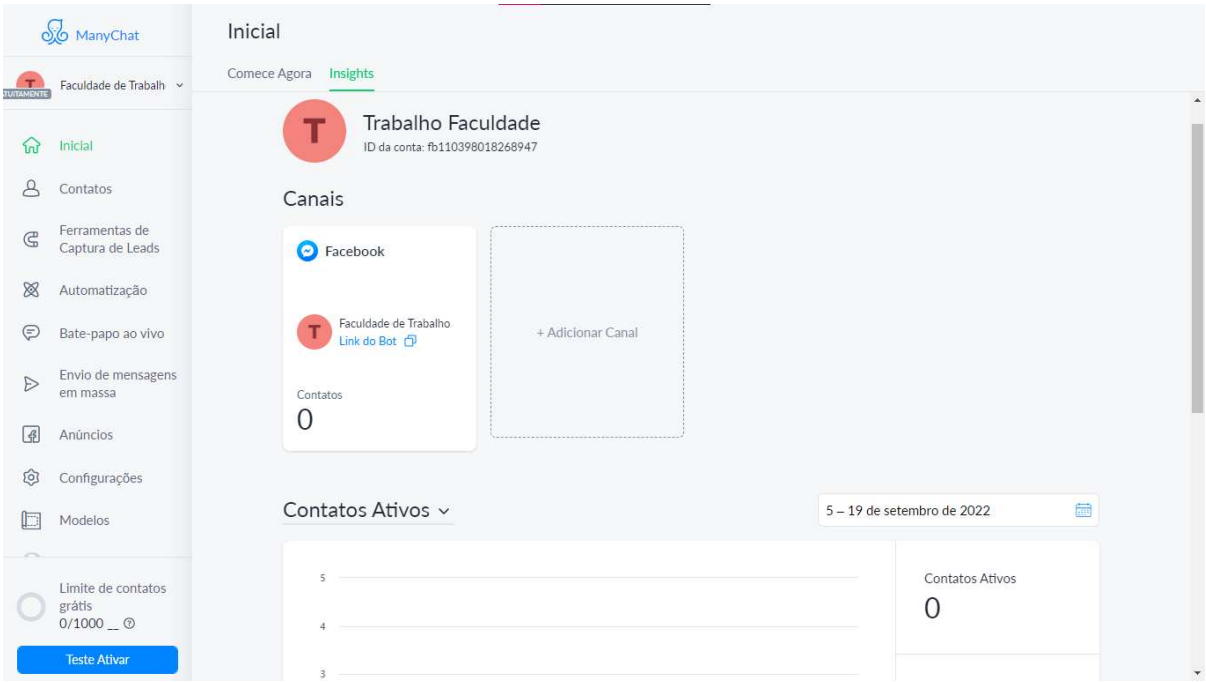
Nesse framework não é necessário a utilização de códigos, com isso é indispensável o conhecimento em programação.

Para este trabalho, foi selecionado o canal Facebook Messenger. Para a criação dos chatbots na plataforma, é necessário que o usuário tenha uma página no Facebook para logar no site do ManyChat.

2.3.2.1 Ciclo de construção do bot

Após a realização do login na plataforma, a tela Dashboard é apresentada, como na Figura 19. A Dashboard é um painel para controle de informações básicas de conta e da lista de usuários que utilizam o bot.

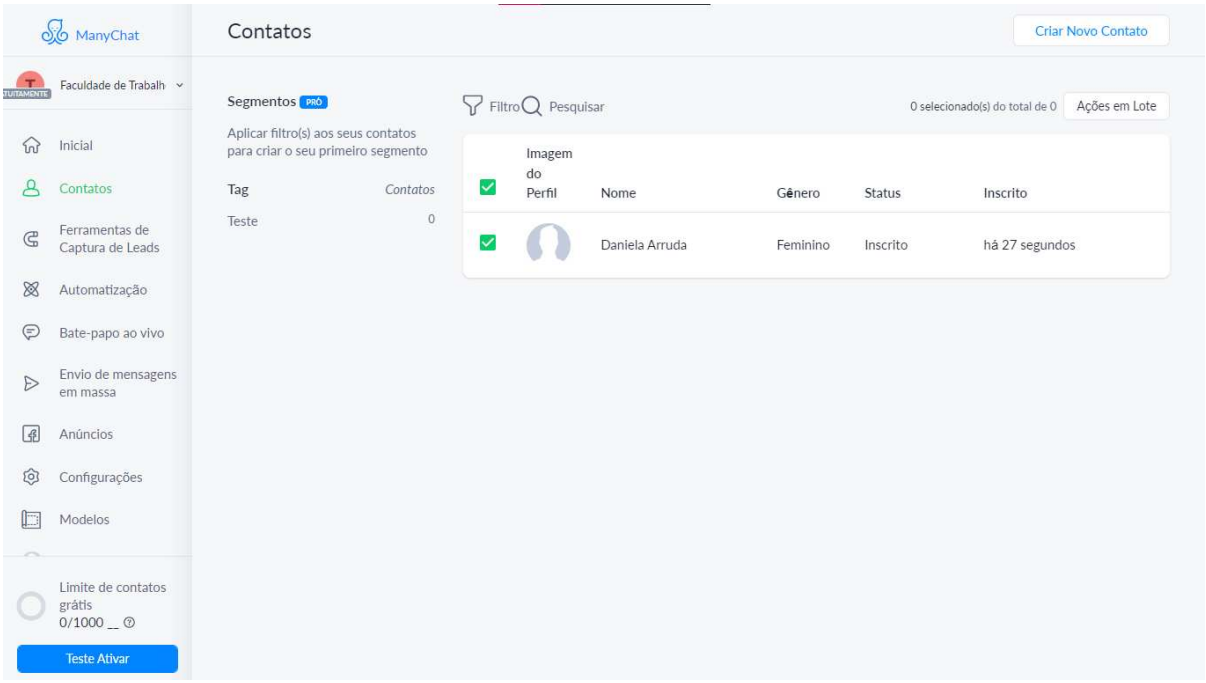
Figura 19: Tela de dashboard do ManyChat.



Fonte: As autoras.

A tela Contatos é utilizada para consultar os contatos do bot dentro da página do Facebook. A contagem só é contabilizada quando o usuário enviar uma mensagem. A Figura 20, mostra o contato de teste que foi criado para a realização deste trabalho.

Figura 20: Tela contatos do ManyChat.

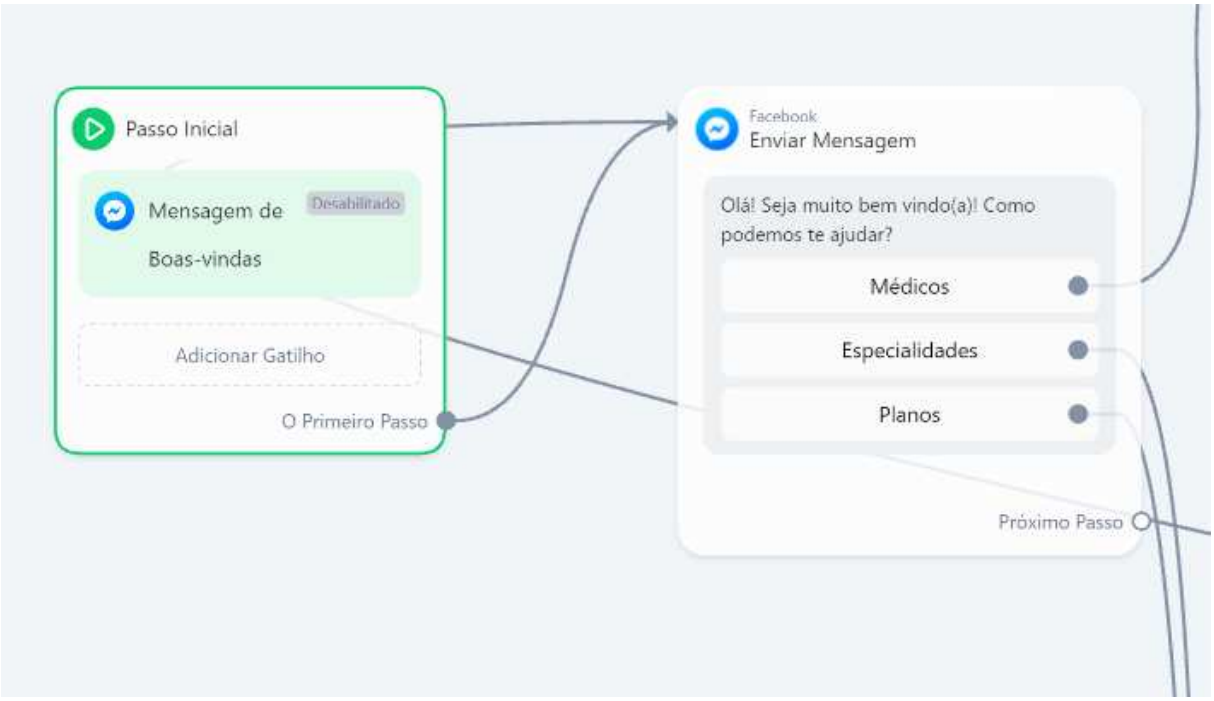


Fonte: As autoras.

Para realizar a análise, todos os bots foram construídos com as mesmas características do anterior, ou seja, os mesmos dados e sequencias. Com isso o chat feito no ManyChat também foi separado em três partes: médicos, especialidades e convênios de uma clínica médica fictícia. O teste de conversação deste bot foi realizado no Facebook Messenger. Veja a seguir como funciona internamente o fluxograma do chatbot criado.

A Figura 21 representa o primeiro contato do usuário com o bot. Nela ele pode-se escolher o que deseja consultar primeiramente.

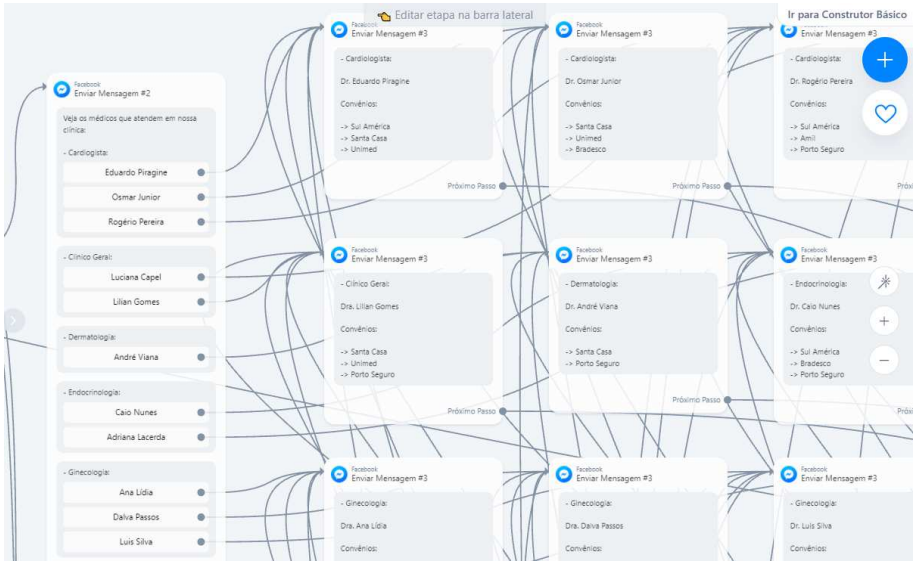
Figura 21: Primeiro contato do usuário com o bot.



Fonte: As autoras.

Quando o usuário selecionou o botão Médico, automaticamente ele foi respondido com uma lista de médicos e após escolher o desejado, apareceu os detalhes, como o nome completo do doutor/doutora, a especialidade e o convênio que o mesmo atende, como na Figura 22.

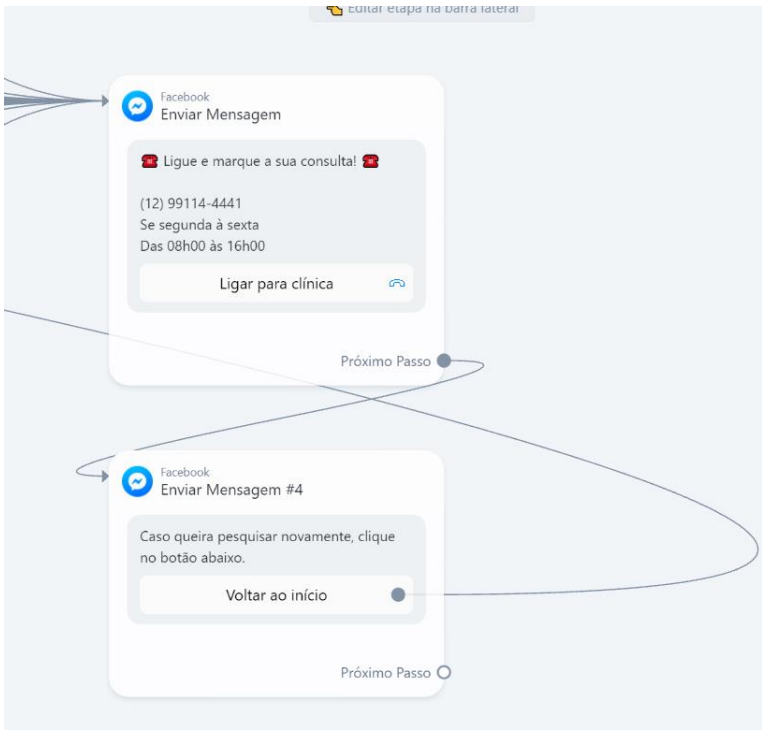
Figura 22: Fluxograma do bot com lista de médicos.



Fonte: As autoras.

Por fim, a informação de contato apareceu na conversa e o cliente pode voltar ao início para uma nova pesquisa de consulta, como na Figura 23.

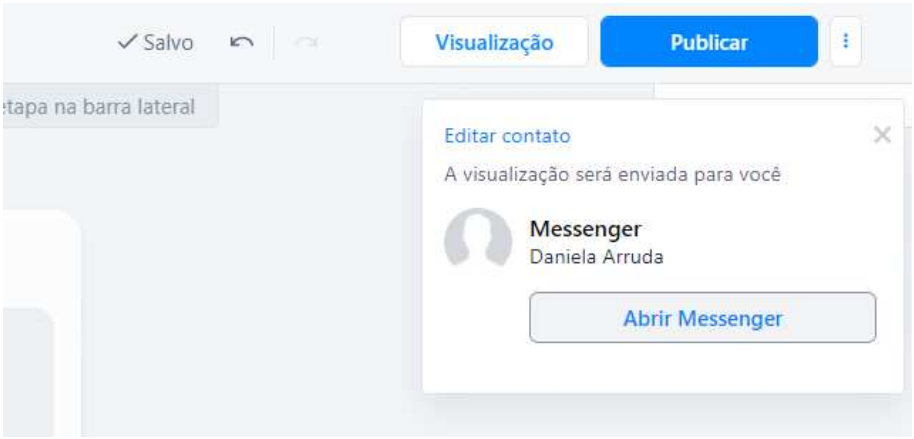
Figura 23: Informações de contato da clínica fictícia.



Fonte: As autoras.

As próximas figuras mostram uma breve conversa entre um paciente com o robô, realizado através do Facebook Messenger, onde o usuário deseja pesquisar os médicos da clínica, como na Figura 24.

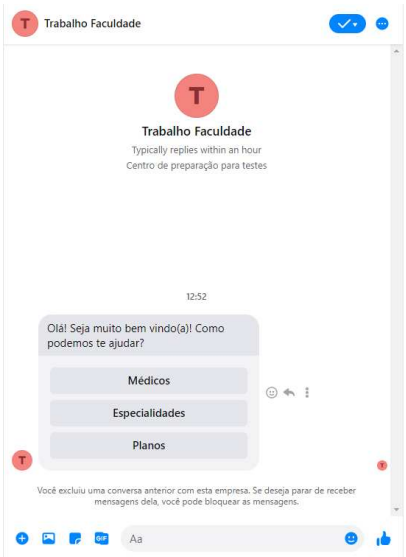
Figura 24: Recurso Facebook Messenger.



Fonte: As autoras.

A Figura 25 apresenta o recurso Facebook Messenger, juntamente com o primeiro contato do usuário com o bot.

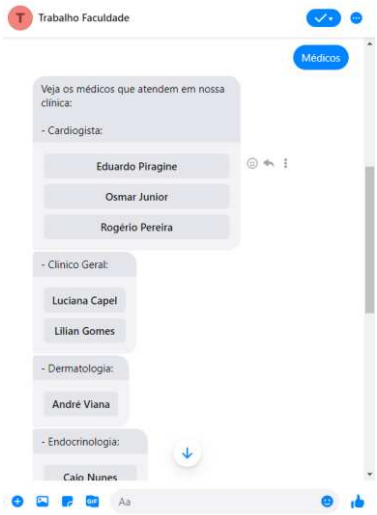
Figura 25: Primeiro contato do usuário com o bot.



Fonte: As autoras.

A Figura 26 mostra a lista de médicos, após o usuário ter selecionado o botão.

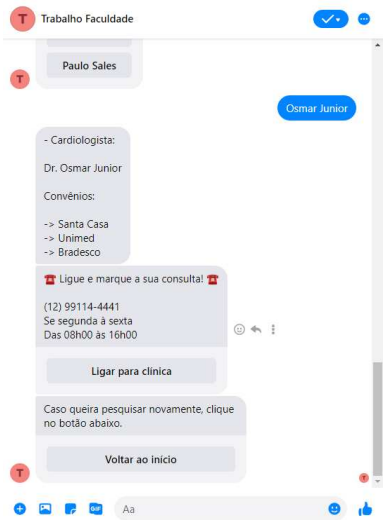
Figura 26: Lista de médicos apresentada pelo bot.



Fonte: As autoras.

Assim que o paciente escolheu o especialista, um conjunto de detalhes foi apresentado na tela juntamente com as informações de contato da clínica, como na Figura 27.

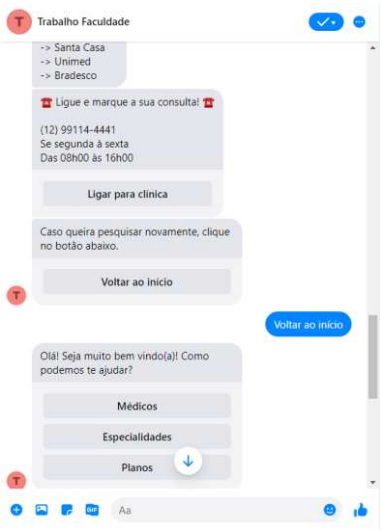
Figura 27: Detalhes do médico selecionado e informações de contato.



Fonte: As autoras.

A seguir, a simulação é realizada quando o paciente decide realizar uma nova pesquisa, consultando as especialidades, após clicar em Voltar ao início, assim como na Figura 28.

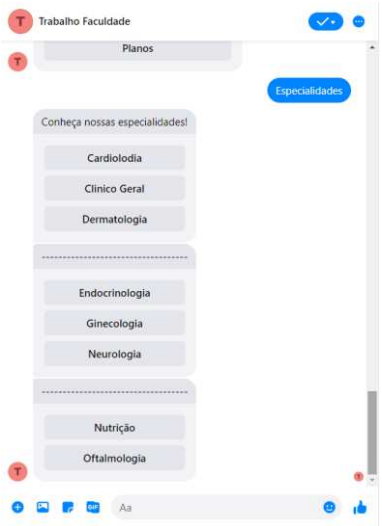
Figura 28: Representação do botão de início e nova pesquisa.



Fonte: As autoras.

A Figura 29 apresenta a lista de especialidades da clínica.

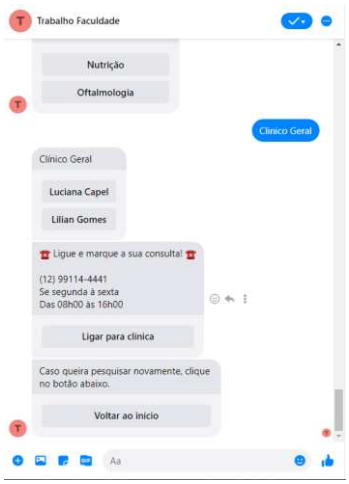
Figura 29: Lista de especialidades da clínica fictícia.



Fonte: As autoras.

Assim que o usuário escolheu a especialidade, uma lista de médicos foi apresentada na tela, como na Figura 30.

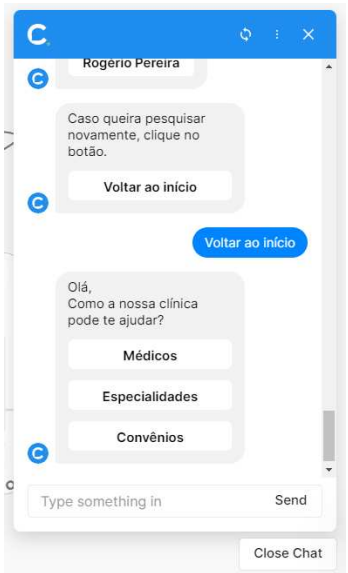
Figura 30: Escolha da especialidade da clínica fictícia.



Fonte: As autoras.

Por fim, o próximo teste é realizado no momento em que o usuário escolhe saber quais são os convênios atendidos na clínica, após clicar em voltar ao início, como na Figura 31.

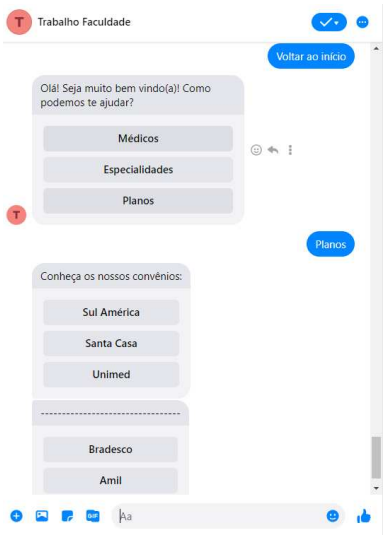
Figura 31: Representação do botão de início e nova pesquisa.



Fonte: As autoras.

A Figura 32 apresenta a lista de convênios da clínica, para o usuário realizasse a pesquisa.

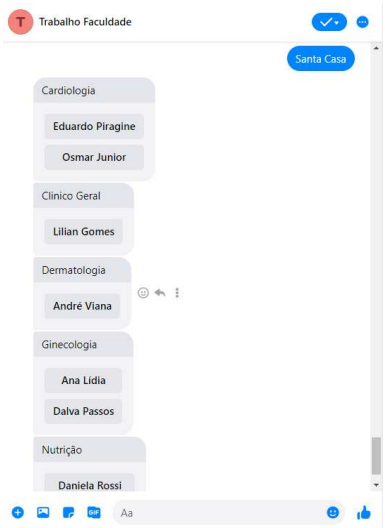
Figura 32: Lista de convênios da clínica fictícia.



Fonte: As autoras.

A Figura 33 mostra a lista de especialidades e médicos atendidos pelo convenio selecionado.

Figura 33: Escolha do convênio e detalhes das especialidades e médicos.



Fonte: As autoras.

2.4. BIBLIOTECAS

Esta sessão apresenta o processo de criação de chatbots pelas bibliotecas ChatterBot e NLTK, explica sobre cada biblioteca, como ela trabalha e por fim ilustra o funcionamento do chatbot através da simulação de marcação de uma consulta médica com um usuário.

2.4.1. **ChatterBot**

ChatterBot é uma biblioteca em Python que torna mais fácil a geração de respostas automatizadas para a entrada de um texto ou áudio de um usuário. A biblioteca usa uma seleção de algoritmos de aprendizado de máquina para produzir diferentes tipos de respostas, tornando mais fácil assim para os desenvolvedores criarem bots de bate-papo e automatizar conversas com usuários.

Por conta de seu design independente de idioma, o ChatterBot permite ser treinado em qualquer idioma. Além disso, seu aprendizado de máquina permite que ele mesmo melhore seu próprio conhecimento de possíveis respostas à medida que interage com humanos e outras fontes de dados informativos.

O chatbot começa com uma instância não treinada sem saber como se comunicar, mas cada vez que um usuário insere uma instrução, a biblioteca salva o texto inserido e o texto ao qual a instrução foi respondida. À medida que ele recebe mais entradas, o número de respostas que ele pode responder e a precisão de cada resposta em relação à instrução de entrada aumentam

O programa seleciona a resposta de correspondência mais próxima a declaração conhecida e, em seguida, escolhe uma resposta da seleção de respostas conhecidas para essa declaração.

2.4.1.1 Construção do chatbot

O processo inicial para a criação do chatbot é a instalação das bibliotecas, é necessário instalar a biblioteca chatterbot e chatterbot corpus, como na Figura 34.

Cada corpus é um protótipo de diferentes declarações de entrada e suas respostas, eles são usados para treinar os bots, como na Figura 34.

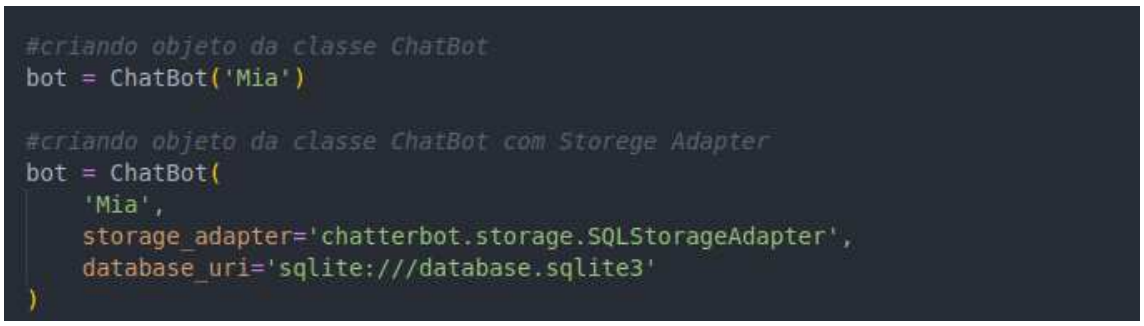
Figura 34: Importando bibliotecas necessárias.

Fonte: As autoras.

A criação da instância do chatbot é a parte de nomear o chatbot e declara-lo, é necessário também estabelecer adaptadores de armazenamento para o bot, esses adaptadores vão permitir o armazenamento para treinamento do mesmo. Nesse chatbot foi usado um banco de dados SQL, com o observado na Figura 35. Há também a possibilidade de posicionar adaptadores lógicos, como o nome indica, o adaptador lógico regula a lógica por trás do chatbot, ou seja, ele escolhe respostas para qualquer entrada fornecida.

O chatterbot permite usar vários adaptadores lógicos, quando mais um for usado, maior será o nível de confiança na resposta, neste chatbot está sendo usado os adaptadores lógicos BestMatch e TimeLogicAdapter, como mostrado na Figura 36.

Figura 35: Criando chatbot e estabelecendo adaptadores de armazenamento.



Fonte: As autoras.

Figura 36: Estabelecendo adaptadores lógicos.

```
#criando objeto da classe ChatBot com Logic Adapter
bot = ChatBot(
    'Mia',
    logic_adapters=[
        'chatterbot.logic.BestMatch',
        'chatterbot.logic.TimeLogicAdapter'],
)
```

Fonte: As autoras.

A etapa final para criar um chatbot pelo chatterbot é seu treinamento através dos módulos disponíveis no chatterbot.

Assim que o bot recebe um conjunto de dados, ele produz as entradas essenciais no gráfico de conhecimento do chatbot para representar a entrada e saída de maneira correta. Para isso acontecer é necessário importar a biblioteca ListTrainer, criar um objeto passando o chatbot e chamar o método train() passando a lista de sentenças, como observado na Figura 37.

Figura 37: Chamando o ListTrainer e declarando algumas sentenças.

```
trainer = ListTrainer(bot)

trainer.train([
'Oi',
'Ola',
'Como posso ajudar?'
])

trainer.train([
'Tudo bem e você?',
'Como vai?',
'Tudo bem?'
'Estou bem',
'Como tem passado?',
'Muito bem',
'Como está seu dia?'
])

trainer.train([
'Opções',
"Especialidades\n" +
"Planos\n"
"Marcar consulta\n"
])
```

Fonte: As autoras.

Por último para testar as habilidades de conversação do chatbot, é chamado o método get_response() junto ao método while que quando recebe um dado de entrada, obtém uma resposta apropriada, pois os dados já estão armazenados no banco de dados, caso receba a declaração “sair” o loop é encerado e interrompe o programa, como mostrado na Figura 38.

Figura 38: Laço while para execução do chatbot.

```
response = bot.get_response('Poderia por favor reformular sua frase,

name=input("Digite o seu nome: ")
print("Bem-vindo(a) a clinica Bem Estar!" +
      "Eu sou Mia a assistente virtual!\n" +
      "Para ver as opções disponiveis digite 'Opções'," 
      "para sair digite 'Sair'.")
while True:
    request=input(name+':')
    if request=='Sair' or request =='sair':
        print('Mia: Agradecemos o contato!')
        break
    else:
        response=bot.get_response(request)
        print('Mia:',response)
```

Fonte: As autoras.

2.4.1.2. Chatbot construído

Na Figura 39 é possível observar o primeiro contato com o chatbot.

Figura 39: Saudações iniciais.

```
Digite o seu nome: Franciane
Bem-vindo(a) a clinica Bem Estar!Eu sou Mia a assistente virtual!
Para ver as opções disponiveis digite 'Opções',para sair digite 'Sair'.
```

Autor: As autoras

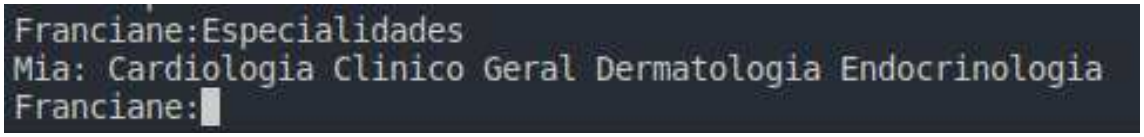
A Figura 40 ilustra o bot listando as opções disponíveis no serviço.

Figura 40: Listagem das opções.

Autor: As autoras.

A Figura 41 ilustra o bot listando as especialidades disponíveis no momento.

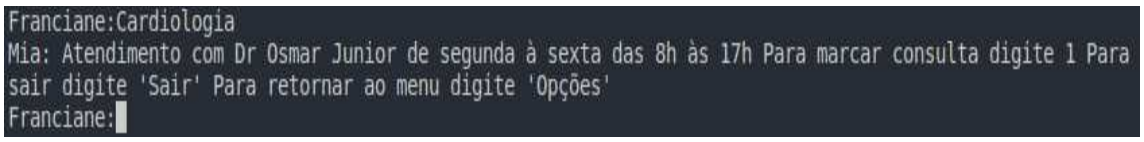
Figura 41: Listando especialidades.



Autor: As autoras.

Ao digitar a especialidade escolhida o chatbot traz as informações referentes àquela escolha, como na Figura 42.

Figura 42: Detalhes da especialidade escolhida.



Autor: As autoras.

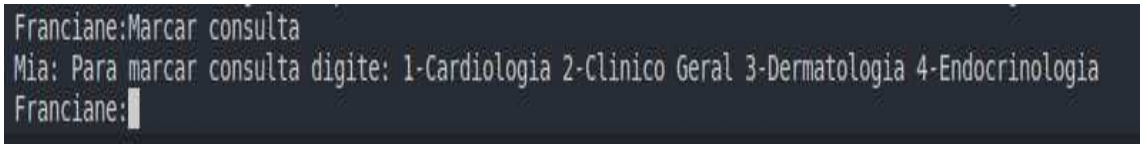
Ao digitar “Planos” é listado todos os planos aceitos nas especialidades, como na Figura 43.

Figura 43: Listando planos.

Autor: As autoras.

Ao digitar “Marcar consulta” é possível ver quais consultas estão disponíveis para marcar, como na Figura 44.

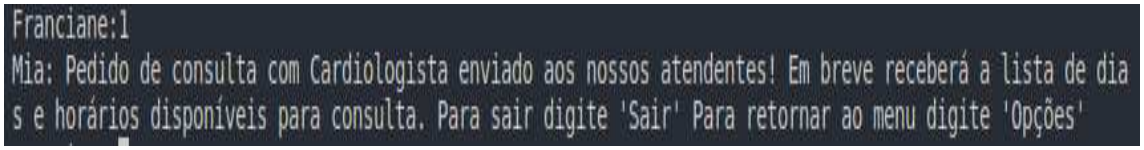
Figura 44: Marcar consulta.



Autor: As autoras.

Ao clicar no número fornecido uma consulta é marcada com a especialidade definida, como mostra a Figura 45.

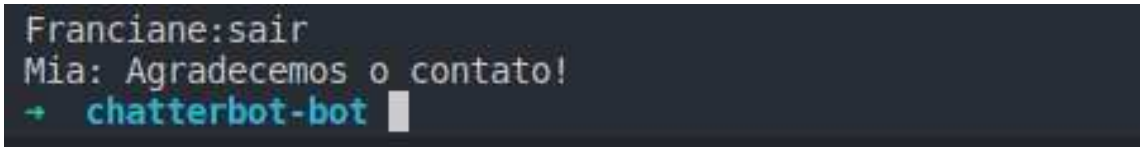
Figura 45: Exemplo de marcação de consulta.



Autor: As autoras.

Ao digitar “sair” o chatbot é encerrado, como mostra a Figura 46.

Figura 46: Encerrando o chatbot.



Autor: As autoras.

2.4.2. **NLTK – Natural Language Toolkit**

NLTK é uma biblioteca em Python que oferece um conjunto de ferramentas de linguagem natural para a construção de programas, dentre essas ferramentas estão bibliotecas de processamento de texto para classificação, tokenização, marcação, análise e raciocínio semântico, além de datasets e alguns algoritmos essenciais para análise e pré-processamento textual.

2.4.2.1. Pré-processamento do texto

O pré-processamento do texto é algo muito importante no processo de criação, pois para a máquina algumas palavras ou estruturas não importam e não fazem diferença, e a biblioteca oferece praticamente todas as ferramentas necessárias para um bom pré-processamento, como por exemplo:

Stopwords: Existem palavras na construção textual chamadas de stopwords, tais palavras, dentro de uma abordagem de NLP, são irrelevantes e sua remoção colaboram com a análise textual. Alguns exemplos de stopwords comuns no português são preposições, artigos, conjunções, entre outras. Quando analisadas pela máquina essas palavras não possuem significância, podendo apenas atrapalhar os processos de aprendizado de máquina. Por esse motivo, geralmente, no pré-processamento é removido essas palavras.

A biblioteca NLTK auxilia nesse processo, pois fornece as stopwords de uma língua, assim é possível removê-las de um texto mantendo somente as que não são.

Lemmatização e Stemmatização: Assim como Stopwords, ter verbos conjulgados em um texto não faz diferença quando a máquina vai processá-lo. Por isso, existem duas ferramentas chamadas Lemmatização e Stemmatização. Ambas fazem a mesma coisa: Quando passado um texto como argumento, elas reduzem todas as formas conjugadas à sua raiz. A única diferença, entretanto, é que a função que lemmatiza o texto a sua raiz, enquanto a função que stemmatiza apenas “corta” as palavras ao meio usando a raiz como base, o que pode gerar palavras que não existem.

Tokenização: A tokenização é transformar elementos em tokens, ou seja, strings dentro de uma lista. O *word_tokenize* é a ferramenta responsável por receber o texto como argumento e retornar todas as palavras do texto em forma de tokens.

Já o *sent_tokenize* transforma frases de um texto em tokens, porém é necessário que o texto já esteja com a pontuação correta, pois ele utiliza os pontos finais como parâmetro para cortar o texto.

2.4.2.2. Criação dos dados de treinamento do bot

Os dados de treinamento é o que vai possibilitar o treinamento chatbot no reconhecimento das frases enviadas para ele, geralmente se é criado um arco de palavras para cada frase.

Basicamente cada um desses arcos são uma representação simples de cada palavra em uma frase como um conjunto de palavras.

Um exemplo seria considerar a sequência: “John likes to watch movies. Mary likes movies too”.

Tabela 1 – Dados de treinamento de um bot.

Separado	John	likes	to	Watch	movies	Mary	likes	movies	too
Único	John	likes	to	Watch	movies	Mary	too		
Arco	John 1	likes 2	to 1	Watch 1	movies 2	Mary 1	too 1		

Depois de criar o conjunto de dados de treinamento, é necessário embaralhar os dados e converter as listas em array Numpy para que seja possível utiliza-lo, após isso, usando como base a tabela arco, é necessário separar os recursos e a coluna de destino dos dados de treinamento seguindo o modelo da imagem.

2.4.2.3. Treinando a rede neural

Após criado a base de dados de treinamento, é necessário criar a rede neural para o seu treinamento, a rede será construída usando uma camada de entrada, a forma de entrada será o tamanho do documento, uma camada oculta, uma camada de saída e duas camadas de descarte.

As camadas ocultas são responsáveis por todo o processamento dos dados da entrada e a camada de saída fornece as probabilidades de palavras diferentes nos dados de treinamento, garantindo uma precisão de quase 100% da rede neural.

Como a rede é treinada através de um conjunto de dados, para fazer previsões é necessário fazer o seu treinamento também. As previsões são necessárias pois ela retorna qual é o resultado mais provável de ser o que o usuário está esperando. Isso quer dizer que o modelo seleciona o índice de maior probabilidade e encontra a tag e as respostas desse índice especifico.

2.4.2.4. Construção do chatbot

Ao iniciar a criação do chatbot é necessário importar algumas bibliotecas e criar uma lista de palavras, classes e documentos, inicialmente vazias, como mostrado na Figura 47.

Figura 47: Importando as bibliotecas necessárias e criando listas.

```
import warnings
warnings.filterwarnings("ignore")
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
import json
import pickle

import numpy as np
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Activation, Dropout
from tensorflow.python.keras.optimizers import gradient_descent_v2
import random

from keras.models import load_model

#importando o bot de treinamento do arquivo de pre-processamento

words = []
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open("Train_Bot.json").read()
intents = json.loads(data_file)
```

Autor: As autoras.

O pré-processamento de dados se refere a manipulação ou descarte dos dados antes de serem usados para garantir ou aprimorar o desempenho e é uma etapa importante no processo de mineração dos dados. No projeto foi usado Corpus Data que contém valores JSON aninhados e com ele é possível atualizar as listas vazias

existentes de palavras, documentos e classes, já o tokenize é usado para dividir um texto ou frase em palavras, é possível observar isso na Figura 48 e na Figura 49.

Figura 48: Pré-processamento dos dados.

```
#pre-processamento do arquivo json
#tokenization

for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokeniza cada palavra
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #adiciona documentos ao corpo
        documents.append((w, intent['tag']))

        #adiciona a lista de classes
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

Autor: As autoras.

Figura 49: Pré-processamento do texto.

```
#lemmatize, deixando todas as palavras em letra minuscula e removendo duplicações
lemmatizer = WordNetLemmatizer()
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))

#ordenando as classes
classes = sorted(list(set(classes)))

#documents = combinação entre os patterns e intents
print(len(documents), "documents")

#classes = intents
print(len(classes), "classes", classes)

#words = todas as palavras
print(len(words), "unique lemmatized words", words)

#criando um arquivo pickle para armazenar os objetos python que usaremos durante a previsão
pickle.dump(words, open("words.pkl","wb"))
pickle.dump(classes, open("classes.pkl","wb"))
```

Autor: As autoras.

Nessa parte é criado o “saco de palavras” (bag of words) para cada frase. Um saco de palavras é uma representação simples de cada texto em uma frase.

Depois de criar os dados de treinamento é necessário embaralhar os dados e converter as listas em uma array NumPy para que seja possível usá-lo no modelo, é possível visualizar isso na Figura 50.

Figura 50: Criando treinamento dos dados.

```
#criando o treinamento dos dados
training = []

#criando array vazio para a saída
output_empty = [0] * len(classes)


#conjunto de treinamento, bag of words para cada frase
for doc in documents:
    #iniciando bag of words
    bag = []
    #lista de palavras tokenizadas para o padrão
    pattern_words = doc[0]

    #lemmatize cada palavra - cria palavra base, na tentativa de representar palavras relacionadas
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]

    #cria nossa matriz de bag of words com 1, se a correspondência de palavras for encontrada no padrão atual
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
    #a saída é um '0' para cada tag e '1' para a tag atual (para cada padrão)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    training.append([bag, output_row])

#embaralhar recursos e converte-los em arrays numpy
random.shuffle(training)
training = np.array(training)

#cria listas de treinamento e teste
train_x = list(training[:,0])
train_y = list(training[:,1])

print("Training data created")
```

Autor: As autoras.

Nesta etapa, criaremos um modelo de rede neural sequencial simples usando uma camada de entrada (a forma de entrada será o comprimento do documento), uma camada oculta, uma camada de saída e duas camadas de eliminação, como na Figura 51.

Figura 51: Criando um modelo de rede neural.

```
#Cria modelo NN para prever as respostas
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

#Compilar o modelo.
sgd = gradient_descent_v2.SGD(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

#ajustando e salvando o modelo
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbot.h5', hist) # we will pickle this model to use in the future
print("\n")
print("*"*50)
print("\nModel Created Successfully!")

#carrega o arquivo de modelo salvo
model = load_model('chatbot.h5')
intents = json.loads(open("Train_Bot.json").read())
words = pickle.load(open('words.pk1','rb'))
classes = pickle.load(open('classes.pk1','rb'))
```

Autor: As autoras.

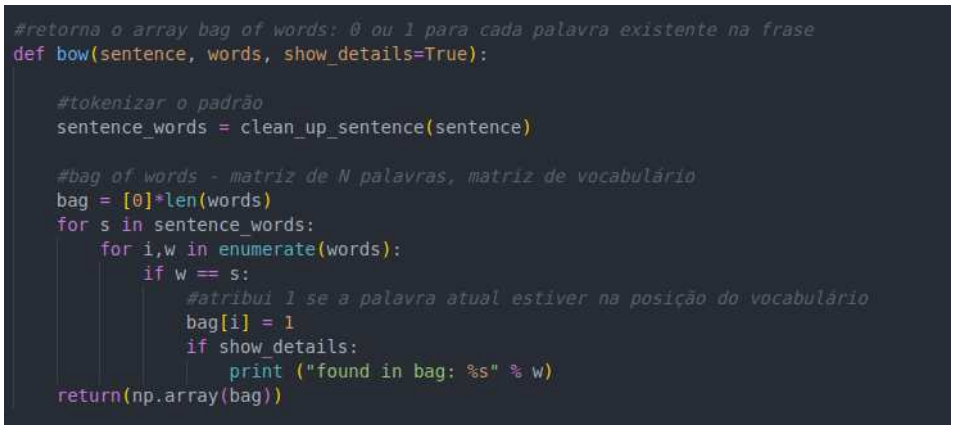
A sequência de figuras abaixo ilustra a criação de funções básicas para o bom funcionamento do chatbot, a Figura 52 detalha a criação da função de limpar a frase e separá-las. A Figura 53 mostra a criação dos “sacos de palavras” utilizando a função de limpar sentença. A Figura 54 mostra a criação da função que prevê a classe aonde pode estar presente a sentença dita pelo usuário. A Figura 55 cria a função que busca a resposta adequada a sentença do usuário na classe prevista e a retorna na tela. A Figura 56 mostra a função que inicia a interação com o chatbot. A Figura 57 e Figura 58 mostra a utilização da biblioteca Tkinter para construção da interface gráfica do bot.

Figura 52: Função para limpar a frase.

```
def clean_up_sentence(sentence):

    #tokeniza o padrão - divide as palavras em um array
    sentence_words = nltk.word_tokenize(sentence)

    #derivar cada palavra - criar forma abreviada para palavra
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
```

Autor: As autoras.

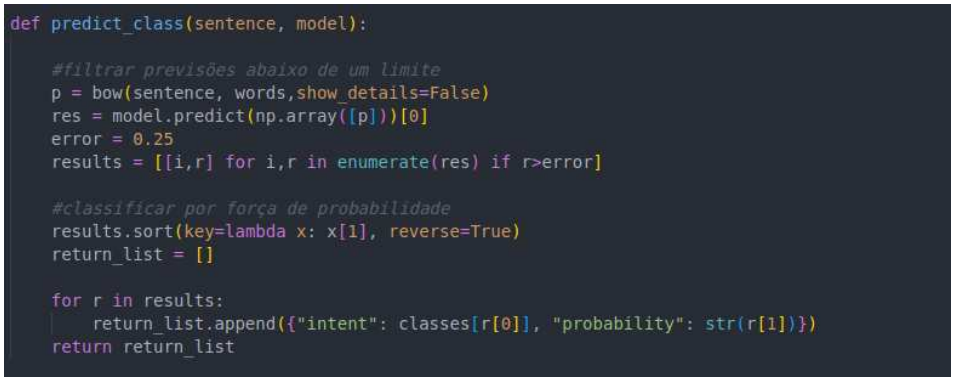
Figura 53: Função para criar a bag of words usando a função de limpar.



```
#retorna o array bag of words: 0 ou 1 para cada palavra existente na frase
def bow(sentence, words, show_details=True):

    #tokenizar o padrão
    sentence_words = clean_up_sentence(sentence)

    #bag of words - matriz de N palavras, matriz de vocabulário
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                #atribui 1 se a palavra atual estiver na posição do vocabulário
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))
```

Autor: As autoras.

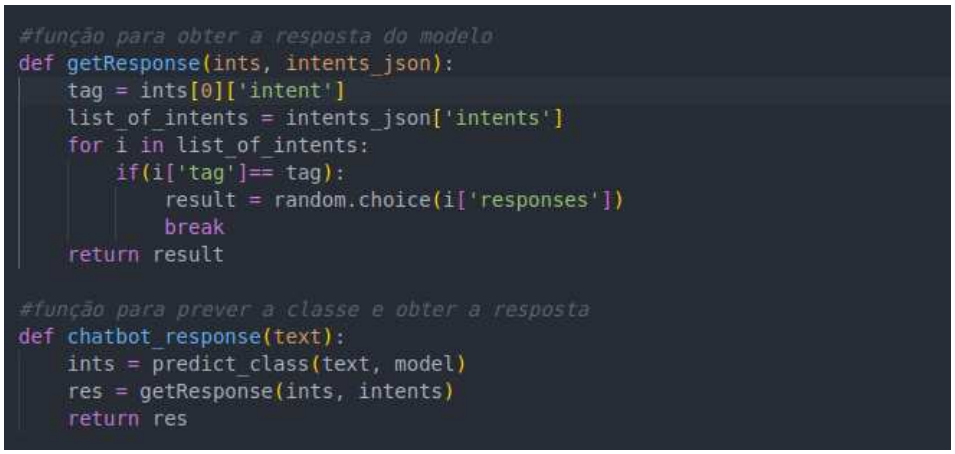
Figura 54: Função para prever a classe de destino.



```
def predict_class(sentence, model):

    #filtrar previsões abaixo de um limite
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    error = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>error]

    #classificar por força de probabilidade
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []

    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list
```

Autor: As autoras.

Figura 55: Funções para obter respostas do modelo.



```
#função para obter a resposta do modelo
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result

#função para prever a classe e obter a resposta
def chatbot_response(text):
    ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res
```

Autor: As autoras.

Figura 56: Função para iniciar o chatbot através do start_chat().

```
#função para prever a classe e obter a resposta
def chatbot_response(text):
    ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res

def start_chat():
    print("Bot: Ola, sou Mia! Sua assistente virtual.\n\n")
    while True:
        inp = str(input()).lower()
        if inp.lower()=="sair":
            break
        if inp.lower()== '' or inp.lower()== '*':
            print('Por favor reescreva seu pedido!')
            print("-"*50)
        else:
            print(f"Bot: {chatbot_response(inp)}"+'\n')
            print("-"*50)

#inicia o chatbot
start_chat()
```

Autor: As autoras.

Figura 57: Utilizando o Tkinter para criar um GUI para o chatbot.

```
import random
import tkinter as tk
from tkinter import *
from main import *

root=tk.Tk()
filename="Chat Bot"
root.title(f"Chat Bot")
root.geometry('500x400')
root.resizable(False, False)
message=tk.StringVar()

chat_win=Frame(root,bd=1,bg='white',width=50,height=8)
chat_win.place(x=6,y=6,height=300,width=488)

textcon=tk.Text(chat_win,bd=1,bg='white',width=50,height=8)
textcon.pack(fill="both",expand=True)

mes_win=Entry(root,width=30,xscrollcommand=True,textvariable=message)
mes_win.place(x=6,y=310,height=60,width=380)
mes_win.focus()

textcon.config(fg='black')
textcon.tag_config('usr',foreground='black')
textcon.insert(END,"Bot: Eu sou Mia! Sua assistente virtual.\n\n")
mssg=mes_win.get()

exit_list = ['sair','tchau','quit','ate mais','sair do chat']

def greet_res(text):
    text=text.lower()
    bot_greet=['Ola','oi','como vai']
    usr_greet=['oi','ola','hello','hola','tudo bem','como vai']
    for word in text.split():
        if word in usr_greet:
            return random.choice(bot_greet)
```

Autor: As autoras.

Figura 58: Função send_msz() envia mensagem ao usuário.

```
def send_msz(event=None):
    usr_input = message.get()
    usr_input = usr_input.lower()
    textcon.insert(END, f'You: {usr_input}'+'\n','usr')
    if usr_input in exit_list:
        textcon.config(fg='black')
        textcon.insert(END,"Bot: Até mais! Foi ótimo falar com você\n")
        return root.destroy()
    else:
        textcon.config(fg='black')
        if greet_res(usr_input) != None:
            lab=f"Bot: {greet_res(usr_input)}"+'\n'
            textcon.insert(END,lab)
            mes_win.delete(0,END)
        else:
            lab = f"Bot: {chatbot_response(usr_input)}"+'\n'
            textcon.insert(END,lab)
            mes_win.delete(0,END)

button_send=Button(root,text='Send',bg='dark green',activebackground='grey',
                   command=send_msz,width=12,height=5,font=('Arial'))
button_send.place(x=376,y=310,height=60,width=110)
root.bind('<Return>', send_msz,button_send)
root.mainloop()
```

Autor: As autoras.

2.4.2.5. Chatbot construído

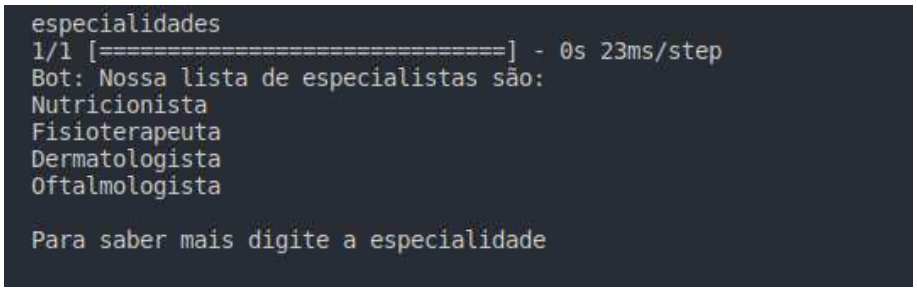
A sequência de figuras abaixo ilustra o chatbot em funcionamento e simula uma interação com o usuário prestando auxilio para marcação de uma consulta.

Figura 59: Saudações iniciais.

```
oi
1/1 [==============================] - 0s 86ms/step
Bot: Olá! Como posso te ajudar?
 Para ver lista de especialistas digite: Especialidades
 Para marcar uma consulta digite: Marcar consulta

--------------------------------------------------
ola
1/1 [==============================] - 0s 21ms/step
Bot: Olá! Como posso te ajudar?
 Para ver lista de especialistas digite: Especialidades
 Para marcar uma consulta digite: Marcar consulta

--------------------------------------------------
eai
1/1 [==============================] - 0s 21ms/step
Bot: Olá! Como posso te ajudar?
 Para ver lista de especialistas digite: Especialidades
 Para marcar uma consulta digite: Marcar consulta
```

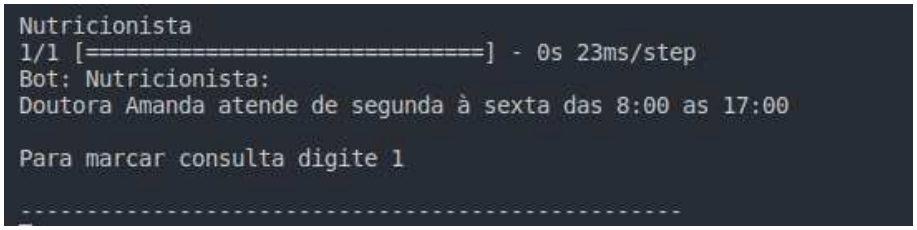
Autor: As autoras.

Figura 60: Funcionalidade especialidade lista todos os médicos disponíveis.



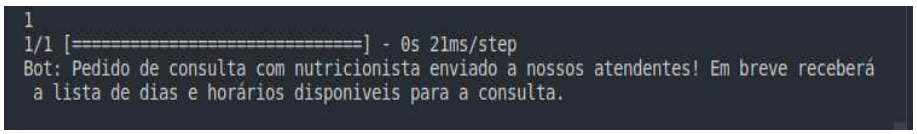
Autor: As autoras.

Figura 61: Ao digitar a especialidade é possível ver mais informações sobre cada uma.



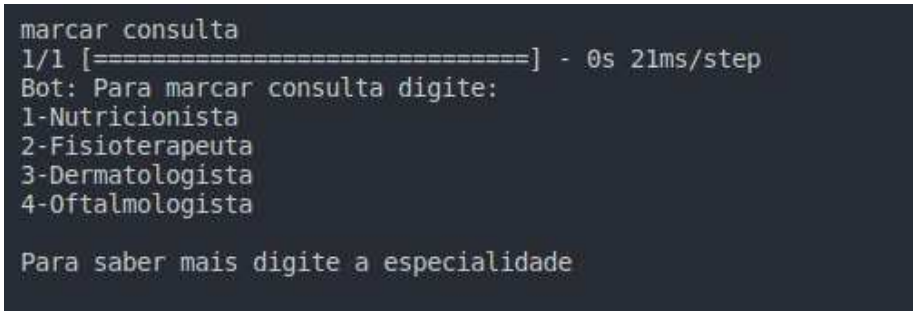
Autor: As autoras.

Figura 62: Ao digitar para marcar a consulta o pedido é enviado aos atendentes.



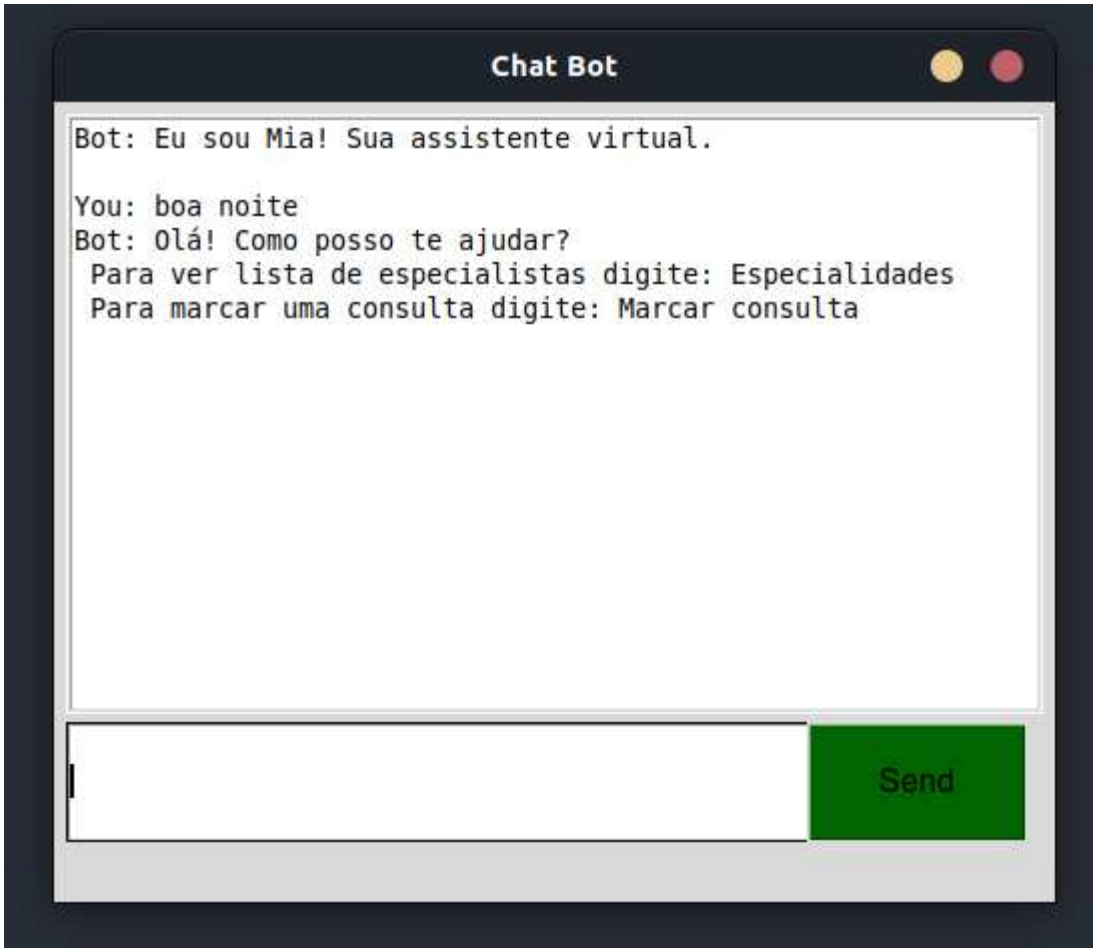
Autor: As autoras.

Figura 63: A opção de marcar consulta leva diretamente as marcações.



Autor: As autoras.

Figura 64: GUI do chatbot.



Autor: As autoras.

3. PARÂMETROS PARA ANÁLISE

3.1. MATERIAIS E MÉTODOS

Os critérios listados nas tabelas abaixo foram baseados no artigo “Evaluation Framework for Simulation Software” (V. Hlupic, Z. Irani and R. J. Paul) do Brunel University, Department of Information Systems and Computing, Uxbridge, UK. Selecionamos os parâmetros que mais nos ajudariam a avaliar os frameworks escolhidos.

Os critérios podem ser aplicados à avaliação de qualquer pacote de framework de uso geral ou específico. Eles são colocados em 10 grupos de critérios de acordo com sua necessidade. Devido à grande quantidade de avaliações, são listados critérios individuais dentro de cada grupo. Também é fornecido um pacote de classificação dos critérios elencados. De acordo com cada tipo de critério, a classificação determina se, por exemplo, um certo recurso existe no pacote, determina a qualidade ou lista os tipos disponíveis de alternativas em determinada característica. A descrição mais completa é fornecida juntamente com as tabelas abaixo.

3.1.1. **Recursos Gerais**

A Tabela 2 mostra os critérios classificados no grupo que descrevem as características gerais do framework. A maioria dos critérios estão relacionados a aspectos de modelagem como, o tipo de lógica necessária para modelagem (se houver), o método de alterar o estado do modelo (baseado no processo, em atividades), etc. Existem também alguns critérios que avaliam o nível de experiência e educação formal em chatbot exigida e examinam o quão fácil é aprender e usar o framework.

Tabela 2 - Critérios para recursos gerais.

Critério	Classificação
Tipo de framework	Framework baseado em dados Framework baseado em dados com capacidade de programação adicional Linguagem de simulação
Objetivo	Propósito geral Estudo de casos Outro objetivo
Abordagem de modelagem	Baseado em processos Baseado em atividades Baseado em eventos
Representatividade dos modelos	Alto Médio Baixo
Versões de framework para diferentes sistemas operacionais	Ubuntu Windows 7 Windows 8 Windows 10 OS
Experiência necessária para uso do framework	Nenhum Alguns Substancial
Educação formal em chatbot necessária para o uso do framework	Nenhum Alguns Substancial
Facilidade de uso	Alto Médio Baixo
Facilidade de aprendizado	Fácil Difícil
Modelos de simulação em tempo real	Possível Não possível
Popularidade e tamanho da comunidade	Grande Média Pequena

Suporte	Sim Não
Disponibilidade de recursos no mercado	Sim Não
Versatilidade	1 a 5

3.1.2. **Aspectos Visuais**

As apresentações dos modelos e animações são características importantes para a simulação. Os critérios incluídos nesse segundo grupo referem-se ao tipo e qualidade e das facilidades fornecidas pelo framework. A Tabela 3 mostra os critérios classificados em aspectos visuais.

Tabela 3 - Critérios para aspectos visuais.

Critério	Classificação
Animação	Possível Não possível
Tipo de animação	Animação completa Semi-animação (estado a estado) Animação simultânea
Desenvolvimento de layout de animação	Separado Simultaneamente ao desenvolvimento do modelo Antes do desenvolvimento do modelo Após o desenvolvimento do modelo Flexível
Facilidade para personalizar a visualização do chatbot	Forneceu Não forneceu
Modo de reprodução	Forneceu Não fornecido

Editor de tela	Forneceu Não fornecido
Facilidade de uso do editor de tela	Fácil Difícil
Indicação do status do elemento	Forneceu Não fornecido

3.1.3. **Aspectos de Codificação**

A Tabela 4 mostra os critérios classificados em aspectos de codificação. A possibilidade de codificação adicional é uma característica importante de um pacote. Esse recurso determina a robustez e a flexibilidade do framework. Os critérios incluídos nesse grupo, determinam se o pacote permite programação, se o acesso ao código é possível, se fornecem as características do código adicionado, etc.

Tabela 4 - Critérios para aspectos de codificação.

Critério	Classificação
Flexibilidade de programação	Forneceu Nenhum programa adicional fornecido
Gerador de programa	Forneceu Não fornecido
Acesso ao código-fonte	Possível Não possível
Legibilidade do código-fonte	Alto Médio Baixo
Legibilidade do código adicionado	Alto Médio Baixo

Precisão do código adicionado	Alto Médio Baixo
Instalação de armazenamento, recuperação e manipulação de dados	Forneceu Não fornecido
Qualidade das instalações de armazenamento, recuperação e manipulação de dados	Alto Médio Baixo
Funções integradas	Forneceu Não fornecido
Funções do usuário	Possível Não possível
Variáveis globais	Forneceu Não fornecido
Nomes das funções, variáveis e atributos	Definido pelo usuário Definido pelo sistema
Texto / Manipulação de código	Possível Não possível
Comprimento das linhas no editor de codificação	Ampla Médio Pequeno
Suporte de conceitos de programação	Forneceu Não fornecido
Qualidade do suporte para conceitos de programação	Alto Médio Baixo
Interface para programas escritos pelo usuário	Forneceu Não fornecido
Conceitos de programação orientada a objetos	Forneceu Não fornecido

3.1.4. **Eficiência**

Os critérios classificados no grupo de eficiência determinam a eficácia e o poder do framework para desenvolvimento de chatbot. A eficácia é demonstrada pela capacidade do framework de modelar uma variedade de sistemas complexos e pelas características que podem economizar tempo necessário para a modelagem e melhorar a qualidade, como a reutilização, confiabilidade, tempo de compilação e execução. A Tabela 5 mostra os critérios desse grupo.

Tabela 5 - Critérios de eficiência.

Critério	Classificação
Robustez	Alto Médio Baixo
Nível de detalhe	Alto Médio Baixo
Número de elementos no chatbot	Grande Médio Pequeno
Reutilização do chatbot	Possível Não possível
Salvar o status do chatbot	Possível Não possível
Economia automática	Possível Não possível
Iteração	Possível Não possível
Adaptabilidade às mudanças do chatbot	Alto Médio Baixo
Tempo de compilação	Longo

	Médio Curto
Tempo de execução do chatbot	Longo Médio Curto
Sensibilidade a maiúsculas e minúsculas	Forneceu Não fornecido
Escala de tempo para a construção do chatbot	Longo Médio Curto
Confiabilidade	Alto Médio Pequeno
Modelos genéricos pré-existentes	Forneceu Não fornecido
Construção automática de modelos	Forneceu Não fornecido
Facilidade de edição de modelo	Fácil Difícil

3.1.4. **Testabilidade**

A Tabela 6 mostra os critérios para testabilidade. Este grupo abrange critérios que examinam quais instalações para a verificação do modelo é fornecido pelo pacote. Essas instalações incluem mensagens de erro, exibições dos valores de lógica dos elementos como funções e variáveis.

Tabela 6 - Critérios para testabilidade.

Critério	Classificação
Verificações lógicas	Forneceu Não fornecido
Mensagens de erro iterativas	Forneceu Não fornecido
Qualidade das mensagens de erro	Alto Médio Baixo
Momento do diagnóstico de erro	Entrada de modelo Compilação Execução do modelo Combinação
Facilidade de depuração	Fácil Difícil
Exibição de atributos	Possível Não possível
Acesso aos atributos	Possível Não possível
Exibição de variáveis	Possível Não possível
Exibição do estado do elemento	Possível Não possível

3.1.5. **Compatibilidade**

Os critérios de compatibilidade avaliam se o pacote pode ter interface para outros aplicativos ou sites, a fim de trocar dados com estes sistemas. A Tabela 7 mostra os critérios inclusos neste grupo. A integração com linguagens de programação não está inclusa neste grupo de critérios, pois está contida em aspectos de codificação.

Tabela 7 - Critérios para compatibilidade aplicativos ou sites.

Critério	Classificação
Integração com Facebook Messenger	Possível Não possível
Integração com WhatsApp	Possível Não possível
Integração com SMS	Possível Não possível
Integração com Sites	Possível Não possível

3.1.6. **Entrada** / **Saída**

A Tabela 8 mostra os critérios para uma classificação dos pacotes em relação à entrada e saída de dados. Os critérios incluídos neste grupo mostram como o usuário pode apresentar os dados para o pacote e o tipo de qualidade do relatório de entrada e saída fornecido pelo framework. Esses critérios avaliam, por exemplo, se o pacote tem uma interface baseada em menu e quão compreensíveis são os relatórios.

Tabela 8 - Critérios para Entrada / Saída.

Critério	Classificação
Interface orientada por menu	Forneceu Não forneceu
Menus suspensos	Forneceu Não forneceu
Tipos de seleção de menu	Por mouse Outro
Botões de seleção	Forneceu Não forneceu

Caixas de diálogos	Forneceu Não forneceu
Manutenção de banco de dados para entrada / saída	Forneceu Não forneceu
Relatórios gerais de produção	Forneceu Não forneceu
Disponibilidade	Forneceu Não forneceu
Relatório	Forneceu Não forneceu

3.1.7. **Suporte**

Os critérios a seguir avaliam a qualidade e o tipo do suporte fornecido pelo framework, o que pode facilitar aprender a usar o pacote. Esses critérios não incluem somente suporte técnico na forma de documentação, mas também incluem uma variedade de serviços fornecidos, que mantenha o usuário informado sobre futuras melhorias. A Tabela 9 mostra os critérios incluídos neste grupo.

Tabela 9 - Critérios para suporte ao usuário.

Critério	Classificação
Documentação (Manuais)	Forneceu Não forneceu
Qualidade da documentação	Alto Médio Baixo
Glossário	Forneceu Não forneceu

Material de informação técnica e promocional (e-mail, quadro de avisos)	Forneceu Não forneceu
Grupo de discussão na internet	Forneceu Não forneceu
Guia do professor para licenças educacionais	Forneceu Não forneceu
Tutorial	Forneceu Não forneceu
Curso de treinamento (básico, avançado)	Forneceu Não forneceu
Curso de treinamento sob medida	Forneceu Não forneceu
Duração dos cursos de treinamento	Longo Médio Curto
Help-line (Suporte por telefone)	Forneceu Não forneceu
Manutenção do pacote	Forneceu Não forneceu
Consultoria	Forneceu Não forneceu

3.1.8. **Recursos Técnicos e Financeiros**

A Tabela 10 mostra os critérios que examinam as características do pacote em relação ao seu custo e características técnicas. Alguns dos critérios abordados aqui são: quão caro é comprar um determinado pacote, para instalá-lo e mantê-lo, etc.

Tabela 10 - Critérios para recursos financeiros e técnicos.

Critério	Classificação
Preço	Se aplica Não se aplica
Requerimentos	Fornece Não fornece
Manutenção	Frequentemente Não necessita Pouco
Treinamentos	Requere Não requere
Investimentos	Alto Baixo Médio
Retorno	Alto Médio Baixo

3.1.9. **Sobre o Chatbot**

A Tabela 11, representa as características descritas abaixo.

Consciência do Bot: Avalia como o bot parece ter controle sobre a conversação e se responde apropriadamente.

Boas maneiras: Capacidade dos chatbots de exibir comportamento educado e hábitos de conversação. Saudações, desculpas, sutilezas sociais e introduções.

Rigor: mede as habilidades gramaticais e sintáticas formais do chatbot.

Originalidade: a capacidade do chatbot de produzir o que parecia ser material original e também sua capacidade de tomar a iniciativa nas conversas

Tabela 11 - Sobre o Chatbot

Critério	Classificação
Consciência do Bot	1 a 5
Boas maneiras	1 a 5
Rigor	1 a 5
Originalidade	1 a 5
Personalidade	1 a 5
Relacionamento com o usuário	1 a 5
Durabilidade	1 a 5
Adaptação	1 a 5
Imitabilidade	1 a 5

4. ESTUDO DE CASO

A estrutura de avaliação dos chats apresentada neste trabalho foi realizada através de um estudo de caso. Este estudo refere-se a uma clínica médica fictícia. Veja a seguir a descrição do problema para ser solucionado através de bots.

A clínica Espaço Bem-Estar (fictícia) deseja inovar oferecendo serviços de atendimento baseado em chatbots. A proposta dela é a seguinte: um chat automatizado capaz de auxiliar os pacientes com as informações de médicos, especialidades, convênios e marcação de consultas, facilitando assim o trabalho dos (as) recepcionistas e clientes. A plataforma desejada para a utilização do chatbot é: Facebook Messenger, através da página da clínica.

4.1. IDENTIFICAÇÃO DOS TESTES

Para o desenvolvimento dos testes, o primeiro passo realizado foi criar todos os chatbots, deixando-os completos com todos os dados da clínica fictícia do estudo de caso. Depois uma tabela de avaliação foi criada para cada chat e preenchida com os resultados obtidos, como será explicado no parágrafo seguinte.

Para avaliar os resultados de acordo com os parâmetros selecionados, foi necessário colocar os bots em ação. O primeiro parâmetro utilizado foi o de recursos gerais, onde foi preciso verificar os tipos, o objetivo, a modelagem, a representatividade, a facilidade de uso, o suporte, entre outros, de cada chat. O segundo critério analisado foi o de aspectos visuais, para verificar as animações, layouts e telas dos frameworks. O terceiro parâmetro foi o aspecto de codificação, utilizado para analisar os códigos e a programação das bibliotecas. A quarta análise foi feita para verificar a eficiência dos agentes de mensagem. O parâmetro de testabilidade foi o quinto a ser utilizado, verificando os critérios que examinam as mensagens de erros, exibições de valores de lógica das funções e variáveis. A compatibilidade foi um critério utilizado para avaliar se as bibliotecas e frameworks possuem integração com aplicativos ou sites. O sétimo parâmetro utilizado foi o de

entrada/saída, onde foi possível verificar como os usuários podem apresentar os dados e o tipo de relatório gerado pelas ferramentas utilizadas. O critério de suporte avalia a qualidade e o tipo de suporte fornecido pelos pacotes utilizados. O nono critério foi o de recursos técnicos e financeiros, que examinaram as características das dos recursos utilizados, em relação ao custo e características técnicas. Por fim o parâmetro sobre o chatbot foi o último utilizado para avaliar os chats. Ele examina a consciência dos bots, boas maneiras, rigor e originalidade.

4.2. RESULTADOS

4.2.1 Framework Chatfuel

A Tabela 12 mostra a avaliação feita no framework Chatfuel de acordo com os parâmetros preestabelecido acima e detalha, de acordo com cada sessão, o que a ferramenta possui.

De maneira geral, é possível observar pela avaliação que o Chatfuel é um framework baseado em dados e tem como objetivo a criação de chatbots sem a necessidade de experiência ou conhecimento em programação. Ele possui diversas integrações com canais de comunicação, como o Facebook Messenger, Instagram, SMS e Telegram. Não há necessidade de educação formal em chatbots para a utilização, o aprendizado e o uso e aprendizado é fácil. Possui um bom suporte via internet e modelos de simulação em tempo real.

Em relação aos aspectos visuais, apresenta uma simulação simultânea ao desenvolvimento do chat, porém não forneceu a facilidade para personalizar a visualização do bot e a edição da tela. Como não é utilizado nenhuma programação, os critérios de codificação não se aplicam para essa avaliação.

Por ser um framework de fácil aprendizado, o Chatfuel contém modelos pré-existentes de bots, uma escala de tempo de construção consideravelmente curta e possibilidade de iteração, para a realização de testes. Ele possui mensagens de erro iterativas de alta qualidade.

A interface é orientada por botões e menus suspensos, a seleção de botões por mouse e uma caixa de diálogo, porém não forneceu manutenção de banco de dados.

O suporte ao usuário possui documentação de alta qualidade no site do fornecedor, com informações técnicas, grupos de discussões, tutoriais, vídeos curtos de treinamento no primeiro acesso, suporte ao telefone e consultoria.

Para a realização de chats básicos, não é necessária manutenção do pacote fornecido, que não tem custo e o valor o upgrade é consideravelmente médio e cobrado em dólar.

Sobre o chatbot, como ele é de respostas prontas, curtas e diretas, possui um bom relacionamento com o usuário.

Tabela 12 – Avaliação dos critérios – Chatfuel

Recursos Gerais	- Tipos de framework: Framework baseado em dados - Objetivo: Estudo de caso. Facilitar a criação de chatbots, sem a necessidade de experiência ou conhecimento em programação. Versões de framework para diferentes canais de comunicação: Facebook Messenger, Instagram, SMS e Telegram. - Experiência necessária para uso do framework: Nenhum. - Educação formal em chatbot necessária para uso do framework: nenhuma. - Facilidade de uso: alto. - Facilidade de aprendizado: fácil. - Modelos de simulação em tempo real: possível. - Suporte: sim. - Disponibilidade de recursos no mercado: sim. - Versatilidade: 5.

Aspectos Visuais	- Animação: Possível - Tipo de animação: semi-animação (estado a estado) - Desenvolvimento de layout de animação: Simultaneamente ao desenvolvimento do chat. - Facilidade para personalizar a visualização do chatbot: não forneceu. - Editor de tela: não forneceu. - Indicação do status do elemento de criação e testes: forneceu.
Critérios para aspectos de codificação	Não se aplica nesse framework.
Eficiência	-Robustez: alta. - Nível de detalhe: alto; - Número de elementos no chatbot: grande. - Reutilização do chatbot: Possível. - Iteração: possível. - Adaptabilidade às mudanças do chatbot: possível. - Escala de tempo para a construção do chatbot: curto. - Confiabilidade: alta. - Modelos genéricos pré-existentes: forneceu. - Construção automática de modelos: não forneceu. - Facilidade de edição de modelo: fácil.
Testabilidade	- Verificações lógicas: não forneceu. - Mensagens de erros iterativas: forneceu - Qualidade das mensagens de erro: alto.

Compatibilidade	- Integração com Facebook Messenger: Possível - Integração com Instagram: Possível - Integração com WhatsApp: Possível - Integração com SMS: Possível - Integração com Telegram: Possível - Integração com Sites: Não possível
Critérios para Entrada / Saída	- Interface orientada por menu: forneceu. - Menus suspensos: forneceu. - Tipos de seleção de menu: por mouse. - Botões de seleção: Forneceu - Caixa de diálogos: forneceu. - Manutenção de banco de dados para entrada / saída: não forneceu. - Relatórios gerais de produção: não forneceu. - Disponibilidade: não forneceu. - Relatório: não forneceu
Critérios para suporte ao usuário	- Documentação (Manuais): forneceu. - Qualidade da documentação: alta. - Glossário: não forneceu. - Material de informação técnica e promocional (e-mail, quadro de avisos): forneceu. -Grupo de discussão na internet: Forneceu. -Tutorial**:** forneceu. - Curso de treinamento (básico, avançado)**:** Forneceu. Curso de treinamento sob medida: não forneceu. Help-line (Suporte por telefone)**:** Forneceu. Manutenção do pacote**:** forneceu. Consultoria: forneceu.

Recursos técnicos e financeiros	- Preço: se aplica. - Manutenção: pouco. - Treinamentos: requere. - Investimentos: médio. - Retorno: médio.
Sobre o Chatbot	- Consciência do bot: 1 - Boas maneiras: 5 - Rigor: 5 - Originalidade: 2 - Personalidade: 3 - Relacionamento com o usuário: 5 - Durabilidade: 5 - Adaptação: 5 - Imitabilidade: 4

4.2.2 Framework ManyChat

A Tabela 13 mostra a avaliação feita no framework ManyChat de acordo com os parâmetros preestabelecido no capítulo 3 e detalha, de acordo com cada sessão, o que a ferramenta possui.

De maneira geral, a avaliação do ManyChat apresentou resultados similares ao do Chatfuel, apresentando algumas diferenças, que será explicado a seguir.

Em recursos gerias, o ManyChat diferente do Chatfuel possui capacidade de integração com plug-in JSON, e só é possível conexão com Facebook messenger e Instagram.

A facilidade de uso do ManyChat também é alta, porém, a ferramenta é toda em inglês, assim como o seu canal de suporte, enquanto o Chatfuel possui em português.

Para os aspectos visuais, eficiência, testabilidade e critérios de entrada/saída, não foram encontradas diferenças entre as ferramentas. Assim como o Chatfuel, os critérios para aspectos de codificação não se aplicam.

O ManyChat não possui uma documentação consideravelmente alta, pois apresenta pouco conteúdo, não fornece um grupo de discussão na internet e vídeos de treinamento/tutoriais de ajuda.

Por fim, assim como o Chatfuel, essa ferramenta possui pacotes gratuitos para a criação de bots básicos. Sobre o chatbot, como ele também é de respostas prontas, curtas e diretas, possui um bom relacionamento com o usuário.

Tabela 13 – Avaliação dos critérios – ManyChat

Recursos Gerais	- Framework com capacidade de integração com plug-in JSON e com aplicativos de terceiros. - Possui o objetivo de facilitar a criação de chatbot, sem a necessidade de conhecimento em programação. - É possível conectar com o Facebook Messenger e Instagram. - Não é necessária nenhuma experiência para o uso do framework. - Facilidade de aprendizado e de uso é de alto a médio, devido ser em inglês. - É possível realizar simulações em tempo real. - Existe um bom canal de suporte, porém, só na língua inglesa. - Está disponível um pacote gratuito no mercado. - Tem grande versatilidade.
Aspectos Visuais	- Animação: Possível - Tipo de animação: semi-animação (estado a estado)

	Desenvolvimento de layout de animação: Simultaneamente ao desenvolvimento do chat. - Facilidade para personalizar a visualização do chatbot: não forneceu. Editor de tela: não forneceu. Indicação do status do elemento de criação e testes: forneceu.
Critérios para aspectos de codificação	Não se aplica nesse framework
Eficiência	-Robustez: alta. - Nível de detalhe: alto; - Número de elementos no chatbot: grande. - Reutilização do chatbot: Possível. - Iteração: possível. - Adaptabilidade às mudanças do chatbot: possível. - Escala de tempo para a construção do chatbot: curto. - Confiabilidade: alta. - Modelos genéricos pré-existentes: forneceu. - Construção automática de modelos: não forneceu. - Facilidade de edição de modelo: fácil.
Testabilidade	- Verificações lógicas: não forneceu. - Mensagens de erros iterativas: forneceu - Qualidade das mensagens de erro: alto.
	- Integração com Facebook

Compatibilidade	Messenger: Possível - Integração com Instagram: Possível - Integração com WhatsApp: Através de API - Integração com SMS: Não possível - Integração com Sites: Não possível
Critérios para Entrada / Saída	- Interface orientada por menu: forneceu. - Menus suspensos: forneceu. - Tipos de seleção de menu: por mouse. - Botões de seleção: Forneceu - Caixa de diálogos: forneceu. - Manutenção de banco de dados para entrada / saída: não forneceu. - Relatórios gerais de produção: não forneceu. - Disponibilidade: não forneceu. - Relatório: não forneceu
Critérios para suporte ao usuário	- Documentação (Manuais): forneceu. - Qualidade da documentação: médio. - Glossário: não forneceu. - Material de informação técnica e promocional (e-mail, quadro de avisos): forneceu. -Grupo de discussão na internet: não forneceu. -Tutorial**:** forneceu. - Curso de treinamento (básico, avançado)**:** não forneceu. Curso de treinamento sob medida: não forneceu. Help-line (Suporte por telefone)**:** Forneceu. Manutenção do pacote**:** forneceu. Consultoria: forneceu.

Recursos técnicos e financeiros	- Preço: se aplica. - Manutenção: pouco. - Treinamentos: requere. - Investimentos: médio. - Retorno: médio.
Sobre o Chatbot	- Consciência do bot: 1 - Boas maneiras: 5 - Rigor: 5 - Originalidade: 2 - Personalidade: 3 - Relacionamento com o usuário: 5 - Durabilidade: 5 - Adaptação: 5 - Imitabilidade: 4

4.2.3 Biblioteca ChatterBot

A Tabela 14 mostra a avaliação feita na biblioteca ChatterBot de acordo com os parâmetros preestabelecidos acima e detalha, de acordo com cada sessão, o que a biblioteca possui.

De uma maneira geral é possível observar pela avaliação que a biblioteca tem como objetivo a criação de chatbots, é de fácil aprendizado, bastante popular, é muito versátil, embora não possua aspectos visuais isso não se torna um empecilho para a utilização da mesma, sendo opensource é possível acessar seu código fonte, sendo ele de alta legibilidade, possui um grande número de recursos disponíveis, é possível reutilizar o chatbot para outras construções, em poucas linhas de código é possível ter um bot funcionando, seu tempo de compilação é rápido, é possível fazer integração com diversos aplicativos, sites e programas, possui uma documentação de alto nível disponível, atualizada e conta com suporte ao usuário, é gratuito, não necessita de muita manutenção, de custo baixo e alto retorno.

Sobre o chatbot é necessário algum tempo de treino para que o mesmo não pareça tão robótico, consiga se adaptar as respostas adequadas e consiga manter um fluxo de conversa que soe natural.

Tabela 14 – Avaliação dos critérios – ChatterBot

Recursos gerais	- Biblioteca em Python. - Geração de respostas automatizadas para a entrada de um texto ou áudio de um usuário. - Pode ser usada em qualquer sistema operacional. - Necessário conhecimento mínimo em Python para sua utilização. - Não é necessária educação formal. - Alta facilidade de uso. - Alta facilidade de aprendizado. - Grande popularidade. - Possui suporte. - Disponível no mercado. - Versátil.
Aspectos visuais	A biblioteca em si não fornece.
Aspectos de codificação	- Possui acesso ao código fonte da biblioteca. - Código fonte de alta legibilidade. - Alta precisão de código adicionado. - Possível instalar armazenamento. - Nomes das funções definidos pelo sistema. - Há suporte para conceitos de programação.
Eficiência	- Altamente robusto. - Alto nível de detalhe. - Grande número de elementos disponíveis.

	- Possível reutilização do chatbot. - Alta adaptabilidade às mudanças. - Tempo de compilação rápido. - Possui sensibilidade a letras maiúsculas e minúsculas. - Pode ser construído um chatbot em pouco tempo. - Alta confiabilidade.
Testabilidade	- Possui verificações lógicas. - Alta qualidade nas mensagens de erro. - Diagnostico do erro no momento da compilação. - Fácil depuração.
Compatibilidade	- Possível integração com Facebook Messenger. - Possível integração com Whatsapp. -Possível integração com SMS. - Possível integração com sites.
Entrada/Saída	- Entrada através de input pelo terminal.
Suporte	- Fornece documentação. - Alta qualidade da documentação. - Possui glossário. - Fornece tutorial. - Fornece manutenção do pacote.
Recursos técnicos e financeiros	- Gratuito. - Pouca manutenção. - Baixo custo de investimento. - Alto retorno. - Não requere treinamento.

Sobre o chatbot	- Consciência do bot: Depende do tempo de treinamento. - Boas maneiras: 5 - Rigor: 5 - Originalidade: 4 - Personalidade: 4 - Relacionamento com o usuário: 4 - Adaptação: 5 - Imitabilidade: 5

4.2.4 Biblioteca NTLK

A Tabela 15 mostra a avaliação feita na biblioteca NTLK de acordo com os parâmetros preestabelecidos acima e detalha, de acordo com cada sessão, o que a biblioteca possui.

De uma maneira geral é possível observar através da avaliação que a biblioteca tem como objetivo trabalhar com dados de linguagem humana, ou seja, processamento de linguagem natural, não é necessário um conhecimento profundo em programação para conseguir utilizá-la, é versátil, possui suporte online, está disponível gratuitamente, possui uma variedade de recursos disponíveis, não possui recursos visuais, alta taxa de confiabilidade, fácil edição do modelo, durante seu teste mostra mensagens de erros apontando aonde estão os mesmos, permite integração com diversos aplicativos e sites, possui documentação atualizada, glossário, tutorias de como usar a biblioteca, cursos para treinamento, possui manutenção recorrente do pacote, necessita de pouca manutenção e o comportamento do chatbot é baseado no seu treinamento, o formato da sua base de dados ajuda o bot a responder corretamente desde o primeiro contato com o usuário, porém isso faz com que ele não soe muito natural, pois suas respostas estão todas estabelecidas para determinadas situações.

Tabela 15 – Avaliação dos critérios – NTLK

Recursos gerais	- Biblioteca em python. - Possui o objetivo de trabalhar com dados de linguagem humana, ou seja, processamento de linguagem natural. - É possível trabalhar com qualquer sistema operacional. - Necessário conhecimento mínimo na linguagem Python para construir um chatbot básico. - Facilidade de uso é baixa. - Fácil aprendizado. - Por ser uma biblioteca popular possui um bom suporte. - Sendo open source está disponível gratuitamente no mercado. - Tem grande versatilidade.
Aspectos visuais	- A biblioteca não possui recursos que possibilitam aspectos visuais.
Eficiência	- Robustez alta. - Nível de detalhe alto. - Números de elementos no chatbot grande. - É possível fazer a reutilização do chatbot. - Possui grande adaptação a mudanças. - Tempo de compilação curto, - Em pouco tempo é possível construir um chatbot. - Taxa de confiabilidade alta. - Fácil edição do modelo.
Testabilidade	- Possui verificações lógicas. - Mensagens de erros são iterativas. - Qualidade das mensagens de erro

	é alta. - Momento do diagnóstico de erro é durante a compilação. - Fácil depuração.
Compatibilidade	- Possível integração com Facebook Messenger. - Possível integração com o Whatsapp. - Possível integração com sites. - Possível integração com SMS.
Suporte	- Possui documentação. - Qualidade alta da documentação. - Possui glossário. - Possui grupos de discussão na internet. - Possui tutorias na internet. - Possui cursos de treinamento. - Possui manutenção do pacote.
Recursos técnicos e financeiros	- Gratuito. - Necessita pouca manutenção. - Baixo investimento. - Alto retorno.
Sobre o chatbot	- Consciência do bot: 5 - Boas maneiras: 5 - Rigor: 5 - Originalidade: 3 - Personalidade: 3 - Relacionamento com o usuário: 3 - Durabilidade: 5 - Adaptação: 5 - Imitabilidade: 4

4.3 ANÁLISE DOS RESULTADOS

É possível observar, através de todos os testes realizados nos frameworks e analisando seus resultados que no critério de recursos gerais o Chatfuel e o ManyChat são muitos semelhantes, a diferença é que o primeiro é compatível com o Facebook Messenger, Instagram, SMS e Telegram, enquanto o outro só é possível conectar com o Facebook Messenger e Instagram. Nos quesitos de aspectos visuais, eficiência, testabilidade, critérios de entrada/saída, recursos técnicos e financeiros e os critérios sobre os chats, nenhuma diferença foi encontrada nos testes.

A compatibilidade é diferenciada, pois o Chatfuel pode se conectar com o Facebook Messenger, Instagram, WhatsApp, SMS, Telegram e sites, enquanto o ManyChat é compatível somente com o Facebook Messenger e Instagram, com o WhatsApp é possível através de API. Os critérios para suporte ao usuário possuí uma pequena diferença, o Chatfuel possui uma documentação de alta qualidade, com uma abordagem completa de cada tópico necessário na criação do chatbot, fornecendo grupos de discussão na internet e cursos de treinamentos. Já o ManyChat apresenta uma pequena documentação, com poucos tópicos e não possui um grupo de discussão, dificultando assim, o estudo aprofundado da ferramenta. Por sim, os critérios para aspectos de codificação não se aplicam nos testes dos frameworks, pois não foi utilizado nenhuma programação, somente os recursos disponíveis nas plataformas.

Levando em conta todos os testes feitos nas bibliotecas e analisando seus resultados é possível observar que no aspecto técnico e financeiro, ambas possuem as mesmas vantagens sendo gratuitas, possuindo alto retorno e baixo orçamento e necessitando de pouca manutenção, é possível observar também que as duas possuem documentação atualizada, suporte às bibliotecas e canal de comunicação com os desenvolvedores, o que torna possível desenvolver um chatbot de maneira rápida e eficiente com pouco tempo de aprendizado, suas testabilidades também são bem parecidas apontando os erros no momento da compilação, com mensagens claras e mostrando diretamente aonde se encontra o erro no código. Por serem bibliotecas feitas unicamente para criação de chatbots nenhuma possui interface gráfica, mas isso rapidamente se resolve, pois é possível utiliza-las com bibliotecas

que permitem a criação de interface gráfica, tornando o chatbot criado de fácil utilização por qualquer usuário. Outra semelhança em ambas é sua compatibilidade, sendo de fácil integração com diversos aplicativos e websites.

O que mais difere uma biblioteca da outra é sua codificação e comportamento do chatbot, enquanto com a NLTK a construção chega a ser um pouco mais complexa, necessitando de um conhecimento maior, a ChatterBot em poucas linhas já é possível colocar um bot para dar respostas automatizadas a entrada dos usuários.

Por outro lado, a base de dados de treinamento da NLTK é mais bem elaborada, sendo dividida em “Intenções” aonde cada intenção tem suas possíveis entradas de dados e as possíveis respostas as entradas, com isso o treinamento do bot passar a ser mais “certeiro” pois ele sabe exatamente aonde achar as respostas adequadas, o que torna sua utilização para desenvolvimento de chatbots aonde o mesmo precisa dar respostas mais lógicas e rápidas a perguntas diretas muito mais ágil, porém o seu comportamento fica mais robótico com todas as respostas já pré-definidas. Com o ChatterBot é ao contrário da NLTK, a biblioteca requer um treinamento maior para conseguir responder as entradas adequadamente, mas por possuir um armazenamento de cada caso, com o tempo logo é possível manter um fluxo de conversa aparentando ser a conversa com um humano, o que é o intuito da biblioteca, sendo excelente para o desenvolvimento de chatbots aonde a conversa com o bot precisa fluir mais naturalmente.

Por fim, ambas as bibliotecas suprem bem as necessidades na criação de um chatbot, se integram a várias outras aplicações, são de fácil aprendizado, seu treinamento é ágil e em pouco tempo já é possível possuir um bot funcionando.

4.3. **TRABALHOS FUTUROS**

O presente trabalho selecionou pontos específicos a serem trabalhados, englobando apenas alguns dos aspectos de criação de chatbot. Dessa forma, pode-se citar outros pontos importantes que podem ser melhores estudados em uma eventual continuidade de pesquisa:

I) Estudar e conhecer outros recursos de criação de chatbots;

II) Os frameworks foram testados somente no aplicativo do Facebook Messenger, portanto é aconselhado a utilização de todas as compatibilidades;

III) Pesquisar e entender outros parâmetros para avaliar mais profundamente as bibliotecas e frameworks.

IV) Conectar os recursos utilizados com calendários para os pacientes terem a possibilidade de verificar os dias e horários disponíveis para consultas;

V) Conectar um banco de dados para verificar se o usuário já é paciente da clínica, facilitando o cadastro;

VI) Treinar os chats para ficarem mais humanizados e menos robóticos.

5. CONCLUSÃO

O presente trabalho propôs um estudo sobre chatbot e os métodos de desenvolvimento, utilizando quatro pacotes diferentes, onde duas são bibliotecas e dois são frameworks, com isso, são dois tipos de recursos de criação.

Os quatro chatbots implementados demonstraram algumas diferenças e semelhanças entre seus resultados. A principal causa para essa semelhança é devida os aspectos técnicos e financeiros, pois os recursos utilizados possuem vantagens sendo gratuitos, alto retorno e baixo orçamento, sem a necessidade de manutenção.

Com eles, é possível desenvolver bots de maneira rápida e eficiente com pouco tempo de aprendizado. A diferença entre os agentes de criação de chatbot é a compatibilidade com aplicativos e sites. Enquanto uma é possível conectar somente no Facebook Messenger, os outros são compatíveis com sites da web, WhatsApp, SMS e etc.

6. REFERÊNCIAS BIBLIOGRÁFICAS

CHATFUEL. ChatFuel, 2022. **Documentação Chatfuel**. Disponível em: <https://docs. chatfuel.com/ en/>. Acesso em 05 de Set de 2022.

CHATFUEL. Chatfuel, 2022. **Planos Chatfuel**. Disponível em: <https://chatfuel.com/ pricing-new/>. Acesso em 05 de Set de 2022.

MANYCHAT. ManyChat, 2022. **How to use ManyChat**. Disponível em: <https://manychat.com/resources/how-to>. Acesso em 05 de Set de 2022.

WEB HOLSTING, Web Holsting, 2019. **Chatbot for your bussiness: Chatfuel, Verloop, ManyChat, and Gupshup Compaed**. Disponível em: <https://www.web hostingsecretrevealed.net/blog/web-business-ideas/three-chatbots-thatll-boost-your-business-for-free/>. Acesso em 15 de Set de 2022.

SANTOS, S.S. **Desenvolvimento do chatbot Ellen como ferramenta de alerta e acompanhamento para pessoas com doenças crônicas não transmissíveis**, 2018. 45f. Graduação em Engenharia Biomédica – Universidade Federal do Rio Grande do Norte, Natal/RN, 2018. Disponível em: <https://eb.ct.ufrn.br/wp-content/uploads/2019/03/Su%C3%A9llen-Santos.pdf>. Acesso em: 18 de Ago de 2022.

GHIDINI, I; MATTOS, W. W. **Desenvolvimento e aplicação de um chatbot para auxiliar o atendimento ao cliente**, 2018. 75f. Graduação em Sistemas de Informação - Universidade do Sul de Santa Catarina, Palhoça, 2018. Disponível em: <https://repositorio.animaeducacao.com.br/bitstream/ANIMA/11038/3/TCC%20Chatterbot%20-%20Itamar%20Ghidini%20e%20Winicius%20Mattos%20FINAL.pdf>. Acesso em: 10 de Set de 2022.

HLUPIC, V; IRANI, V; PAUL, R. J. **Evaluate Framework for Simulation Software**, 1999. 17f. Brunel University, Departament of Information and Computing, Uxbrid, UK, 1999.

NVOIP. NVOIP, 2021. **Chat: saiba a importância dessa ferramenta para a sua empresa**. Disponível em: <https://www.nvoip.com.br/blog/chat/>. Acesso em 15 de Set de 2022.

RDD. Recanto do Dragão, 2014. **Como surgiu o sistema de bate papos na internet**. Disponível em: <https://recantododragao.com.br/como-surgiu-o-sistema-de-bate-papos-na-internet/>. Acesso em 15 de Set de 2022.

ANALYTICS VIDHYA. Analytic Vidhya, 2021. **Build a simple Chatbot using NLTK Library in Python**. Disponível em: <https://www.analyticsvidhya.com/blog/2021/07/build-a-simple-chatbot-using-python-and-nltk/>. Acesso em 20 de Ago de 2022.

NLTK. Natural Language Tollkit, 2022. **Documentation**. Disponível em: <https://www.nltk.org/>. Acesso em 20 de Ago de 2022.

MEDIUM. Medium, 2020. **Introdução a NLTK com Dom Casmurro**. Disponível em: <https://medium.com/turing-talks/uma-an%C3%A1lise-de-dom-casmurro-com-nltk-343d72dd47a7>. Acesso em 20 de Ago de 2022.

PYKIT. Pikit, 2022. **How to make AI Chatbot in Python using NLP (NLTK) in 2022**. Disponível em: <https://pykit.org/chatbot-in-python-using-nlp/>. Acesso em 01 de Set de 2022.

DATACAMP. Datacamp, 2020. **Building a Chatbot using Chatterbot in Python**. Disponível em: <https://www.datacamp.com/tutorial/building-a-chatbot-using-chatterbot/>. Acesso em 15 de Set de 2022.

UPGRAD. UpGrade, 2022. **How to make a Chatbot in Phython Step By Step [Chatterbox Guide]**. Disponível em: <https://www.upgrad.com/blog/how-to-make-chatbot-in-python/>. Acesso em 15 de Set de 2022.

UPGRAD. UpGrade, 2022. **How to make a Chatbot in Phython Step By Step [Chatterbox Guide]**. Disponível em: <https://www.upgrad.com/blog/how-to-make-chatbot-in-python/>. Acesso em 15 de Set de 2022.

STACK OVERFLOW. Stack overflow, 2019. **Como adicionar treinamento externo no Chatterbot**. Disponível em: <https://pt.stackoverflow.com/questions/341358/como-adicionar-treinamento-externo-no-chatterbot>. Acesso em 30 de Set de 2022.

CHATTERBOT. Chatterbot, 2021.**Training**. Disponível em: <https://chatterbot.readthedocs.io/en/stable/training.html>. Acesso em 30 de Set de 2022.

CHATTERBOT. Chatterbot, 2021. **About ChatterBot**. Disponível em: <https://chatterbot.readthedocs.io/en/stable/>. Acesso em 30 de Set de 2022.

CHATTERBOT. Chatterbot, 2021. **About ChatterBot**. Disponível em: <https://chatterbot.readthedocs.io/en/stable/>. Acesso em 30 de Set de 2022.

PYPI. Pypi, 2021. **ChatterBot 1.0.8**. Disponível em: <https://pypi.org/project/ChatterBot/>. Acesso em 30 de Set de 2022.

B2B STACK. B2B Stack, 2019. **O que é chatbot: tudo o que você precisa saber**. Disponível em: <https://blog.b2bstack.com.br/o-que-e-chatbot/>. Acesso em 18 de Ago de 2022.

MOBILE TIME. Mobile time, 2021. **Base de robôs de conversação no Brasil dobra em um ano**. Disponível em: < https://www.mobiletime.com.br/noticias/27/08/2021/brasil-dobra-a-quantidade-de-robos-de-conversacao-em-um-ano/>. Acesso em 30 de Set de 2022.

COMUNIQUE-SE PORTAL. Comunique-se portal, 2021. **Crescimento do uso de chatbots no Brasil em 2021**. Disponível em: < https://portal.comunique-se.com.br/264310-crescimento-do-uso-de-chatbots-no-brasil-em-2021/>. Acesso em 30 de Set de 2022.

MICROSOFT LEARN. Microsoft Learn, 2022. **O que é SDK do Bot Framework.** Disponível em: <https://learn.microsoft.com/pt-br/azure/bot-service/bot-service-overview?view=azure-bot-service-4.0>. Acesso em 18 de Ago de 2022.

LE WAGON. Le Wagon, 2020. **O que é um framework? Para que serve e por que você deveria saber.** Disponível em: < https://www.lewagon.com/pt-BR/blog/o-que-e-framework>. Acesso em 18 de Ago de 2022.