BRUNO BRANDÃO INÁCIO
PEDRO DE MORAES LIGABUE

CHATBOTS PARA ANTECIPAÇÃO DE DEMANDA DE CLIENTES NUMA DISTRIBUIDORA DE ENERGIA ELÉTRICA

São Paulo
2020

BRUNO BRANDÃO INÁCIO
PEDRO DE MORAES LIGABUE

CHATBOTS PARA ANTECIPAÇÃO DE DEMANDA DE CLIENTES NUMA DISTRIBUIDORA DE ENERGIA ELÉTRICA

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro .

São Paulo
2020

BRUNO BRANDÃO INÁCIO
PEDRO DE MORAES LIGABUE

CHATBOTS PARA ANTECIPAÇÃO DE DEMANDA DE CLIENTES NUMA DISTRIBUIDORA DE ENERGIA ELÉTRICA

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro .

Área de Concentração:
Engenharia de Computação

Orientadora:
Dra. Anarosa Alves Franco Brandão

São Paulo
2020

Nome: INÁCIO, Bruno Brandão

Nome: LIGABUE, Pedro de Moraes

Título: Chatbots para antecipação de demanda de clientes numa distribuidora de energia elétrica

Monografia (Trabalho de Conclusão de Curso) apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Bacharel em Engenharia na área de Engenharia de Computação.

Trabalho entregue para o departamento em:

Responsáveis

Profa. Dra. Anarosa Alves Franco Brandão
(Orientadora)

Bruno Brandão Inácio (Orientado)

Pedro de Moraes Ligabue (Orientado)

Aos nossos pais, familiares e amigos, que nos apoiaram durante este ano de 2020.

AGRADECIMENTOS

Aos meus pais, Elaine e Jarbas, e à minha avó Geni, pelo apoio e suporte ao longo de toda a minha vida e pelo acolhimento e compreensão ao longo deste ano.

Bruno

À minha família, que me apoiou todo este tempo, e ao Los Hermanos.

Pedro

À Professora Anarosa, por orientar e apoiar o desenvolvimento do trabalho.

Aos nosso amigos que estiveram conosco e contribuíram para a etapa de testes do projeto.

Aos professores do PCS, que contribuíram com o conhecimento necessário para o desenvolvimento deste trabalho e para a nossa formação.

À Escola Politécnica e à Universidade de São Paulo, por fornecer um ambiente de aprendizado e de convívio ao longo dos últimos 5 anos.

"No man's knowledge here can go beyond his experience"

- John Locke

RESUMO

Neste projeto de formatura são estudados métodos de apoio à interação, em forma de conversa de texto, entre usuários e *chatbots* em um sistema de atendimento ao cliente, no contexto de uma distribuidora de energia elétrica. Os *chatbots* serão integrados a um sistema multiagente desenvolvido no contexto de um projeto de mestrado. Primeiramente foi realizado o desenvolvimento de uma interface gráfica para sistemas *web*, capaz de receber requisições de usuários e de se comunicar com um servidor, possibilitando uma interação simples e intuitiva e contando com recursos de acessibilidade. Em seguida, foi feita a análise e escolha de diferentes algoritmos de Processamento de Linguagem Natural (PLN), de forma que estes fossem combinados em um *pipeline*, construindo um *chatbot* capaz de identificar e responder questões dos usuários de forma coerente.

Palavras-Chave *Chatbots*, Atendimento automatizado.

ABSTRACT

In this final project, methods of supporting the interaction, in the form of a text conversation between users and chatbots, are studied in a customer service system, in the specific domain of an electric power distribution company. The chatbots will be integrated into a multi-agent system developed in the context of a master's project. Firstly, a graphical interface for web systems was developed, capable of receiving requests from users and of communicating with a server, enabling a simple and intuitive interaction, with accessibility features. Then, an analysis and a selection of different Natural Language Processing (NLP) algorithms were done, so that they would be combined in a pipeline, building a chatbot capable of identifying and answering questions from users coherently.

Keywords *Chatbots*, Automated customer service.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

IA - Inteligência Artificial

IHC - Interação Humano-Computador

NLU - *Natural Language Understanding*

PLN - Processamento de Linguagem Natural

SAC - Serviço de Atendimento ao Consumidor

SMA - Sistema Multiagente

SUMÁRIO

1 INTRODUÇÃO

O atendimento a clientes atualmente é baseado em interações com atendentes humanos, que buscam soluções manualmente na base de dados a que têm acesso. Porém, com a chamada transformação digital, vem ocorrendo um processo de automação na indústria e uma crescente adoção de canais digitais por parte tanto de empresas quanto de pessoas. Desta forma, é natural que os canais de Serviço de Atendimento ao Consumidor (SAC) também sejam vistos como uma oportunidade de automatização.

Esta tarefa, no entanto, apresenta como grande desafio a construção de um modelo computacionalmente viável capaz de não só compreender o que o cliente deseja, mas também de gerar uma resposta coerente, realizar as ações adequadas e se comunicar de maneira simples e intuitiva, sendo suficientemente parecido com uma interação humana para fornecer uma melhor experiência aos clientes.

Neste contexto, a área da Inteligência Artificial aparece com grande destaque, se aproveitando do crescente poder computacional disponível nos computadores modernos. A partir da combinação de diferentes técnicas e algoritmos de Processamento de Linguagem Natural, Aprendizado de Máquina e Aprendizado Profundo, podem ser construídos os chamados *chatbots*, capazes de realizar tarefas como identificação de contexto e reconhecimento da intenção de um usuário em uma frase, extração de entidades para obter valores fornecidos e realização de ações de acordo com as necessidades do usuário [1].

O desenvolvimento dos *chatbots* envolve campos da Engenharia da Computação como a Engenharia de *Software* e a Inteligência Artificial e os bons resultados dependem do uso de técnicas e algoritmos modernos. No entanto, apesar da complexidade de desenvolvimento, uma implementação adequada do sistema de atendimento automatizado tem grande capacidade de geração de valor para uma empresa.

1.1 Objetivo

O objetivo deste trabalho é a implementação de um sistema baseado em *chatbots* capaz de realizar o atendimento automatizado de clientes no domínio específico de uma distribuidora de energia elétrica. Esta implementação consiste em uma plataforma *web*, que inclui uma interface (*front-end*) moderna, simples e com recursos de acessibilidade, que possibilita a interação entre o *chatbot* e os usuários, e um servidor (*back-end*) responsável por implementar o *chatbot* e receber as requisições do *front-end*. O *chatbot* é implementado através da combinação de diferentes algoritmos de Processamento de Linguagem Natural em um *pipeline*, que devem ser adequadamente escolhidos através de uma análise comparativa e devem funcionar para um conjunto de fluxos de conversas previamente definidos.

1.2 Motivação

O desenvolvimento deste projeto de formatura explora e integra dois campos muito importantes da Engenharia da Computação, objetivando a resolução de problemas reais: a Engenharia de *Software* para o desenvolvimento de sistemas *web* e o uso de técnicas modernas de Processamento de Linguagem Natural, Aprendizado de Máquina e Aprendizado Profundo, na área de Inteligência Artificial.

A área de IA tem apresentado significativos avanços que possibilitam a construção de *chatbots* capazes de conversar de maneira cada vez mais próxima à humana. Alguns modelos altamente complexos desenvolvidos recentemente com o objetivo de avaliar o potencial das técnicas modernas de PLN envolvem o uso de redes neurais com bilhões de parâmetros que precisam ser aprendidos através de enormes conjuntos de dados e que são capazes de conversar de modo similar a um ser humano e de maneira geral sobre diferentes assuntos [2].

O estado de arte dos *chatbots* de domínio amplo [3] demonstra o grande potencial das técnicas modernas de PLN que, com uma diferença de escala na quantidade de parâmetros a serem aprendidos e na quantidade de dados usados no treinamento, podem ser aplicados na produção de *chatbots* de domínio específico capazes de atender clientes em interações similares a uma conversa com outro ser humano.

1.3 Justificativa

O sistema de atendimento automatizado a clientes, se implementado de maneira adequada, gera benefícios tanto para as empresas quanto para os usuários envolvidos. Do ponto de vista da empresa, os *chatbots* representam um modelo de atendimento altamente escalável e ágil, que permite que a empresa atinja requisitos de qualidade de prestação de serviço e possibilita a geração de valor através da redução de gastos com centrais de atendimento. Esta economia pode ser realocada em recursos de outras áreas a fim de melhorar a experiência dos consumidores e de atrair e satisfazer um público mais jovem que está acostumado com interações mais humanas com a tecnologia. Do ponto de vista dos clientes, o sistema automatizado implica em uma redução do tempo gasto com ferramentas de busca e em filas de espera por suporte, bem como em uma simplificação do acesso a informações [4].

1.4 Organização do Trabalho

Este trabalho está dividido de modo a apresentar primeiramente o referencial teórico de embasamento para o projeto, seguido pela metodologia adotada para o desenvolvimento e pela especificação e proposta de implementação do projeto. Por fim, são apresentados os testes de validação e as considerações finais do projeto.

O capítulo 2 aborda o Referencial Teórico e trata de descrever os principais conceitos teóricos que permeiam o desenvolvimento de um *chatbot*, incluindo conceitos de Sistemas Multiagente e Processamento de Linguagem Natural.

O capítulo 3 sobre a Metodologia descreve o processo de desenvolvimento do projeto, envolvendo pontos considerados para se chegar à arquitetura que foi implementada. Além disso, também apresenta justificativas para as escolhas dos algoritmos para o PLN, a análise para a implantação em nuvem e os testes de validação com usuários.

O capítulo 4 traz a Especificação, que expõe as tecnologias e técnicas envolvidas no desenvolvimento do *software*, como linguagens de programação, bibliotecas e protocolos usados, os detalhes da implementação e os recursos desenvolvidos ao longo do projeto.

O capítulo 5 de Testes e Avaliação apresenta a maneira como foi feita a implantação dos *chatbots* e do *front-end* em nuvem para tornar o projeto acessível aos usuários, bem como os detalhes dos testes realizados tanto para a interface com o usuário quanto para os fluxos de conversa com os *chatbots*.

Por fim, o capítulo 6 apresenta as Considerações Finais do projeto, apresentando um balanço dos resultados obtidos e as perspectivas de continuidade por parte de trabalhos que poderiam se basear neste projeto.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os conceitos pertinentes ao trabalho e que serão utilizados ao longo deste documento. Eles estão divididos em três partes principais: uma introdução sobre **Inteligência Artificial**, uma seção sobre **Comunicação em Sistemas Multiagente**, abordando o tema de *chatbots*, e o **Processamento de Linguagem Natural**.

2.1 Inteligência Artificial

De acordo com Russel e Norvig [5, capítulo 1], o conceito de Inteligência artificial (IA) pode ser definido de quatro maneiras diferentes, dependendo do que se espera de um sistema inteligente. O sistema pode: pensar como um ser humano, agir como um ser humano, pensar racionalmente ou agir racionalmente.

Com base nestas distinções, este projeto busca criar um sistema que se comporte como um ser humano. O sistema será um *chatbot* que deve ser capaz de se comunicar com pessoas utilizando linguagem natural em formato de texto escrito em português.

Aprendizado de Máquina é uma área da Inteligência Artificial que se baseia no uso de algoritmos com capacidade realizar reconhecimento de padrões. A partir disso, são gerados modelos analíticos com capacidade de generalização para extrapolar resultados ou decisões para entradas de dados não vistas anteriormente [6].

Os algoritmos de aprendizado de máquina podem ser divididos em quatro classes: aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semissupervisionada e aprendizagem por reforço. Além disso, técnicas modernas baseadas em redes neurais representam um campo do Aprendizado de Máquina chamado de Aprendizagem Profunda, que possibilita a análise de padrões complexos de dados para gerar modelos que realizam tarefas atingindo índices de acerto similares aos de seres humanos [7].

2.2 Comunicação em Sistemas Multiagente

Sistemas Multiagente são sistemas distribuídos em que os agentes, ao mesmo tempo em que possuem um alto grau de autonomia, se comunicam de uma maneira rigorosamente especificada, através de protocolos de interação bem definidos. No contexto deste tipo de sistema, os agentes são tanto autônomos quanto heterogêneos, isto é, atuam de maneiras bastante independentes e diferentes entre si. Esta característica torna o desenvolvimento de protocolos uma tarefa complexa, uma vez que protocolos de interação priorizam reusabilidade e modularidade [8].

Os métodos de comunicação tradicionais envolvendo Inteligência Artificial partem do pressuposto de que os agentes são desenvolvidos e separados com base em conceitos de cognição, como intenções, objetivos e crenças. A comunicação entre eles, portanto, se basearia também em conexões cognitivas. Além disso, estes métodos têm como ponto de partida a interação humano-computador (IHC) e o entendimento de linguagem natural (*Natural Language Understanding* - NLU) [8].

As ferramentas baseadas nestes métodos buscam interpretar mensagens de usuários para escolher uma tarefa a ser feita, como uma consulta em um banco de dados, por exemplo. Estas ferramentas constroem modelos com base em domínios específicos para conseguir realizar a antecipação de demandas os usuários [8].

O motivo pelo qual estas ferramentas são pensadas como agentes, e não somente aplicações utilizando processamento de linguagem natural (PLN), é relativo ao fato de elas terem adquirido um grau maior de autonomia, ou seja, as ferramentas operavam de forma mais contida e auto-suficiente, e ao fato delas estabelecerem comunicação com outras ferramentas semelhantes, a fim de melhorarem os modelos umas das outras.

Neste contexto, o projeto se baseia nestes métodos tradicionais envolvendo IA para desenvolver o *chatbot*, tanto ao enxergar a aplicação como um sistema auto-contido e autônomo, quanto ao entender o contexto no qual se encaixa, um sistema multiagente já desenvolvido.

2.3 Chatbot

Um *chatbot* é um software capaz de interagir com pessoas de forma conversacional, por texto ou voz, com base em linguagem natural, fazendo com que o computador reproduza o comportamento um ser humano [9].

Chatbots funcionam com base em técnicas de Inteligência Artificial como Processamento de Linguagem Natural e Aprendizado de Máquina, podendo ser divididos em dois tipos principais [9]:

- Declarativos: *chatbots* que focam apenas na execução de tarefas específicas, com um único domínio e são baseados em regras, PLN e poucas parcelas de Aprendizado de Máquina;
- Conversacionais: *chatbots* mais sofisticados, utilizam mais técnicas de aprendizado de máquina, têm funcionamento baseado em dados e podem funcionar como assistentes virtuais que fornecem sugestões aos usuários. Podem ainda ser capazes de prever demandas e se adaptar a contextos e usuários.

No contexto do projeto, há um domínio específico e respostas que são pré-estabelecidas com base em regras de fluxos de conversa, o que se aproxima dos *chatbots* declarativos definidos anteriormente. Ao mesmo tempo, os modelos utilizados para classificação de intenção e extração de entidades são baseados em Aprendizado de Máquina e são feitos com base em um corpus de texto, assumindo também elementos dos *chatbots* conversacionais. Assim, o *chatbot* desenvolvido se comporta como um agente, com suas próprias regras de negócio e funcionalidades, servindo como interface entre o usuário e o resto do sistema multiagente.

2.4 Processamento de Linguagem Natural

O Processamento de Linguagem Natural é uma área de estudo que lida com a comunicação entre seres humanos e computadores através das linguagens naturais, ou seja, as línguas faladas pelos seres humanos. Segundo Russell e Norvig [5, capítulo 22], as linguagens naturais se diferenciam das linguagens formais por, por exemplo, não conterem um conjunto definido de cadeias, formadas a partir de uma gramática com regras definidas, incluírem ambiguidade e mudarem constantemente. Por estes motivos, a modelagem de linguagens naturais é mais consistente se feita com o uso de modelos estatísticos no lugar de regras de produção, como uma gramática formal.

Os métodos de processamento baseados em inteligência artificial podem ser divididos em três tipos: os métodos clássicos, os métodos empíricos e estatísticos, e os métodos utilizando redes neurais [10]. Nas subseções seguintes descrevemos as abordagens clássica e estatística.

2.4.1 Abordagem clássica

Os métodos clássicos, são mais adequados para contextos em que não há um *corpus* extenso de textos. Estes métodos entendem o processamento de linguagens naturais como uma sequência de etapas, dividindo tarefas relativas à sintaxe, à semântica, à pragmática, além da formatação do texto em si. Um *pipeline* de processamento padrão consistiria dos seguintes etapas:

1. **Tokenização**: faz parte do pré-processamento do texto. Trata de identificar e separar palavras de forma a criar unidades independentes (*tokens*) que podem ser associados uns aos outros nas análises sintáticas e semânticas que seguem [11];
2. **Análise Léxica**: desenvolve a separação em palavras da etapa anterior, analisando a função léxica de cada *token*;
3. **Análise Sintática**: busca relacionar as palavras/*tokens* a fim de gerar frases que façam sentido sintaticamente.
4. **Análise Semântica**: trata de extrair o significado das frases, extraindo entidades importantes e intenções.
5. **Análise Pragmática**: elabora a análise de significado, abordando ambiguidade, informações implícitas e significados que não permeiam o conteúdo direto do texto, mas sim o contexto.

Além destas etapas, é comum adicionar etapas intermediárias como:

1. **Remoção de Palavras Vazias**: trata da remoção de palavra muito frequentes mas com pouca relevância (ex.: “e”, “ou”, “para”, “a”, “o” etc.);
2. ***Stemming***: trata de remover os sufixos de palavras de forma a reduzir o número total de palavras diferentes. Um exemplo seria a remoção do “s” no final de palavras no plural, fazendo com que “gato” e “gatos”, por exemplo, recebam a mesma classificação [11];
3. ***Lemmatization***: similar ao *Stemming*, no entanto não é feita uma simples remoção de sufixos, sendo feita uma análise morfológica de cada palavra, a fim de gerar a forma base da palavra, o lema (diferente de radical) [11];

4. **Normalização**: trata de corrigir possíveis erros de escrita no texto, a fim de facilitar o entendimento pelos algoritmos semânticos [11];

5. ***Keyword Extraction***: trata de identificar as palavras-chave de um texto, excluindo artigos e proposições, por exemplo;

2.4.2 Abordagem estatística

Os métodos estatísticos se baseiam em aprendizado de máquina e utilizam um grande *corpus* de textos para desenvolver modelos probabilísticos. Os métodos utilizando redes neurais tem substituído muitas outras abordagens estatísticas, sendo aplicáveis em casos de uso semelhantes, mas sendo normalmente mais poderosos e requisitando *corpus* ainda maiores de textos.

A seguir, são identificadas algumas técnicas de PLN muito úteis para o entendimento da linguagem natural:

1. ***Tokenizer***: trata da separação de um texto em palavras (*tokens*) a fim de facilitar a interpretação. Esta etapa é necessária para a grande parte do restante do *pipeline*. O exemplo mais simples se baseia em uma separação do texto baseado em caracteres de espaço, mas alguns métodos mais avançados utilizam conhecimentos do idioma;

2. ***Bag-of-words***: método para simplificar a representação de texto criando um vetor com número de dimensões suficiente para representar cada palavra do vocabulário presente nos dados de treinamento como uma posição, de modo a possibilitar o uso de algoritmos estatísticos. A representação leva em consideração a frequência da presença das palavras no texto, mas não considera pontos como gramática e ordem de aparição. Para aumentar a robustez da representação, também pode-se incluir a presença de *n-grams*, levando o algoritmo de *bag-of-words* a considerar também a presença de n palavras consecutivas, buscando representar de certa forma a ordem e o contexto das palavras, ou de n conjuntos de caracteres, buscando aceitar palavras com combinação parcial.

3. ***Intent Classification*** (Classificador de Intenção): reconhece a intenção de uma frase de acordo com uma lista de possibilidades previamente definida. O funcionamento desta técnica se baseia em métricas de similaridade entre o texto digitado pelo usuário e os textos utilizados durante o treinamento e os algoritmos são poderosos o suficiente para generalizar de maneira correta a ponto de reconhecer corretamente frases que não estavam presentes durante o treinamento;

4. ***Entity Extraction*** (Extração de Entidade): trata da identificação de termos específicos presentes em partes do texto que retratam alguma entidade, como por exemplo: nome de uma pessoa, nome de um lugar e documento de identificação ou número de telefone. É usado para obter informações relevantes passadas pelo usuário e é capaz até mesmo de reconhecer diferentes palavras como sinônimas. Os algoritmos modernos podem funcionar utilizando desde expressões regulares manualmente definidas até Cadeias de Markov;

5. ***Dependency Parsing***: trata de identificar a relação sintática entre as palavras de um texto;

6. **Response Selector**: é responsável por definir respostas adequadas de acordo com as entradas do usuário. Modelos mais simples podem ser baseados em respostas pré-definidas de acordo com a intenção identificada na fala do usuário e de acordo com fluxos de conversa configurados. Modelos mais complexos podem ser baseados em Redes Neurais Artificiais treinadas para serem capazes de gerar saídas corretas com base generalizações das entradas às quais foram expostas durante treinamento. Estes modelos são muito poderosos e são capazes de reconhecer o contexto de uma conversa e de gerar diferentes frases como resposta para uma mesma situação;

3 METODOLOGIA

A metodologia que guiou o desenvolvimento do projeto pode ser dividida em três frentes, todas passando por etapas de levantamento de requisitos, decisões de projeto, implementação e testes: a determinação da arquitetura da plataforma, a definição dos algoritmos de processamento de linguagem natural e a realização de testes.

3.1 Arquitetura da Plataforma

A determinação da arquitetura da plataforma envolve primeiramente a análise dos requisitos de um sistema de atendimento a clientes baseado em *chatbots*, através de questões como facilidade de acesso, simplicidade de uso do sistema, adequação a conceitos modernos de *design* que os usuários já estejam acostumados, e disponibilidade de recursos de acessibilidade. Em seguida, é necessário definir como os diferentes componentes de *software* devem se organizar a fim de concretizar os requisitos, analisando as tecnologias disponíveis para cada componente.

A plataforma dos *chatbots* é dividida em duas frentes: a interface com os usuários (*front-end*), e o servidor (*back-end*), que trata as requisições feitas pelos usuários e implementa os *chatbots*, ambas se comunicando através da *internet* de acordo com um mesmo protocolo.

As interações podem ser vistas no diagrama:

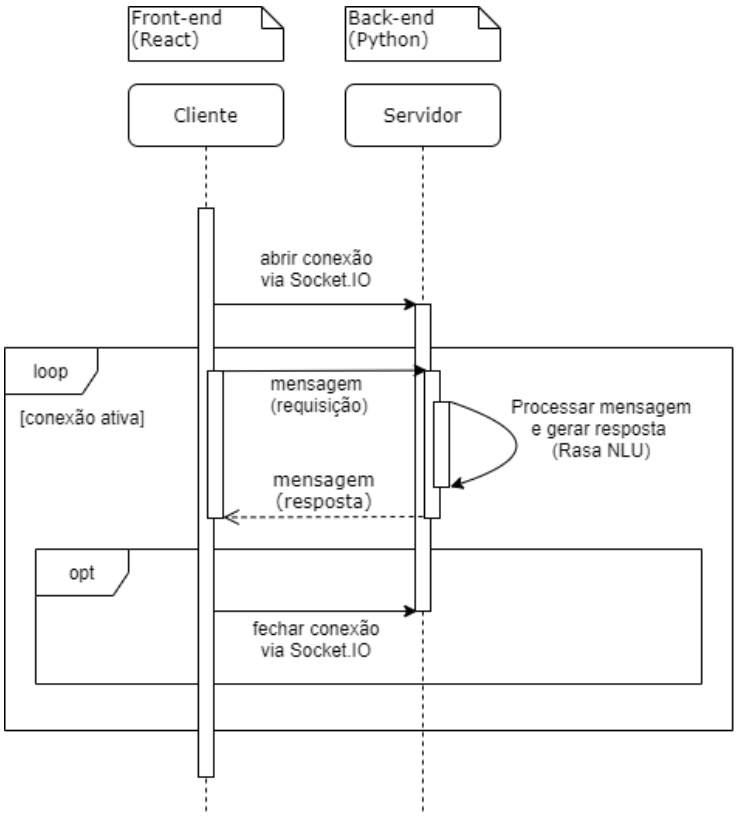


Figura 1: Diagrama de Sequência.

A interação começa com o estabelecimento da conexão gerenciada pela biblioteca **Socket.IO** [12], seguida de uma mensagem do usuário. Esta mensagem é enviada para o *back-end* e, após tratamento, é enviada aos *chatbots*. Uma vez que os *chatbots* tenham gerado uma resposta, esta resposta é enviada de volta para o *front-end*, para que o usuário possa visualizá-la e continuar a interação.

3.2 Estudo de Algoritmos

A definição dos algoritmos passa primeiramente pela escolha da biblioteca de NLU a ser usada como pilar do desenvolvimento do projeto dos *chatbots*. Neste ponto, é preciso analisar as opções disponíveis a partir de pontos relevantes como: código aberto, algoritmos de PNL já implementados e prontos para uso, capacidade de definição de hiper-parâmetros e de modificação nos algoritmos implementados e qualidade dos modelos gerados. Com base nisso, foi encontrada a biblioteca Rasa NLU, uma biblioteca *open source* que possui diversos algoritmos de PNL já implementados e que podem ser facilmente modificados de acordo com as necessidades do projeto.

Em seguida, é necessário realizar a definição dos algoritmos, que se refere à esco-

lha dos componentes que irão compor o *pipeline* de PNL implementado pela biblioteca Rasa NLU para gerar seus modelos. Estes componentes são combinados de forma que as funcionalidades se complementem a ponto de gerar modelos mais complexos capazes de identificar corretamente o que um usuário deseja, extrair informações relevantes de textos de entrada e acionar funcionalidades adequadas, de forma a atender às requisições.

Os componentes podem ser divididos de acordo com as tarefas realizadas e o estudo das técnicas e dos algoritmos pode ser feito partindo dos mais simples e gradualmente aumentando o nível de complexidade, comparando diferentes algoritmos capazes de realizar as tarefas necessárias. Esta comparação entre os diferentes componentes e as diferentes configurações é importante para se definir quais se encaixa melhor no contexto do projeto.

Vale ressaltar que os algoritmos, mesmo dentro de uma mesma categoria, não são necessariamente mutuamente exclusivos. Muitos deles podem e são utilizados em conjunto para se chegar a um resultado apropriado. Em extração de entidade, por exemplo, pode-se utilizar um algoritmo baseado em expressões regulares para extrair informações referentes a documentos, anos ou endereços de e-mail, que possuem formato bem definido, em conjunto com outro algoritmo de extração de entidade mais preparado para obter nomes de pessoas, de cidades ou de meses do ano, que não possuem formato tão definido.

3.3 Planejamento dos Testes

A etapa de testes é realizada após o desenvolvimento do projeto atingir um produto mínimo viável, ponto no qual é possível que usuários do sistema interajam adequadamente tanto com o *front-end* quanto com os *chatbots*.

Os testes são divididos em duas etapas:

- Testes automatizados, implementados pela biblioteca Rasa NLU, em que uma série de entradas é aplicada ao modelo de predição e as saídas são comparadas aos valores esperados, a fim de validar a qualidade dos modelos;
- Testes com usuários reais, etapa na qual o objetivo é analisar o funcionamento do projeto com base na visão e no uso de pessoas não envolvidas no desenvolvimento do *chatbot*. Desta maneira, é possível obter opiniões e avaliações com base em fluxos de conversa não enviesados e que podem divergir em relação aos fluxos inicialmente pensados.

Com estes dois tipos de teste, é possível aprimorar o *chatbot*, removendo erros que

não haviam sido mapeados, deixando-o mais preparado para a utilização em ambientes reais.

3.3.1 Testes Automatizados

Os testes automatizados, ao compararem os valores obtidos com os valores esperados em conversas de teste, possibilitam a geração de gráficos e relatórios analisando métricas de acurácia do modelo. A partir destes testes, é possível indicar quantas entidades foram extraídas corretamente, quantas não foram reconhecidas e quantas foram identificadas de maneira errônea, quantas intenções foram classificadas corretamente, quais foram as entradas que geraram maior dificuldade etc.

Estes testes servem, principalmente, para melhorar os modelos de predição e para entender como alterações nos fluxos esperados de conversa, nos dados de treinamento e nos algoritmos presentes no *pipeline* afetam os resultados do modelo.

3.3.2 Teste em Produção

A aquisição de bons resultados nesta etapa passa pelo planejamento detalhado dos testes, o que inclui considerar como o projeto será disponibilizado, como os dados serão adquiridos e como serão analisados. Primeiramente, define-se que a disponibilização aos usuários de teste deve ser feita através de uma plataforma em nuvem, buscando simplificar a implantação e o acesso. Os possíveis usuários de teste identificados são parentes, amigos pessoais e colegas de classe dos membros envolvidos no projeto.

Durante as interações com o *chatbot*, todo o registro dos diálogos realizados é armazenado em um banco de dados não relacional, que inclui mensagens enviadas e recebidas, intenção identificada em cada mensagem recebida e entidades extraídas das mensagens, possibilitando posterior análise de mensagens e fluxos de conversa que levem a um comportamento inadequado.

Ao mesmo tempo, além de avaliar a adequação do diálogo ao comportamento planejado, também é preciso avaliar a satisfação e a experiência dos usuários com a interação realizada. Para coletar as impressões, ao final de todos os fluxos de conversa é enviado o *link* de um formulário onde os usuários podem preencher seu nome, seu e-mail (opcional), os fluxos de conversa que testou, os fluxos de conversa em que encontrou problemas e suas impressões gerais sobre a aplicação. As respostas deste formulário, aliadas aos registros das conversas, serão utilizadas para chegar a um parecer em relação ao *chatbot*, indicando

quais objetivos foram alcançados e quais não foram. Este formulário será desenvolvido tendo como base algum teste de experiência de usuário encontrado na literatura.

Por fim, é preciso considerar que os usuários de teste do sistema não são os reais clientes da empresa de distribuição de energia elétrica. Por conta disso, estes usuários não possuem contas cadastradas no sistema e solicitar que se cadastrem previamente pode desestimular o uso e causar preocupações com possíveis violações de privacidade e uso de dados pessoais. A solução adotada é gerar um usuário de teste com informações como CPF, CEP e telefone gerados aleatoriamente, e fornecer as credenciais necessárias no início da interação.

4 ESPECIFICAÇÃO

O projeto se trata de uma plataforma *web* que permite ao usuário interagir com um *chatbot* no domínio de uma distribuidora de energia elétrica. Com este objetivo, o projeto é dividido em duas partes: a parte da interface *web* e a parte do *chatbot*.

Para a parte *web* (*front-end*), é importante focar na experiência de usuário interagindo com o sistema. Isto significa criar uma visualização chamativa, moderna, simples, intuitiva e acessível. Um *design* amador ou ultrapassado afasta usuários, já que o visual é a primeira identificação registrada por quem acessa o site. A intuitividade também é importante para que o usuário não fique desmotivado a utilizar o serviço, em função de uma interface complexa demais. Usar elementos visuais de *design* moderno e comuns em aplicações de conversa ajuda a facilitar o uso por parte de usuários. A acessibilidade, por sua vez, é de grande importância, considerando principalmente que o domínio de uma distribuidora de energia elétrica deve ser capaz de atender a toda a população residente no local de atuação da empresa. Com o objetivo de atingir estes requisitos, é preciso utilizar tecnologias e ferramentas modernas para a construção da interface de usuário.

Para a parte do *chatbot* (*back-end*), há dois pontos importantes, que guiam o projeto. O primeiro é relativo à compreensão de texto: como *chatbots* lidam com usuários utilizando linguagem natural, ainda contando com problemas como erros de ortografia, erros de digitação, abreviações e gírias. Assim, é importante implementar e utilizar tecnologias e ferramentas que consigam dar conta do trabalho de processamento utilizando representações e métricas de similaridade capazes de identificar corretamente a intenção do usuário. Atualmente, a maioria destas ferramentas atacam estes problemas utilizando métodos estatísticos e aprendizado de máquina. O segundo ponto é relativo ao domínio da distribuidora de energia elétrica: é de extrema importância levantar corretamente quais são os requisitos funcionais que são encontrados neste contexto. Isto exige um trabalho de pesquisa e busca por ferramentas já existentes e fluxos comuns para esse meio, como por exemplo a emissão de segunda via de contas e identificação de quedas de energia e problemas de transmissão em uma determinada região.

4.1 *Front-end*

O *front-end* da aplicação trata da parte visual do projeto e é através dele que os usuários interagem com o sistema. No caso da plataforma *web*, o *front-end* é apresentado quando o usuário acessa o *site* através de um navegador de internet, tanto por parte de um computador tradicional, quanto através de um dispositivo móvel.

4.1.1 Tecnologias

As tecnologias escolhidas para o desenvolvimento da plataforma *web* que será utilizada pelo cliente foram:

- **Javascript**: linguagem de script interpretada que pode ser utilizada em navegadores para criar páginas dinâmicas com conteúdos atualizados automaticamente. A linguagem permite a realização de tarefas complexas e é suportada por todos os maiores navegadores de internet da atualidade. [13];
- **React**: biblioteca JavaScript de código aberto voltada para o desenvolvimento de interfaces para usuários em formato de página única e que permite geração de páginas estáticas, visando fácil distribuição [14];
- **Material UI**: *framework* baseado em React que implementa conceitos do *design Material*, presente em dispositivos *Android*, para páginas *web* [15];
- **Socket.IO**: biblioteca responsável pelo gerenciamento de conexões e comunicação entre o *front-end* e o *back-end* [12];

4.1.2 Decisões de Projeto

As decisões de projeto para o *front-end* podem ser divididas em três frentes: a de funcionalidades do *front-end*, a de *design* e a de conexão com o *back-end*.

4.1.2.1 Funcionalidades do *front-end*

Em termos de funcionalidades, o *front-end* foi desenvolvido usando a biblioteca **React**, o que permite mostrar todas as mensagens, antigas e novas, recebidas e enviadas, bem como informações relevantes sobre o projeto e sobre o uso do chat, de maneira automática, em tempo real e sem necessidade de recarregamentos na página. A escolha do **React** se deu pelas seguintes razões:

1. Como o objetivo da plataforma é ter a funcionalidade de *chat*, não há necessidade de diversas páginas. O mais adequado é criar uma aplicação de página única (*Single Page Application*), o que é gerado utilizando a biblioteca;

2. As páginas desenvolvidas utilizado o *React* podem ser transformadas em arquivos estáticos, que podem ser facilmente distribuídos de maneira confiável e com baixo tempo de carregamento;

3. Como a aplicação tem tamanho reduzido, não há a necessidade de *frameworks* elaborados, como **Angular** por exemplo. O **React** tem o nível de complexidade ideal pare este tipo de aplicação;

4. **React** é uma tecnologia muito adotada no contexto atual de aplicações *web* e possui boa documentação, o que implica em um fácil acesso a fontes de informações em casos de dúvidas;

5. Os integrantes já tinham experiência com a biblioteca.

4.1.2.2 Design

O *design* foi feito principalmente com base nas classes do *Material Design* [16], muito presente em aplicativos do sistema *Android* e em páginas *web* modernas, através do *framework* para **React** chamada **Material UI**. Este *framework* foi escolhido pois:

1. É amplamente utilizado, possuindo uma documentação extensa;

2. Facilita o desenvolvimento de interfaces responsivas, funcionais tanto em navegadores de *desktops* quanto navegadores para dispositivos móveis;

3. Facilita a criação de componentes complexos, como barras laterais retráteis, por exemplo;

4. Padroniza a identidade visual da aplicação com base no *Material Design*, que já está consolidado em dispositivos *Android* nos últimos anos;

5. Aproxima a interface gráfica a aplicações que os usuários já estão acostumados, considerando-se que o *Android* é o sistema operacional móvel mais utilizado no Brasil [17].

4.1.2.3 Comunicação com o *back-end*

A comunicação com o *back-end* é gerenciada pela biblioteca **Socket.IO**, que é muito popular em ambientes desenvolvidos usando a linguagem JavaScript e apresenta boa documentação e simples acesso a fontes de informações na comunidade de usuários. Esta biblioteca foi escolhida por possibilitar o envio e o recebimento de mensagens em tempo real através de tecnologias de comunicação bidirecional e com bom desempenho, essencial para o bom funcionamento de aplicações do tipo chat.

A biblioteca estabelece conexão usando, sempre que possível, o protocolo *WebSocket* em conjunto com a adição de alguns metadados específicos. Ela também é robusta o suficiente para ser capaz de utilizar outros protocolos de comunicação de maneira transparente ao programador quando se detecta que o *WebSocket* não está disponível, bem como para tratar casos de desconexão e reconexão de usuários. Estes pontos adicionais foram considerados diferenciais para a escolha da biblioteca **Socket.IO** ao invés do uso direto da API nativa do *WebSocket*, que havia sido considerada no início do projeto.

4.1.3 Acessibilidade

Com o intuito de atender a legislação[1] e garantir que o número máximo de usuários consiga utilizar a plataforma, por se tratar de uma aplicação de atendimento público, uma das preocupações durante o desenvolvimento foi garantir a acessibilidade do *site*. A acessibilidade foi verificada a partir de duas ferramentas de análise:

- **WAVE**, da WebAIM, uma organização que trabalha no ramo de acessibilidade na *web* desde 1999 [18];
- **axe**, da Deque [19], uma empresa do ramo de acessibilidade digital desde 1999.

Estas ferramentas verificam, entre outras coisas:

- Se os cabeçalhos presentes no *site* seguem a ordem de importância. Tratando-se de HTML, isto implica em *tags* h1 vindo antes de *tags* h2, não se utilizando uma *tag* h3 e em seguida uma *tag* h5, pulando a *tag* h4, e assim por diante;
- Se as cores do *site* apresentam um contraste suficiente para facilitar a leitura de textos;

[1] Lei Brasileira de Inclusão da Pessoa com Deficiência - Lei 13.146/2015

- Se as diferentes áreas do *site* (cabeçalho, navegação, conteúdo principal etc.), são facilmente distinguíveis;
- Se os ícones, imagens e outros elementos visuais apresentam textos descritivos.

Utilizando estas ferramentas, foi possível determinar os problemas existentes e corrigi-los, de forma que, atualmente, o *site* não conta com problemas críticos de acessibilidade. Segundo as análises feitas, o site desenvolvido já conforma com o padrão AA de acessibilidade da W3C, definido no *Web Content Accessibility Guidelines* (WCAG) 2.0 [20].

4.1.4 Desenvolvimento

O desenvolvimento do *front-end* foi feito utilizando o Git como software de gerenciamento de versões e a plataforma GitHub para hospedagem e compartilhamento do código fonte, facilitando a sincronização do trabalho realizado entre os autores. O repositório contendo todo o código fonte e o histórico e modificações e volução do *front-end* pode ser encontrado em **github.com/BrunoInacio/chat-front-end**.

4.1.5 Resultados

O *front-end* conta com seis telas principais:

- **Início (*Home*)**: expõe sucintamente o contexto do projeto de formatura, o modo de navegação pelo site e avisos importantes com relação à etapa de testes;

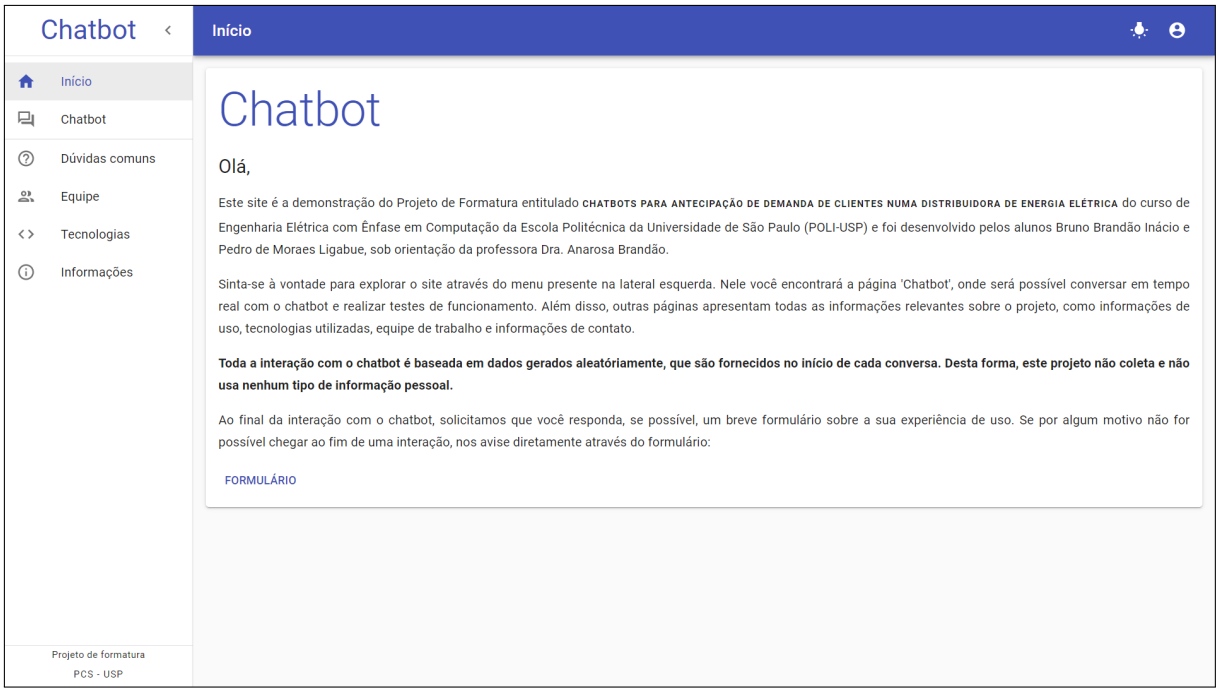


Figura 2: Início (*Home*)

- ***Chat***: expõe a interface de *chat* que será utilizada para a comunicação entre os usuários e os *chatbots*;

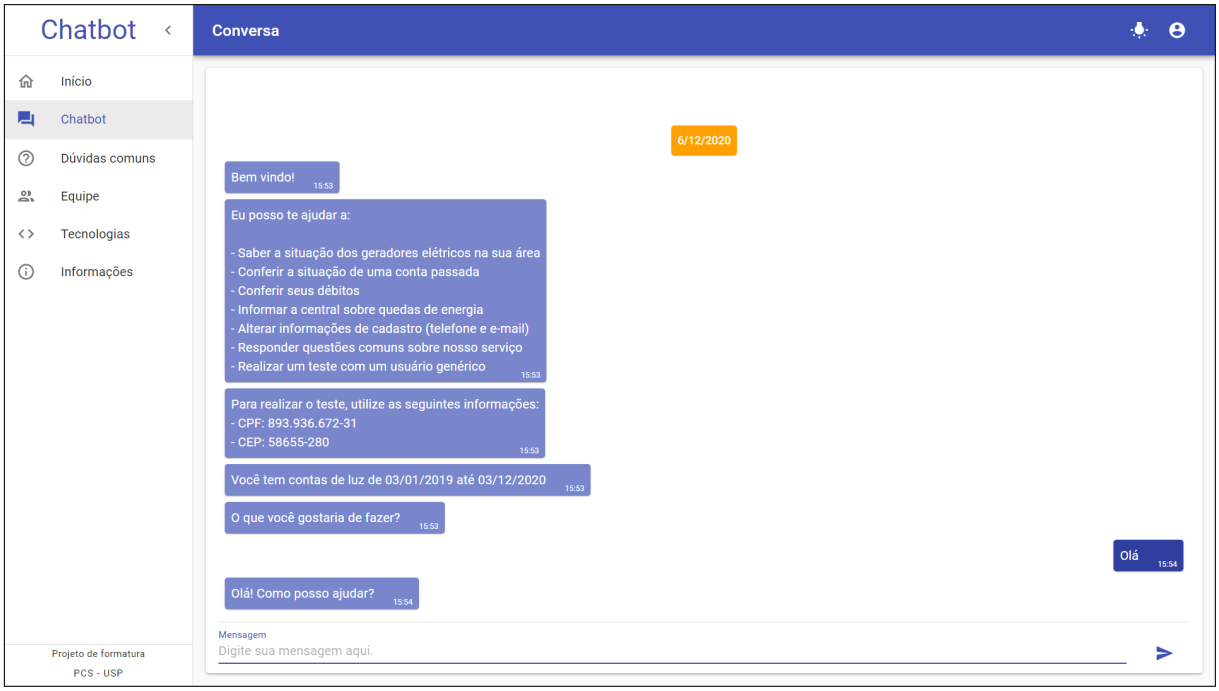


Figura 3: *Chat*

- **Perguntas (FAQ)**: expõe algumas perguntas que podem surgir sobre o projeto de formatura ou sobre o funcionamento dos *chatbots*;

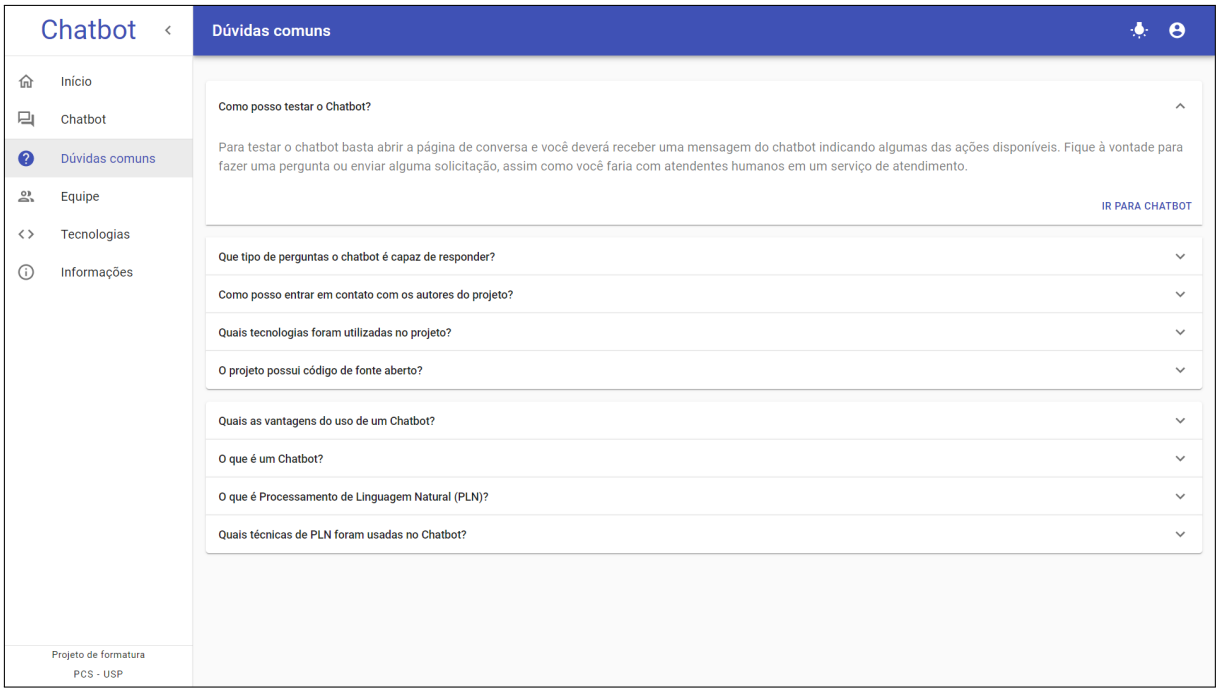


Figura 4: Perguntas (FAQ)

- **Equipe**: expõe informações sobre os membros da equipe: os autores, a orientadora e colaboradores;

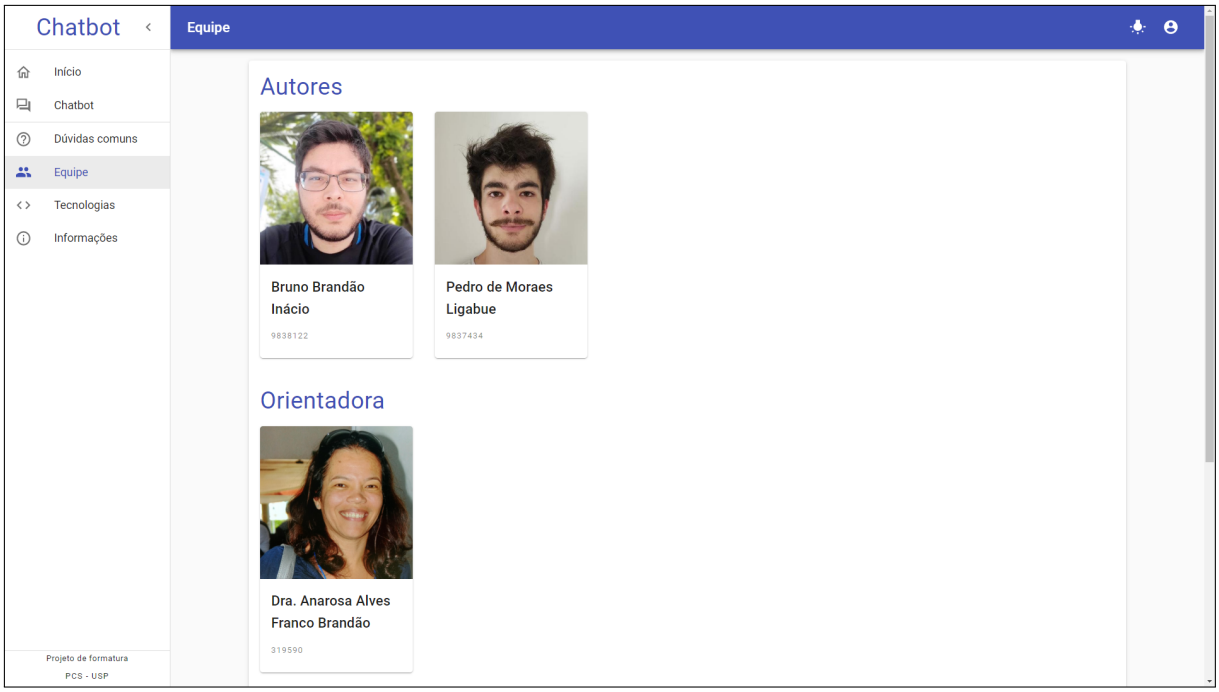


Figura 5: Equipe

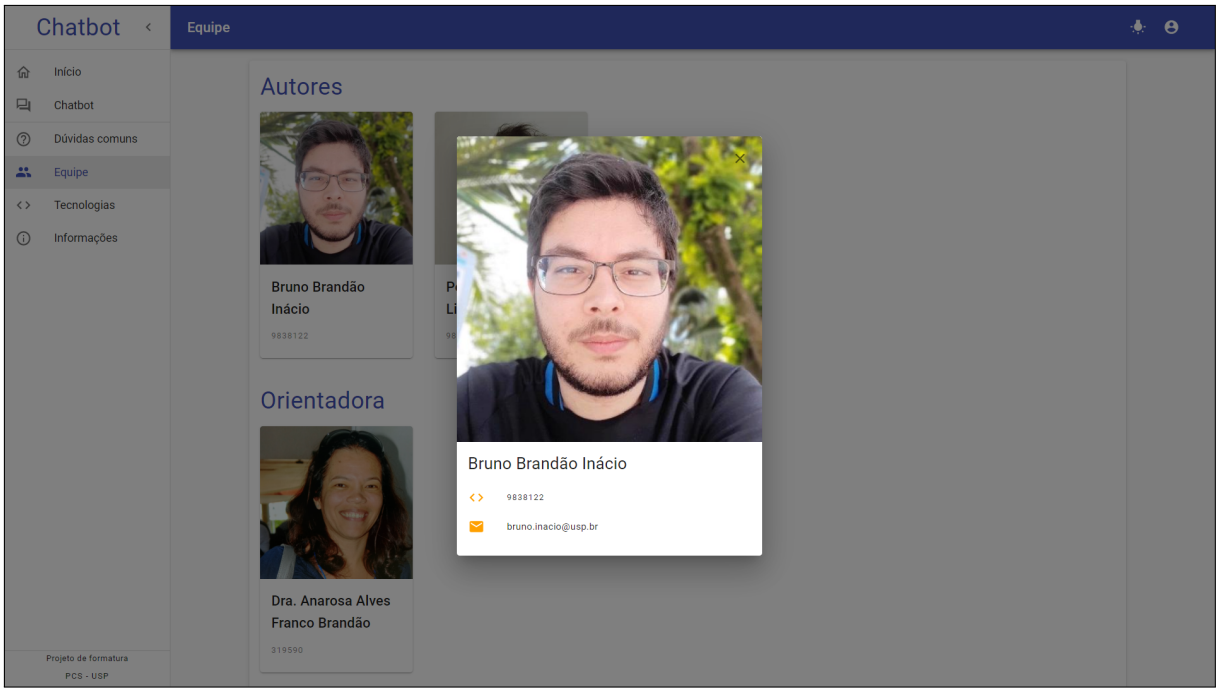


Figura 6: Detalhes expandidos sobre o autor Bruno

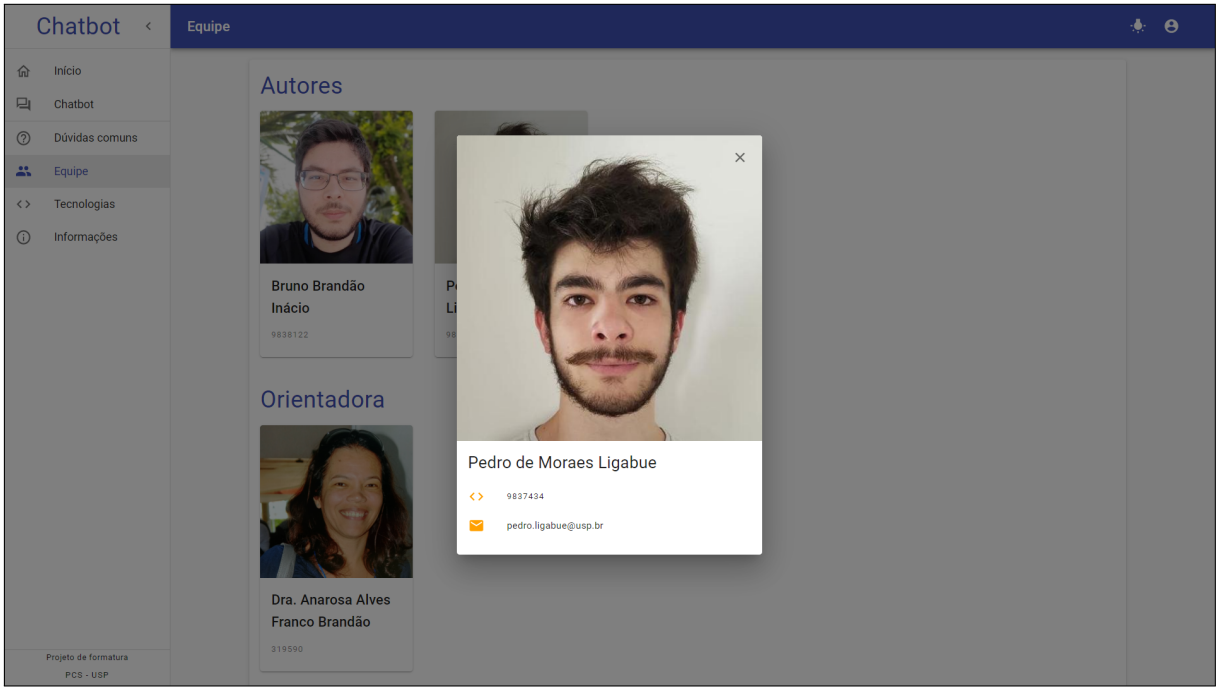


Figura 7: Detalhes expandidos sobre o autor Pedro

- **Tecnologias**: expõe as tecnologias como linguagens de programação e bibliotecas utilizadas para desenvolver a plataforma;

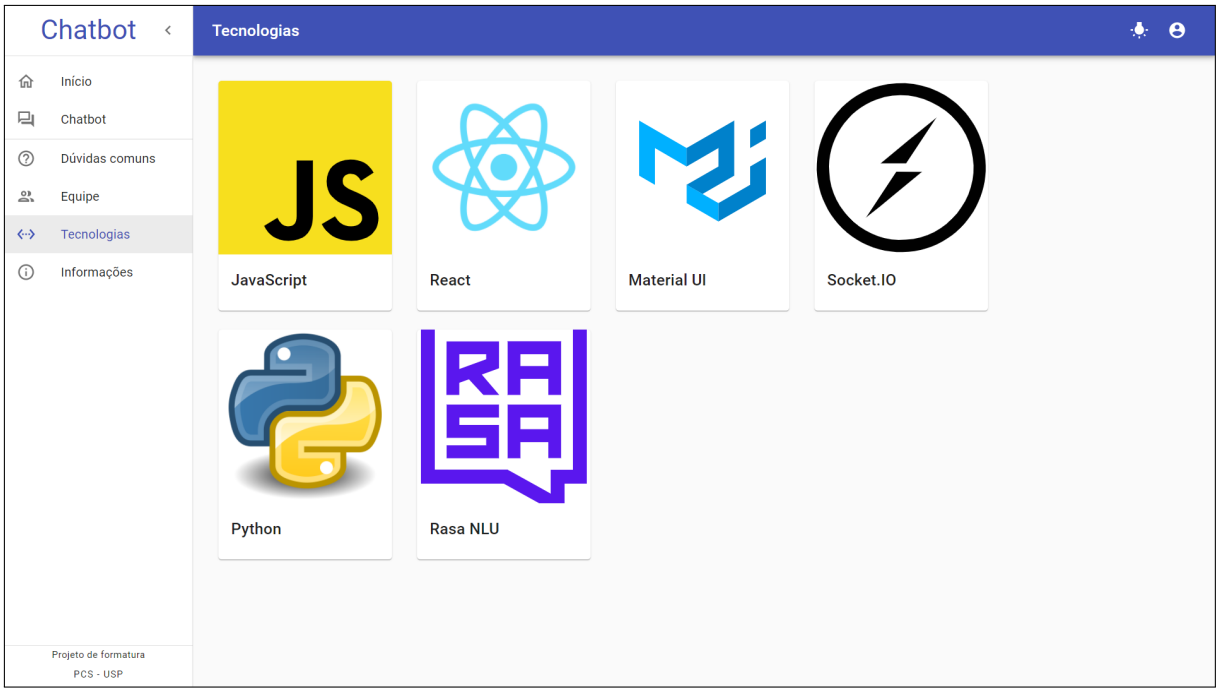


Figura 8: Tecnologias

- **Informações**: contém informações relevantes sobre o projeto de formatura e a plataforma, bem como o endereço de acesso do código de fonte projeto;

Figura 9: Informações

Todas as imagens apresentadas foram obtidas com base na versão do sistema para computadores tradicionais e estão no tema principal. Há também um tema secundário

baseado em cores escuras que pode ser utilizado em todas as páginas e ele possui a seguinte representação:

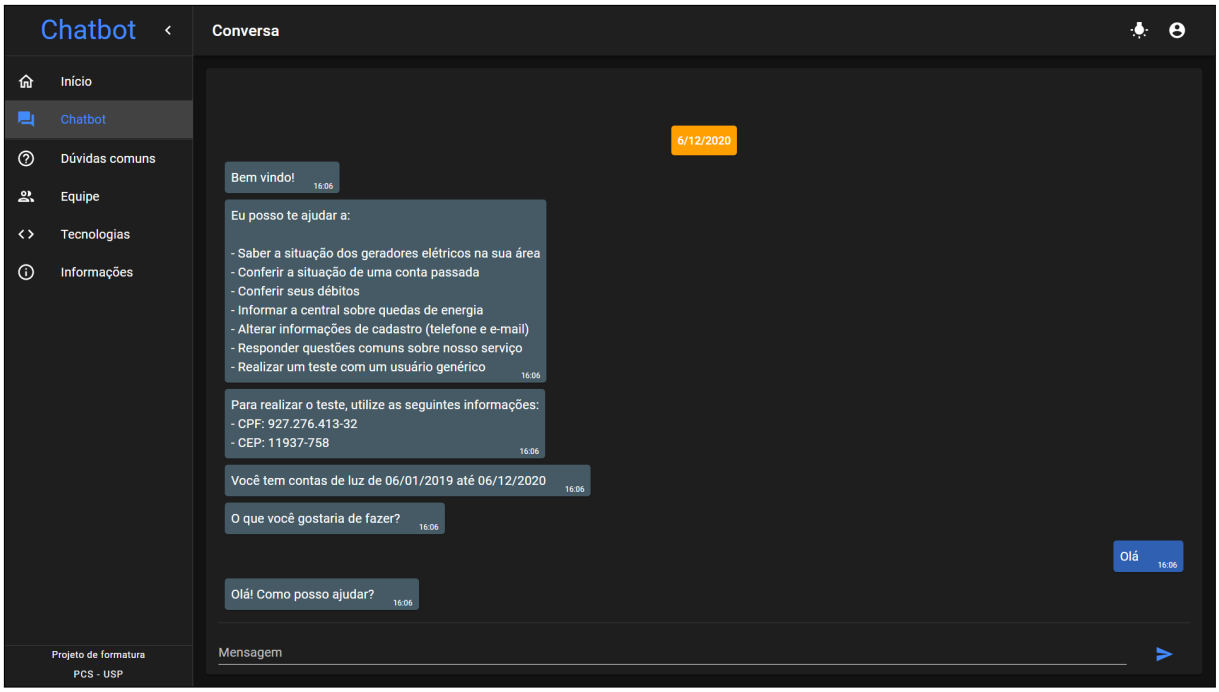
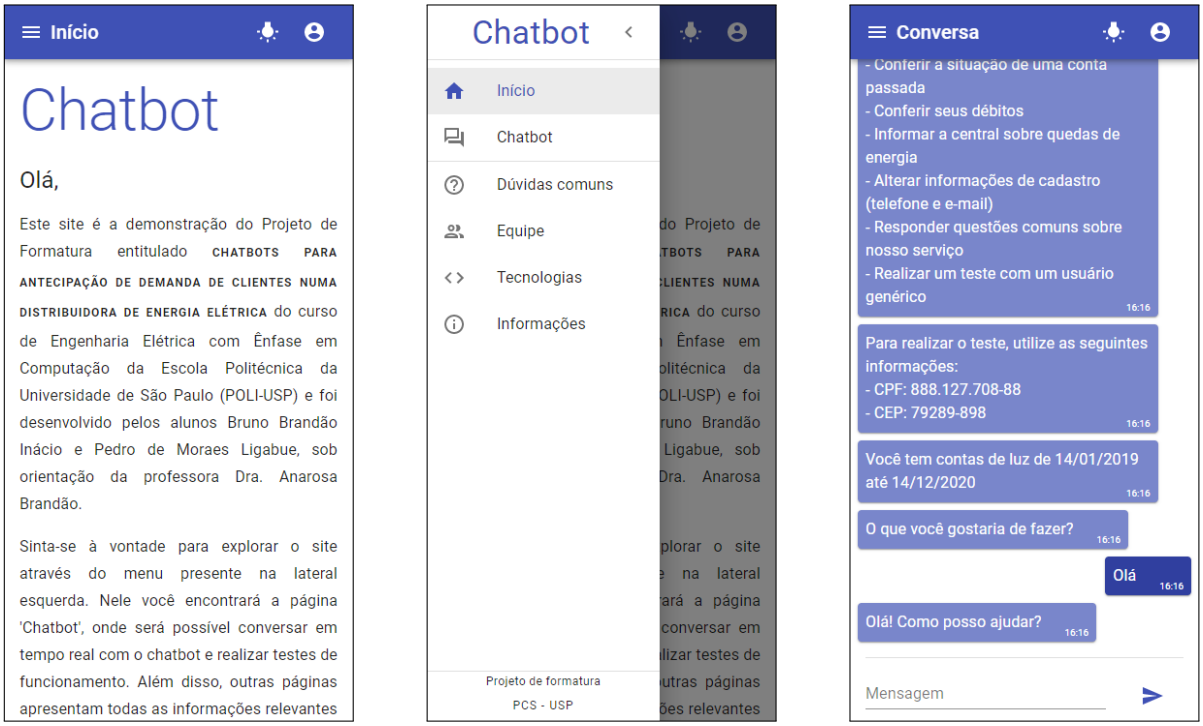


Figura 10: Chat com tema escuro

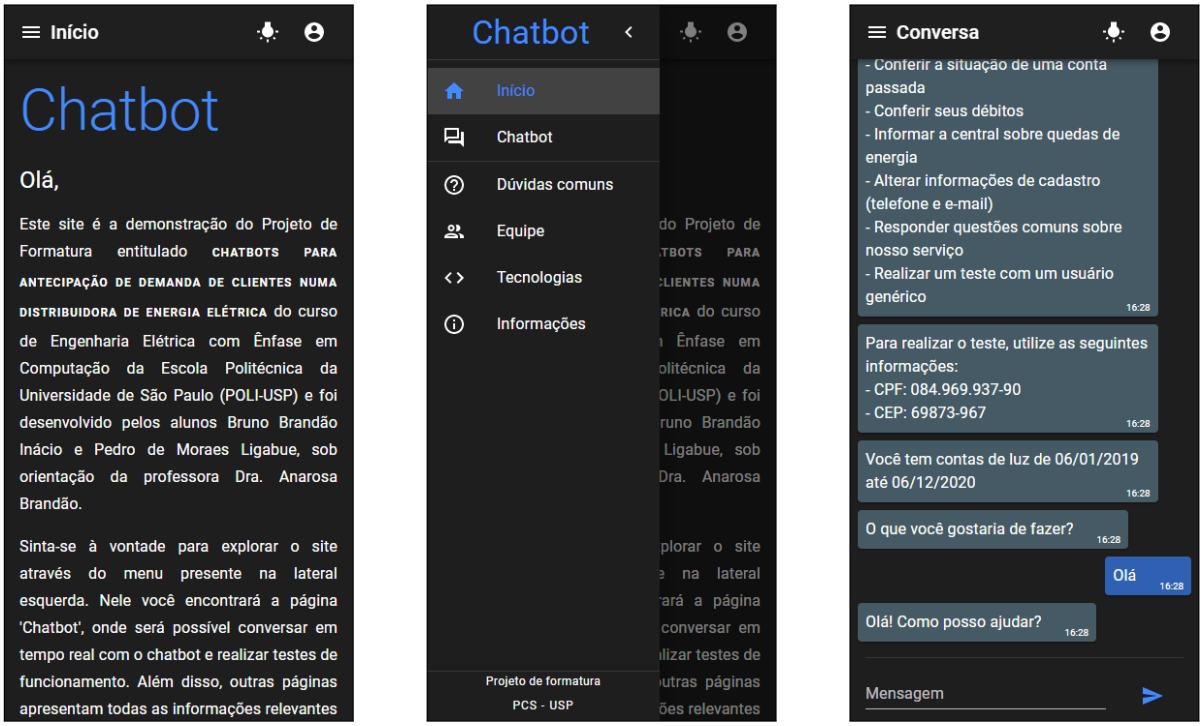
O site também é capaz de se adequar a dispositivos móveis, que possuem telas menores, favorecendo o uso de interfaces sensíveis ao toque. Nestes dispositivos, as informações da barra lateral são exibidas em forma de menu retrátil, que não é visível por padrão para possibilitar a exibição de mais conteúdo das páginas, e que pode ser acessado tanto pelo botão de menu, quanto com um movimento de arrastar a tela partindo da borda esquerda em direção à borda direita.



(a) Página inicial (b) Menu retrátil (c) *Chat*

Figura 11: Visão *mobile*

Todas as telas para dispositivos móveis também possuem uma versão em tema escuro, de acordo com a preferência do usuário. A seguir são mostrados exemplos do tema escuro:



(a) Página inicial (b) Menu retrátil (c) *Chat*

Figura 12: Visão *mobile* no tema escuro

4.2 *Back-end*

O *back-end* é responsável por tratar as requisições feitas pelos usuários através do *front-end*, e pode ser dividido em um serviço que implementa a comunicação com os *chatbots* e outro serviço que realiza ações, como por exemplo consultas ao banco de dados.

4.2.1 Tecnologias

As tecnologias escolhidas para o desenvolvimento do *back-end*, tanto para os *chatbots* quanto para a execução de ações foram:

1. A linguagem de programação **Python** [21];
2. A biblioteca de processamento de linguagem natural **Rasa NLU** [22];
3. O banco de dados não-relacional MongoDB para gerenciar os registros de conversas realizadas [23];
4. O banco de dados relacional SQLite para armazenar dados referentes aos usuários, como documentos e histórico de contas [24].

4.2.2 Decisões de Projeto

As escolhas das principais tecnologias do *back-end* foram definidas com base nos seguintes pontos:

4.2.2.1 Python

A escolha pela linguagem de programação **Python** se deu por três motivos principais:

1. A linguagem já era conhecida pelos integrantes do grupo;
2. É uma linguagem que vem crescendo em popularidade nos últimos anos, especialmente quando se trata de Inteligência Artificial, tanto no mercado quanto na Academia [25];
3. Há uma enorme gama de bibliotecas de aprendizado de máquina, processamento de dados e processamento de linguagem natural desenvolvidas em **Python**, incluindo a que foi escolhida, **Rasa NLU**.

4.2.2.2 Rasa NLU

A escolha pela biblioteca **Rasa NLU** para o desenvolvimento do *chatbot* foi feita pelos seguintes motivos:

1. No trabalho de mestrado que serve como a base deste projeto de formatura, os *chatbots* desenvolvidos para demonstrar as funcionalidades do sistema multiagente foram feitos utilizando a biblioteca **Rasa NLU**. Desta maneira, ao utilizarmos a mesma biblioteca, podemos reaproveitar parte do que já havia sido feito;
2. A biblioteca Rasa NLU já inclui por padrão um canal **Socket.IO**, o que facilita a conexão com o *front-end*. Inicialmente foi considerado utilizar uma biblioteca chamada **websockets** para fazer a conexão, mas esta ideia foi abandonada quando o canal disponibilizado pela Rasa NLU se mostrou mais vantajoso, não só por já estar configurado para estabelecer a conexão, como também por processar as requisições e deixá-las prontas para o processamento de linguagem;
3. A biblioteca cria seus modelos utilizando o conceito de *pipeline* e fornece diversas opções de algoritmos para cada tarefa necessária no processamento de linguagem

natural (ver **2.4**). Isto permite que os componentes, cada um com uma tarefa específica, como por exemplo extração de entidade e classificação de intenção, possam ser combinados e configurados através de hiper-parâmetros, ou substituídos, conforme os requisitos do projeto.

4. Os diferentes componentes apresentados pela biblioteca foram desenvolvidos utilizando outras bibliotecas referência na área de IA e PLN em Python, como **TensorFlow**, **spaCy**, **scikit-learn**.

5. A biblioteca conta com várias ferramentas de auxílio de desenvolvimento. Entre elas estão:

   - Uma linha de comando para conversar com o *chatbot* sem a necessidade de uma interface gráfica desenvolvida separadamente;
   - Uma ferramenta de aprendizado interativo, em que o *chatbot* recebe um *input*, gera sua resposta e o desenvolvedor pode definir se a resposta foi adequada ou não (e em que pontos ela errou);
   - Um comando simples para gerar modelos novos conforme novas história e domínios são adicionados;
   - Uma ferramenta de testes capaz de conferir se as previsões e resultados esperados estão condizentes e gerar um relatório contabilizando os erros e acertos.

6. A biblioteca possui código de fonte aberto e é livre, o que possibilita análises mais profundas sobre os algoritmos já implementados e até mesmo modificações nestes algoritmos a fim de cumprir tarefas específicas;

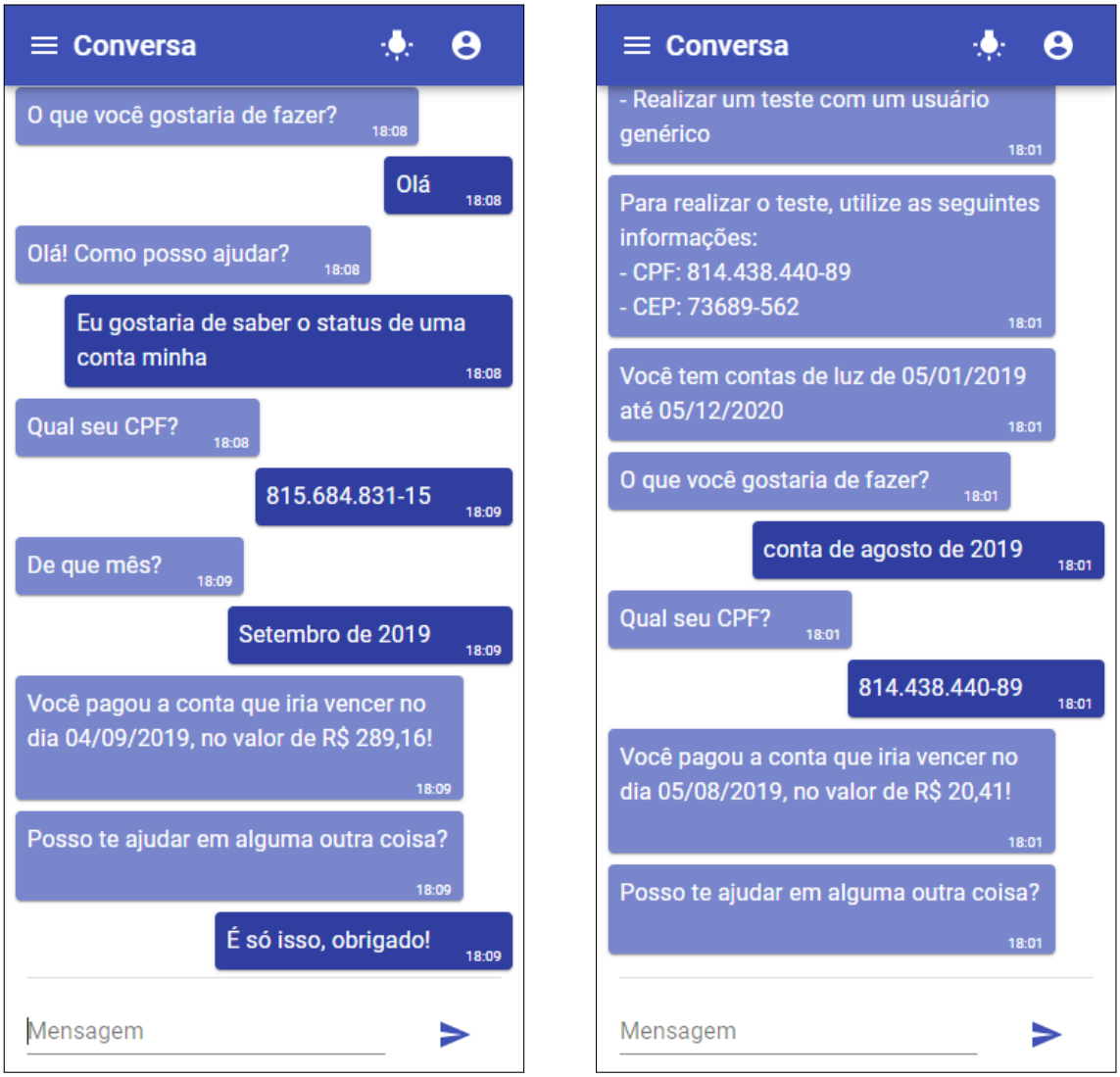
4.2.3 Desenvolvimento

O versionamento do projeto foi feito através do software Git e a plataforma GitHub foi utilizada para hospedar, sincronizar e compartilhar o código fonte entre os autores do projeto. O repositório contendo todo o código fonte do *back-end* pode ser encontrado em **github.com/Pligabue/rasa_chatbot**.

4.2.4 Fluxos de conversa

O *chatbot* conta o seguintes fluxos de interação com os clientes:

- Verificar valor da conta de luz: usuário pede segunda via de uma conta, confirma o CPF e a data e recebe o valor da conta com o status (pago ou não);
- Informação sobre queda de energia: usuário informa que sua região está sem energia, fornece seu CEP e uma ocorrência é criada no banco de dados. Se o problema já for reconhecido, essa informação é repassada ao usuário;
- Consulta de status da região: usuário pede informações sobre a energia na sua região, fornece seu CEP e o *chatbot* responde com o status (se está funcionando ou não e o tempo estimado de reparo, caso não esteja funcionando);
- Perguntas frequentes (FAQ): usuário pode fazer perguntas a respeito do projeto, como por exemplo o que o *chatbot* pode fazer, quem são os autores do projeto e quais as tecnologias utilizadas;
- Cadastro de informações pessoais (e-mail e telefone): usuário informa que quer atualizar uma informação sua, informa o novo valor e o *chatbot* exibe o valor atual e o valor para qual será mudado;
- Consulta de débitos e de faturas pagas: *chatbot* recupera todas as contas não pagas do usuário, exibe as informações de cada uma e mostra o total a ser pago;
- Fluxo para testes: usuário diz que quer realizar um teste e recebe as informações de um usuário de teste gerado na hora.



(a) Exemplo de uma conversa coerente, em que o usuário se comporta como se estivesse falando com um ser humano.

(b) Exemplo de uma conversa menos coerente, em que o usuário diz diretamente o que quer para o *chatbot*.

Figura 13: Exemplos de conversas com o *chatbot*

4.2.5 Pipeline

A escolha dos componentes do *pipeline* foi feita se baseando primeiramente em recomendações presentes na documentação da biblioteca e em seguida foram feitas alterações visando adequação aos requisitos do projeto que foram enumerados. Os componentes atuais são:

1. **WhitespaceTokenizer**: separa o texto em *tokens* com base em *whitespace* (espaço, mudança de linha, tab etc.);

2. **RegexFeaturizer**: verifica se as expressões regulares dos dados de treinamento aparecem na mensagem, a fim de facilitar a identificação de intenções e a extração de entidades que possuem formato bem definido, como CPF, CEP e endereço de e-mail;

3. **LexicalSyntacticFeaturizer**: caracteriza os *tokens* com base em características como a tipografia, os prefixos e sufixos, se é apenas composto por dígitos etc. É utilizado para a extração de entidades;

4. **CountVectorsFeaturizer**: utiliza um modelo *bag-of-words* para criar uma representação das mensagens dos usuários em forma de um vetor de frequência das palavras presentes no texto, que pode ser utilizado para identificar a similaridade entre a mensagem e os exemplos conhecidos durante o treinamento. O algoritmo foi utilizado duas vezes no pipeline, uma para a representação de palavras diretamente e outra para a representação de *n-grams* de caracteres presentes nas palavras, permitindo identificação de combinações parciais;

5. **DIETClassifier**: utilizado para classificação de intenções;

6. **CRFEntityExtractor**: utilizado para extrair entidades presentes nas mensagens, como nomes próprioes e meses do ano;

7. **EntitySynonymMapper**: utilizado para mapear entidades através de sinônimos definidos nos dados de treinamento, como por exemplo diferentes maneiras de se representar um determinado mês do ano;

8. **ResponseSelector**: utilizado para selecionar respostas com base na mensagem do usuário diretamente, verificando as semelhanças entre as duas.

A principal alteração foi a utilização do **CRFEntityExtractor** para a extração de entidades, que por padrão era feita pelo **DIETClassifier**. Esta alteração foi feita pois o **CRFEntityExtractor** consegue aproveitar cadeias encontradas por expressões regulares através do componente **RegexFeaturizer**.

Os testes dos modelos são feitos utilizando a própria ferramenta de testes da biblioteca, na qual é possível observar pontos como quais intenções foram consideradas mais prováveis para uma dada entrada, quais entidades foram identificadas e até mesmo se o modelo é capaz de seguir um fluxo de conversa pré-estabelecido. Estes testes permitem a observação do impacto de mudanças nos algoritmos, nos fluxos pré-estabelecidos e nos dados fornecidos.

4.2.6 Sistema Multiagente

No contexto do projeto, o *chatbot* se encaixa como um agente de um sistema multiagente. Este sistema foi desenvolvido por Henrique Donâncio Nunes Rodrigues, em sua dissertação de mestrado "Avaliação de Escalabilidade e Desempenho da Camada de Transporte de Mensagens em Plataformas Multiagente".

A dissertação trata de analisar diversas implementações de um sistema multiagente, utilizando diferentes *frameworks*, de forma a definir a implementação mais escalável para o uso em uma distribuidora de energia elétrica. Chegou-se à conclusão de que o *framework* **JADE** [26], em Java, era o mais escalável. Esta implementação teve como base as seguintes especificações:

Tabela 1: Especificações do SMA

Sistema Operacional	Windows 10 Pro
Java	Sun JDK 1.8_211
JADE	4.5

A integração com os *chatbots* já estava prevista na dissertação, sendo que a arquitetura do sistema foi idealizada da seguinte maneira:

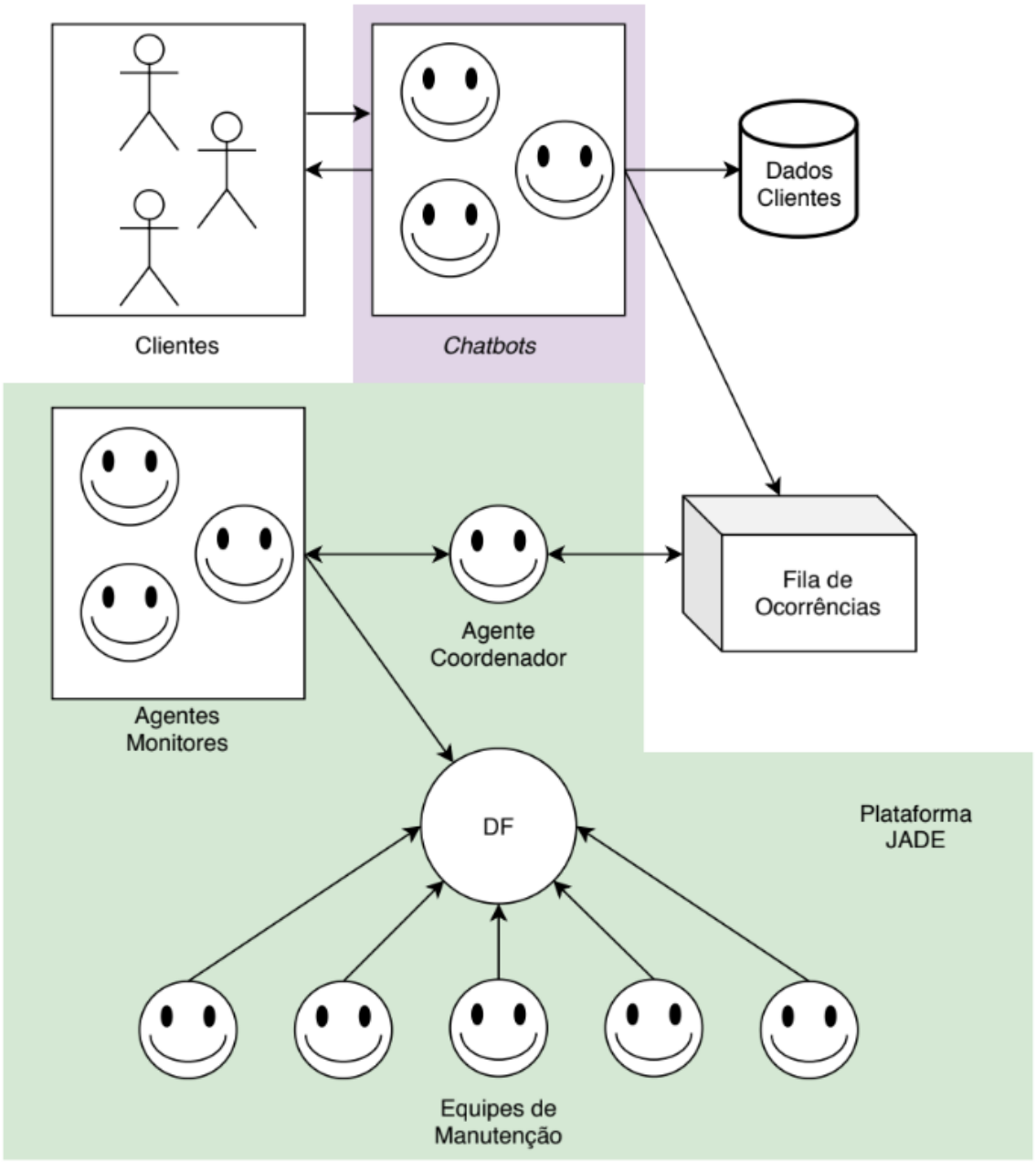


Figura 14: Diagrama da integração dos *chatbots* com o sistema multiagente.
Fonte: Rodrigues, 2019, p. 44 [27]

5 TESTES E AVALIAÇÃO

Uma vez que os fluxos de conversa planejados foram implementados, o projeto passa para a parte de testes. Nesta etapa, o *chatbot* completo é disponibilizado na *web* para que possa ser acessado e testado por pessoas de fora do projeto. Os registros sobre as interações realizadas são armazenadas e também são colhidas pesquisas de opinião com base em um formulário para análise do funcionamento do sistema.

5.1 Implantação do Chatbot

A disponibilização dos *chatbots* para os usuários seguiu a divisão entre *front-end* e *back-end* adotada durante o desenvolvimento e foi feita com base em plataformas em nuvem disponibilizadas pela Amazon AWS [28], com o uso de planos gratuitos fornecidos para estudantes. A separação entre os serviços exigiu a habilitação no *back-end* do compartilhamento de recursos com origens diferentes (CORS - Cross-Origin Resource Sharing) [29], o que foi feito liberando tráfego vindo apenas do endereço reconhecido como *front-end* da aplicação.

O *front-end* desenvolvido com base no *React* pode ser implantado na forma de uma página estática baseada em poucos arquivos, que juntos totalizam um espaço de armazenamento próximo a 2,5 MB, valor baixo para os padrões atuais. Estas características tornam a disponibilização muito simples, pouco custosa e muito eficiente. A plataforma escolhida para isso foi o sistema de armazenamento da AWS, o chamado *Amazon Simple Storage Service (Amazon S3)* [30], que é gerenciado pela própria plataforma e conta com mecanismos de escala automática, de modo a atingir baixo tempo de resposta e alta disponibilidade, sem necessidade de configurações complexas.

O *back-end*, por outro lado, possui requisitos maiores para o bom funcionamento da implantação. O sistema foi desenvolvido em Python e precisa estar em execução durante todo o período de testes, através tanto do serviço de execução de ações quanto do serviço de NLU. O serviço de ações consome em torno de 200MB de memória para ser executado,

enquanto o serviço de NLU sozinho consome 500MB. Além disso, é necessário que todas as sessões de comunicação referentes às conversas em andamento sejam mantidas ativas durante o período da interação. Para atender a estes requisitos, foi utilizado o serviço *Amazon Elastic Compute Cloud (Amazon EC2)* [31] através de uma máquina do tipo *t2.micro*, com 1 núcleo de processamento e 1 GB de memória, executando um sistema Linux em uma distribuição própria da Amazon [32].

5.2 Testes: Interface de Usuário

A partir do *feedbacks* dos usuário em relação à interface da aplicação (*front-end*), foram levantados os seguintes pontos de atenção:

1. Alguns usuários sentiram falta de uma funcionalidade de reenviar as últimas mensagens. Esta funcionalidade se assemelha à de linhas de comando, em que ao pressionar as setas para cima e para baixo, o usuário consegue selecionar um comando feito anterior mente e executá-lo novamente;

2. Alguns usuários gostariam de ter um *placeholder* no momento de preencher dados com formato bem definido, como CPF, CEP, endereço de e-mail e telefone, a fim de evitar o envio de dados no formato incorreto.

Com base nestes pontos, foi possível consolidar novos requisitos funcionais para o *front-end*:

1. Adicionar a funcionalidade de reenviar mensagens anteriores. Isto seria relativamente trivial, levando em consideração que toda lista de mensagens é mantida em um vetor pelo *front-end*. Desta maneira, bastaria a aplicação percorrer este vetor através do uso das setas do teclado e preencher o *input* da mensagem com o texto selecionado;

2. Adicionar metadados às mensagens do *back-end* de forma a informar que tipo de resposta está sendo esperada do usuário. Desta maneira, quando o *back-end* estiver esperando um CPF, por exemplo, o *front-end* seria capaz de exibir o formato esperado para o usuário e validá-lo antes mesmo de enviar os dados para o *back-end*, agilizando a conversa com o usuário.

5.3 Testes: Conversa com o Chatbot

Os testes de usuário revelaram os seguintes pontos fracos do *chatbot*:

1. Quando o usuário se recusava a fornecer um dado em algum dos fluxos de conversa, como por exemplo o CPF, no fluxo de segunda via de uma conta, o *chatbot* entrava em um *loop* infinito em que ele tentava interpretar a entrada do usuário sob o contexto do dado requisitado, mas não conseguia retirar nenhuma informação, apenas dizendo que não havia entendido e pedindo para o usuário repetir, impossibilitando que o usuário realizasse outros fluxos;

2. Quando o usuário tentava trocar de fluxo de conversa, como por exemplo pedindo para ver uma conta sem antes fornecer as informações para trocar o telefone, o *chatbot* não estava sendo consistente na desativação de um fluxo e a ativação do próximo, entrando, em certas ocasiões, em um *loop* infinito em que tentava executar os dois fluxos simultaneamente;

3. Entidades numéricas, telefones e CPFs especialmente, estavam sendo interpretados de maneira incorreta. Isto ocorria com maior frequência no formulário para a troca do telefone cadastrado, quando o usuário não espaçava os dígitos do telefone, escrevendo “5511912345678” no lugar de “55 11 91234-5678”, por exemplo;

4. O *chatbot* não conseguia tratar algumas intenções em contextos específicos que não haviam sido mapeados. Por exemplo, se o usuário simplesmente dissesse ”não”, sem ter sido perguntado nada, o *chatbot*, iniciava um dos fluxos erroneamente;

5. O usuário tentava atualizar dados que não poderia, como o CEP ou o CPF, já que este fluxo só foi disponibilizado para o e-mail e o telefone;

6. O *chatbot* interpretava algumas palavras desconhecidas de maneira equivocada. Por exemplo, ele interpretou “adiós” como o usuário querendo atualizar dados cadastrais.

Em resposta a estes erros e inconsistências, foram feitas as seguintes alterações no *chatbot*:

1. Quando o usuário se recusa a fornecer um dado em um fluxo de conversa, o *chatbot* pergunta para o usuário se ele quer continuar ou não com a interação. O usuário pode optar por encerrar o fluxo, levando o *chatbot* a perguntar ao usuário o que ele deseja fazer em seguida;

2. Foram mapeados novos caminhos de quebra de fluxo de conversa. Desta maneira, passa a ser possível interromper um fluxo e mudar para outro sem que o *chatbot* perca o contexto da conversa. Ele consegue desativar o fluxo anterior sem levantar erros e seguir com o próximo;

3. O erro na identificação de entidade foi mitigado adicionando um corpus maior de CPFs e telefones, facilitando a identificação de entidades de acordo com o contexto, bem como através da definição de expressões regulares para encontrar apenas sequências exatas de caracteres.

4. Novos fluxos de conversa foram criados para intenções sem contexto, de forma que, se o usuário diz "não" sem motivo, o *chatbot* responde perguntando o que o usuário deseja fazer;

5. Foram criadas novas intenções de pedir alterações no CPF e no CEP. Estes pedidos são respondidos com um aviso que o *chatbot* pode apenas alterar o e-mail e o telefone.

Por fim, foram levantados novos requisitos funcionais para o *chatbot*, com base nas necessidades que os usuários demonstraram ter:

1. Fazer alterações em dados cadastrais mais críticos, como CEP. Antes de poder implementar esta funcionalidade, seria necessário adicionar uma validação de usuário mais robusta para o *chatbot*. Isto poderia ser alcançado com uma autenticação em dois passos, em que o usuário valida quem é através de algo que **sabe**, seu CPF e nome, por exemplo, e através de algo que **possui**, recebendo um *token* na sua caixa de e-mail ou por mensagem no telefone;

2. Fazer validações dos dados a serem alterados antes de efetivamente fazer a mudança no banco. Isto poderia ser feito enviando um *token* para o e-mail que o usuário quer usar ou para o telefone novo do usuário. Além disso, poderiam ser utilizadas bibliotecas de validação de CPFs, CEPs e e-mail, de maneira semelhante ao que é feito com telefones através da biblioteca **phonenumbers**.

Estes requisitos não puderam ser concretizados durante o período do projeto. Sendo assim, eles são levantados apenas como sugestões de encaminhamento, caso este trabalho seja continuado no futuro.

6 CONSIDERAÇÕES FINAIS

O objetivo do projeto de formatura tinha como objetivo desenvolver um *chatbot* para atender demandas de clientes no domínio específico de uma distribuidora de energia elétrica, funcionando como um SAC automatizado. Tendo em vista os resultados obtidos através dos testes com usuários, é possível dizer que o objetivo foi atingido, embora com ressalvas importantes.

6.1 Conclusões do Projeto de Formatura

Primeiramente, em relação ao *front-end*, foi possível criar uma interface simples, intuitiva e moderna, que passou no mínimo com uma avaliação AA nos testes de acessibilidade disponíveis *online*, o que está de acordo com o exigido por lei. Além disso, foi possível implementar uma interface de *chat* dinâmica, de maneira semelhante a aplicativos de *chat* comuns, como WhatsApp e Telegram.

Em relação ao *back-end*, foi possível desenvolver um *chatbot* capaz de consistentemente compreender as intenções dos usuários e as entidades incluídas nas mensagens. Em relação ao que tinha sido considerado no início do projeto, não foi possível incluir medidas de segurança maiores para a verificação de usuário, o que seria mandatório em um sistema de uma empresa real. Além disso, por apenas simular o sistema de uma distribuidora de energia elétrica, não foi possível criar fluxos de conversa que dependeriam de ter dados reais, como, por exemplo, passar informações ao vivo da equipe de manutenção, ou enviar mensagens de texto para todos o usuário de uma área afetada por problemas de distribuição, informando-os de que a queda de energia foi reconhecida e estimando prazos para o retorno. Por fim, não foi possível criar tantos fluxos de conversa quanto havia sido pensado, uma vez que cada fluxo adicionado aumentava o nível de complexidade de todos os fluxos anteriores, de forma a evitar conflitos entre eles.

6.2 Perspectivas de Continuidade

Ao mesmo tempo que o *chatbot* foi desenvolvido com sucesso, há ainda um série de funcionalidades a serem adicionadas para que o produto possa ser utilizado no contexto de uma distribuidora de energia elétrica real, como mais fluxos de interação com usuários, integração com outros sistemas da empresa e integração com sistemas externos de envio de mensagens.

A realização de integração com outros serviços possibilitaria, por exemplo, reportar regiões com falhas de distribuições para análises de regiões com maior reincidência, enviar notificações para equipes de manutenção, enviar avisos com a previsão de conserto para a população de áreas afetadas e solicitar verificações de segurança antes de realizar alterações em dados cadastrais.

Outro ponto de continuidade muito importante também seria a integração entre o *chatbot* desenvolvido e o sistema multiagente utilizado como base para o tema deste projeto, que também não foi concretizada.

REFERÊNCIAS

[1] SAS INSTITUTE INC. *What is Natural Language Processing? — SAS*. Disponível em: <https://www.sas.com/en$_u$s/*insights*/*analytics*/*what*−*is*−*natural*−*language*−*processing* − *nlp.html*¿.*Acessoem* : 12*abr*.2020.

[2] FACEBOOK INC. *A state-of-the-art open source chatbot*. Disponível em: <https://ai.facebook.com/blog/state-of-the-art-open-source-chatbot>. Acesso em: 18 out. 2020.

[3] GOOGLE LLC. *Towards a Conversational Agent that Can Chat About... Anything*. Disponível em: <https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html>. Acesso em: 18 out. 2020.

[4] FORBES MEDIA LLC. *How Chatbots Will Transform Customer Experience: An Infographic*. Disponível em: <https://www.forbes.com/sites/blakemorgan/2017/03/21/how-chatbots-will-transform-customer-experience-an-infographic>. Acesso em: 17 maio 2020.

[5] RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. 3. ed. [S.l.]: Prentice Hall, 2010. ISBN 0-13-461099-7.

[6] IBM. *What is Machine Learning?* Disponível em: <https://www.ibm.com/cloud/learn/machine-learning>. Acesso em: 07 dec. 2020.

[7] SAS INSTITUTE INC. *Machine Learning - What it is and why it matters*. Disponível em: <https://www.sas.com/en$_u$s/*insights*/*analytics*/*machine* − *learning.html*¿.*Acessoem* : 18*maio*2020.

[8] CHOPRA, A. K.; SINGH, M. P. Agent communication. In: *Multiagent Systems*. 2. ed. Cambridge, Massachusetts: MIT Press, 2012. cap. 3, p. 101–141.

[9] ORACLE. *What is a Chatbot — Oracle*. Disponível em: <https://www.oracle.com/chatbots/what-is-a-chatbot/>. Acesso em: 12 abr. 2020.

[10] INDURKHYA, N.; DAMERAU, F. J. *Handbook of Natural Language Processing*. 2. ed. 6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742: Chapman Hall/CRC, 2010.

[11] MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. [S.l.]: Cambridge University Press, 2009. Original publicado em 1999. ISBN 1139472100.

[12] SOCKET IO. *What Socket.IO is*. Disponível em: <https://socket.io/docs/>. Acesso em: 20 jun. 2020.

[13] MOZILLA. *Referência JavaScript - JavaScript — MDN*. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference>. Acesso em: 11 abr. 2020.

[14] REACT. *React - A JavaScript library for building user interfaces*. Disponível em: <https://reactjs.org>. Acesso em: 11 abr. 2020.

[15] MATERIAL-UI. *Material-UI: A popular React UI framework*. Disponível em: <https://material-ui.com/>. Acesso em: 11 abr. 2020.

[16] GOOGLE LLC. *Design - Material Design*. Disponível em: <https://material.io/design>. Acesso em: 11 abr. 2020.

[17] BAIN COMPANY. *Impacto econômico e social do Android no Brasil*. Disponível em: <https://www.bain.com/contentassets/20e0815cfd784b4a8dead63475b42380/v02$_b$ − $impactos - do - android - no - brasil.pdf$¿.$Acessoem$: 06$dec.$2020.

[18] WAVE Web Accessibility Evaluation Tool. Disponível em: <https://wave.webaim.org/>. Acesso em: 21 jun. 2020.

[19] AXE - The Standard in Accessibility Testing. Disponível em: <https://www.deque.com/axe/>. Acesso em: 21 jun. 2020.

[20] WEB Content Accessibility Guidelines (WCAG) 2.0. Disponível em: <https://www.w3.org/TR/WCAG20/>. Acesso em: 21 jun. 2020.

[21] PYTHON SOFTWARE FOUNDATION. *About Python — Python.org*. Disponível em: <https://www.python.org/about/>. Acesso em: 11 abr. 2020.

[22] RASA TECHNOLOGIES. *Rasa NLU: Language Understanding for Chatbots and AI assistants*. Disponível em: <https://rasa.com/docs/rasa/nlu/about/>. Acesso em: 11 abr. 2020.

[23] MONGODB, INC. *What is MongoDB?* Disponível em: <https://www.mongodb.com/what-is-mongodb>. Acesso em: 6 dec. 2020.

[24] SQLITE CONSORTIUM. *About SQLite*. Disponível em: <https://www.sqlite.org/about.html>. Acesso em: 6 dec. 2020.

[25] PYPL PopularitY of Programming Language. Disponível em: <http://pypl.github.io/PYPL.html>. Acesso em: 22 jun. 2020.

[26] JAVA Agent Development Framework. Disponível em: <https://jade.tilab.com/>. Acesso em: 26 maio 2020.

[27] RODRIGUES, H. D. N. *"Avaliação de Escalabilidade e Desempenho da Camada de Transporte de Mensagens em Plataformas Multiagente"*. Dissertação (Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo, https://teses.usp.br/teses/disponiveis/45/45134/tde-07102019-150704/pt-br.php, 2019.

[28] AMAZON WEB SERVICES, INC. *What is AWS*. Disponível em: <https://aws.amazon.com/what-is-aws>. Acesso em: 23 out. 2020.

[29] MOZILLA. *Cross-Origin Resource Sharing (CORS)*. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Controle$_{A}cesso_{C}ORS$¿.$Acessoem$: 6dec.2020.

[30] AMAZON WEB SERVICES, INC. *Amazon S3*. Disponível em: <https://aws.amazon.com/s3>. Acesso em: 23 out. 2020.

[31] AMAZON WEB SERVICES, INC. *Amazon EC2*. Disponível em: <https://aws.amazon.com/ec2>. Acesso em: 23 out. 2020.

[32] AMAZON WEB SERVICES, INC. *Amazon Linux 2*. Disponível em: <https://aws.amazon.com/amazon-linux-2/>. Acesso em: 6 dec. 2020.