UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MURILO RODRIGUES PADILHA LEITE

DESENVOLVIMENTO DE ASSISTENTE VIRTUAL PARA ATENDIMENTO IMOBILIÁRIO

Florianópolis
2019

MURILO RODRIGUES PADILHA LEITE

DESENVOLVIMENTO DE ASSISTENTE VIRTUAL PARA ATENDIMENTO IMOBILIÁRIO

Trabalho Conclusão do Curso de Graduação em Engenharia de Controle e Automação do Centro de Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia de Controle e Automação.
Orientador: Prof., Dr., Carlos Montez
Orientador na Empresa: Ricardo Ventura

Florianópolis
2019

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.
Orientações em:
http://portalbu.ufsc.br/ficha

Murilo Rodrigues Padilha Leite

DESENVOLVIMENTO DE ASSISTENTE VIRTUAL PARA ATENDIMENTO IMOBILIÁRIO

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenharia de Controle e Automação" e aprovado em sua forma final pelo Curso Engenharia de Controle e Automação

Local, 12 de Dezembro de 2019.

_____

Prof., Dr., Hector Bessa Silveira
Coordenador do Curso

Banca Examinadora:

_____

Prof.(a), Dr.(a) Carlos Barros Montez
Orientador(a)
Instituição UFSC

_____

Prof., Dr., Rodrigo Tacla Saad
Avaliador(a)
Instituição UFSC

RESUMO

Este documento trata do projeto e desenvolvimento de um chatbot destinado para o atendimento de clientes da imobiliária Brognoli. São apresentadas informações de pesquisa sobre o chatbots e assistentes virtuais, bem como informações sobre as ferramentas utilizadas no desenvolvimento do produto. O documento ainda contém uma explicação minuciosa das lógicas, técnicas e ideias utilizadas no desenvolvimento, bem como a sua aplicação. O chatbot foi desenvolvido principalmente pensando na solução de problemas mais comuns do cliente e entregar serviços automáticos e responsivos, neste documento é apresentada uma análise dos resultados da implementação de dois módulos, o de agendamento de visitas e do de abertura de chamados. Por fim temos as perspectivas do futuro do projeto e de como decorrerá o seu crescimento e desenvolvimento perante aos resultados obtidos.

Palavras-chave: Chatbot, Assistente Virtual, Atendimento ao Cliente, Integrações de Sistemas.

ABSTRACT

This document deals with the design and development of a chatbot intended for customer service of real estate Brognoli. Research information about chatbots and virtual assistants is presented, as well as information about the tools used in product development. The document also contains a thorough explanation of the logic, techniques and ideas used in development, as well as their application. Chatbot was developed primarily with the purpose of solving the most common customer problems and delivering automatic and responsive services. In this document we present an analysis of the results of the implementation of two modules, the visit scheduling and the call opening. Finally we have the prospects for the future of the project and how its growth and development will take place in view of the results obtained.

Key-words: Chatbot, Virtual Assistant, Customer Service, System Integrations.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AIML	Artificial Intelligence Markup Language
API	Interface de Programação de Aplicativos
App	Aplicativo
AV	Assistente Virtual
CRM	Customer Relationship Management
CS	Customer Success
F.A. Q	*Frequently Asked Questions*
HTTP	HyperText Transfer Protocol
IA	Inteligência Artificial
ID	Identificação
NLP	Neuro-Linguistic Programming
NPR	Núcleo do Proprietário
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
RP	Relações Públicas
UX	User Experience
XML	*Xtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 PROBLEMA

O projeto tem como objetivo o desenvolvimento de um Assistente Virtual (AV) para clientes da imobiliária com a finalidade de tratamento de assuntos de pós- venda, para que o AV atue como um *chatbot* a fim de responder perguntas frequentes dos clientes de maneira correta e rápida, como um assistente pessoal que forneça serviços e execute ações importantes para o cliente. Além do tratamento com o cliente já vinculado a imobiliária, o objetivo é o tratamento com possíveis clientes, os chamados *leads,* que são pessoas com um interesse em um dos produtos da imobiliária. A plataforma utilizada para a criação do AV é a blip.ai, que se trata de um ambiente de criação, integração e monitoramento de *chatbots.* A plataforma traz a possibilidade da integração com diversos outros sistemas pertinentes para os serviços que são desejados dentro do AV, como a integração direta com o CRM interno da empresa, o banco de dados e o sistema financeiro.

Os princípios básicos e diagnósticos relacionados a este projeto são as necessidades de respostas mais rápidas e precisas para os clientes de um modo geral, levando em conta a ótica da digitalização do setor imobiliário e a necessidade natural de se automatizar processos repetitivos e bem estabelecidos.

1.2 OBJETIVO GERAL

O objetivo é desenvolver um canal de contato inteligente utilizando o *chatbot* e o atendimento humano, tratando de diversos assuntos durante toda a jornada do cliente, desde o pré-venda com o atendimento dos potenciais clientes e dos atuais clientes da imobiliária (locadores e locatários).

1.3 OBJETIVOS ESPECÍFICOS

Como orientação do projeto, estão definidos os seguintes objetivos específicos:

- Realizar pesquisas em materiais bibliográficos que tenham escopo em assistentes virtuais e integração de sistemas;
- Levantamento de dados internos de funcionalidades e base de conhecimentos relacionados com as dúvidas e dores frequentes dos clientes da imobiliária;
- Estudar a plataforma de criação do assistente virtual blip.ai.;
- Captar informações para atuar no processo comercial;
- Autoatendimento para os clientes poderem abrir solicitações para o *backoffice*;
- Passar informações sobre serviços prestados;
- Gerar e dar acesso a informações financeiras;
- Agendamento de visitas e vistorias;
- Captação de *feedbacks* mais detalhados de clientes e atendentes;
- Design de conversas Humanizadas.

2 CONTEXTUALIZAÇÃO E METODOLOGIA DA PESQUISA

É importante que, antes que se descreva a metodologia-base pela qual o projeto foi desenvolvido, se dê uma contextualização mais aprofundada do porquê desse projeto.

2.1 CONTEXTUALIZAÇÃO

A Brognoli Negócios Imobiliários atua em toda a Grande Florianópolis, com oito agências interligadas direcionadas a aluguéis, vendas e corretagem de seguros. Também conta com uma estrutura de assessoria jurídica e manutenção e reforma de imóveis. Uma rede de soluções imobiliárias completa, com serviços rápidos e inteligentes que facilitam a vida das pessoas. *O chatbot* desenvolvido mostrou-se bastante efetivo para responder às perguntas frequentes feitas pelos clientes da empresa. Devido ao bom resultado, sugeriu-se que, além de somente responder a padrões de perguntas, que houvesse a possibilidade de executar tarefas e acessar serviços externos poderiam trazer vantagens no comportamento e inteligência do mesmo, possibilitando melhor assistência aos usuários e caracterizando-o como um assistente virtual. Inicialmente propôs-se desenvolver um assistente virtual na forma de assistente pessoal exclusivamente dentro do contexto das necessidades de clientes e funcionários. O assistente virtual teria acesso a diversos serviços web da Brognoli, podendo executar diversas funções através dos mesmos.

2.2 METODOLOGIA DA PESQUISA

Gil (2010) explica que o processo de pesquisa deve ser formal e sistemático com o objetivo específico de encontrar soluções para problemas utilizando ferramentas e procedimentos científicos. O processo é configurado em diferentes fases, passando pela identificação e estruturação do problema, até a apresentação dos resultados e discussão dos mesmos.

Segundo Demo (1996), a pesquisa é uma atividade cotidiana, um hábito, um questionar sistemático crítico e criativo, juntamente com atuação pertinente na realidade permeado com a discussão no sentido teórico e prático.

Design Science (JÄRVINEN, 2004) foi o método de pesquisa utilizado no projeto. O método foi recomendado pelo orientador do projeto na empresa, que já o tinha utilizado para um projeto semelhante e considerava adequado para a situação. A definição do *Design Science*, segundo Järvinen (2007), é a de desenvolver uma solução qualquer para sanar ou aprimorar um problema previamente estudado e documentado. Essa técnica é baseada em uma metodologia progressiva e cíclica de utilização de conhecimentos e teorias já comprovadas para trazer uma maior qualidade para o que está sendo desenvolvido e avaliar o seu progresso.

2.3 PROCEDIMENTOS GERAIS PARA DESENVOLVIMENTO DO ASSISTENTE VIRTUAL

Essa seção do capítulo dois se refere ao direcionamento utilizado na realização das etapas deste trabalho.

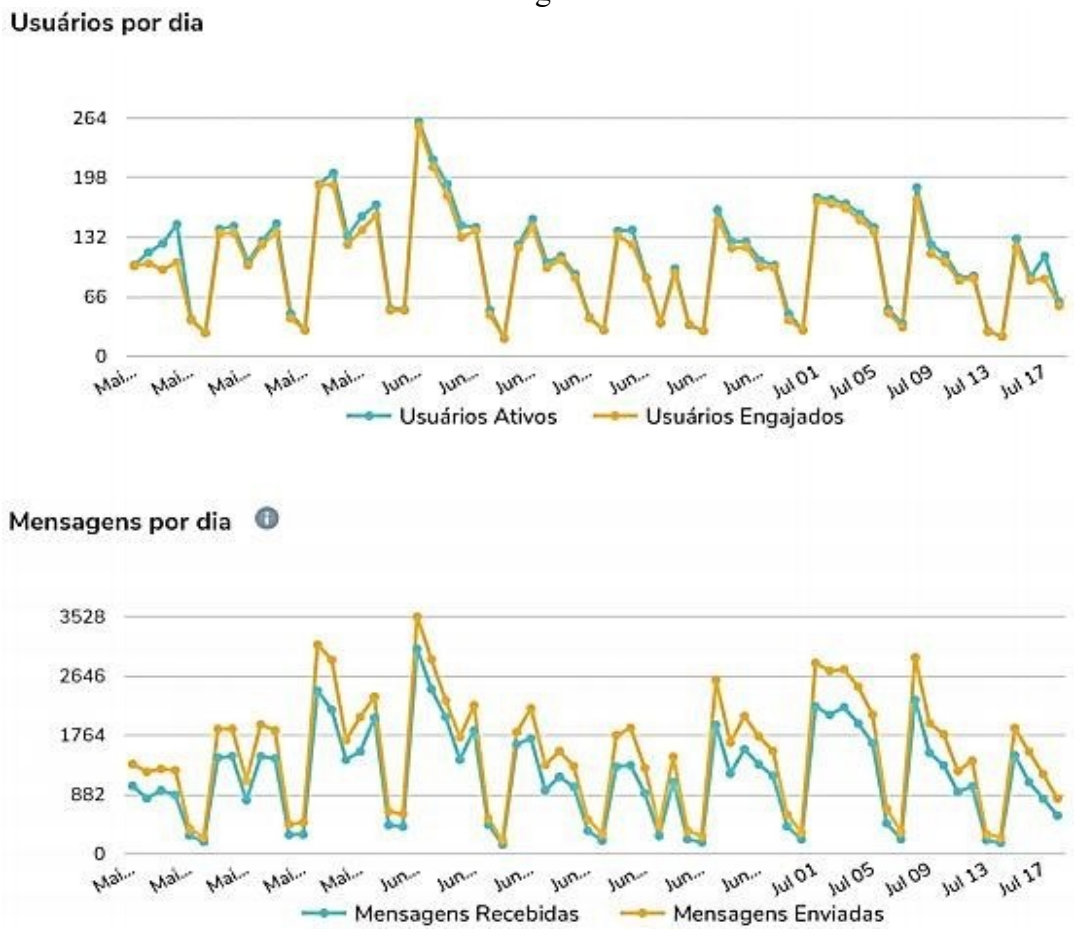
A primeira etapa foi fazer uma avaliação dos resultados obtidos pelo *chatbot* previamente implementado na Brognoli, para se utilizar o mesmo como base para o novo projeto. Em paralelo a isso foram realizados estudos mais aprofundados sobre o desenvolvimento de AV e *chatbots*, bem como um aprofundamento nos conhecimentos na linguagem de programação utilizada nesse projeto, o javascript. Outros assuntos estudados para esse projeto foram serviços web, protocolo SOAP e Arquitetura Orientada a Serviços (SOA).

Após a conclusão destas tarefas, foi feito um levantamento dos serviços web disponibilizados pela Brognoli e analisado como eles poderiam ser utilizados para compor as funcionalidades do *chatbot.* Hoje nosso *bot* é desenvolvido no *Blip* com uma estrutura híbrida: NLP + Fluxo de blocos de conversação se utilizando principalmente de menus com perguntas e respostas.

De forma geral, tentamos identificar o assunto principal do motivo do contato do usuário através das intenções cadastradas (provedor de IA: IBM Watson) e, a partir disso, direcionamos o usuário para um fluxo específico com menus e seguindo esses fluxos principalmente baseado em regras. Os assuntos principais tratados

hoje pelo *bot* são: Quero alugar 37%, Assuntos financeiros 20%, Processo de desocupação 7%, Proprietário 5%, Manutenção 4%, Vistoria 3%. No contexto "Quero alugar", por exemplo, temos um fluxo que possibilita o usuário de agendar uma visita. Nesse fluxo temos uma integração com o CRM (Bitrix24) utilizado pela Brognoli Imóveis via requisição HTTP, que cria um Lead com os dados do cliente. Para o atendimento financeiro, por exemplo, também consumimos algumas APIs referentes a emissão de segunda via de boletos. Outra forma de integração que também usamos é da forma CRM para BLiP. É usado principalmente para o envio de mensagens ativas do *WhatsApp*. Ações específicas no CRM disparam uma chamada num *webhook* externo (Integromat) e lá são feitas chamadas HTTP para o *bot*, principalmente para enviar essas mensagens ativas. Em diversos fluxos do *bot* também executamos scripts para verificar e tratar dados de forma geral.

Figura 1- Funcionamento do bot de maio a abril de 2019, divido em Usuários/dia e Mensagens/dia.



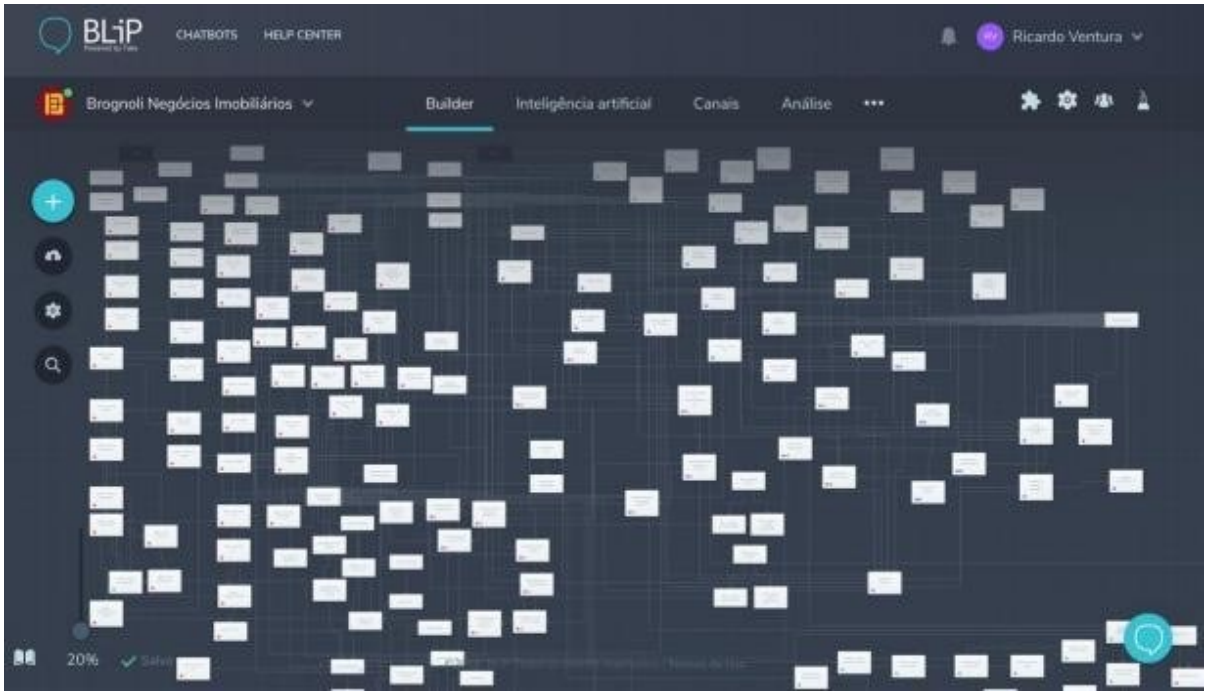
Fonte: O autor, 2019.

2.3.1 Entendimento do Problema

A identificação do problema e entendimento do mesmo foram os temas da primeira etapa do projeto. Após um período de análise, entendeu-se que para possibilitar o desenvolvimento de um trabalho mais robusto, era necessário explorar mais áreas de atuação para o bot.

Builder é a interface visual para desenvolvimento do *chatbot* dentro da plataforma do BLIP: O *Builder*, num primeiro momento, facilita muito a o início de um projeto, ele é bastante intuitivo e facilita a visualização no início. Conforme os assuntos e as funcionalidades do nosso *bot* foram aumentando, o *builder* se tornou bem mais complicado de entender, ficou mais difícil visualizar os fluxos e também de fazer mudanças em grande escala. Algumas mudanças influenciam vários blocos, então todos eles precisam ser modificados, o que resulta num trabalho longo e suscetível a erros.

Figura 2 – Interface Builder.



Fonte: O autor, 2019.

Error Handling: A atualização feita recentemente facilitou bastante o tratamento de erros nas "Ações" do *bot*, mas antes disso era bastante trabalhoso ter que lidar com cada saída possível com um bloco novos, visto que no nosso caso, em alguns blocos, utilizamos mais de dez ações em sequência. Como tínhamos pressa em implementar algumas funcionalidades, acabamos por não tratar os erros da forma adequada.

2.3.2 Sugestões

- Mapeamento de informações;
- Estrutura com *sub-bots;*
- Consultoria BLIP;
- Outras empresas de IA.

Com o problema e as dores do atendimento devidamente identificados e mapeados, realizou-se pesquisas e estudos em bibliografias especializadas do assunto sobre os diferentes tipos e categorias de assistentes virtuais, suas funcionalidades e suas aplicações. A necessidade de se ampliar a equipe foi identificada para acelerar o desenvolvimento do produto e também para elevar a qualidade do mesmo.

Para isso, entendemos que precisamos reestruturar o que já foi desenvolvido, melhorar a organização, documentação, etc. O intuito é expandir o projeto, mudando a estrutura de um *chatbot* único, que trata todos os assuntos, para a estrutura de *sub-bots,* cada um especialista numa área de atuação da imobiliária. *Sub-bots:* Quero alugar, Comprar/Vender, CS Financeiro, Manutenção, Desocupação, CS Proprietário, Vistoria. Pretende-se mapear mais assuntos e dúvidas que hoje são tratadas pelo atendimento humano e criar mais intenções e fluxos de conversa com esses assuntos para diminuir o volume de pessoas que são direcionadas ao atendimento humano. Acredita-se que essa divisão em *sub-bots* vai facilitar bastante esse processo. Desenvolvendo uma estrutura de IA bem robusta para conseguir responder a essas perguntas e, no futuro, usar esses dados para aplicações mais avançadas. Um outro objetivo é que o *chatbot* realmente realize mais ações além de só responder as dúvidas. Integrar mais serviços ao *bot* para, por exemplo, realizar a abertura de chamados no CRM da imobiliária.

2.3.3 Desenvolvimento

O desenvolvimento de módulos já reprojetados foi imediato, com as definições feitas e problemas maiores já mapeados. Esses módulos entraram em produção no *chatbot.*

Hoje o contato inteligente está presente no site da imobiliária através do *Widget* e também no *WhatsApp* com o número de contato principal. Com objetivo de que o contato inteligente trate de assuntos comerciais com os potenciais clientes e também atenda aos clientes atuais (locadores e locatários). Também foi utilizado o Desk para solucionar problemas mais complexos e urgentes com atendentes especializados.

Hoje utilizamos os provedores de IA alimentando todos os dados diretamente no BLiP, o que facilita bastante no início. Mas como queremos utilizar de forma massiva esses serviços de IA, eu gostaria também de entender se é possível usar, por exemplo, algumas funcionalidades nativas do Watson. E se sim, como eu poderia fazer esse tipo de coisa. Por exemplo temos @sys-date e @sys-time que facilitam muito na hora de marcar uma agenda. Queria entender se é possível usar essas funcionalidades da forma que as coisas são integradas hoje, inserindo as intenções direto no *Blip*. Se não conseguir usar dessa forma, gostaria de entender como funcionaria para usá-las utilizando o Watson apenas via chamada de API, com boa parte das coisas feitas só lá, etc. De forma geral seria um curso de boas práticas para o desenvolvimento de um *chatbot* escalável na plataforma de vocês. Eu assisti a todas a aulas disponibilizadas pela *Take* no site e foram muito proveitosas, mas senti essas dificuldades citadas quando o projeto foi se tornando maior e mais complexo.

2.3.4 Avaliação

Os *chatbot* está em produção e sendo utilizado pelos clientes da imobiliária. A avaliação está diretamente relacionada aos números gerados nas trocas de mensagens com os usuários bem como no uso das funcionalidades disponíveis.

Os resultados desses testes forneciam material para melhorar essas funções e também evitar futuros problemas em novas funções implementadas. Esses ciclos de melhoramento já estavam previstos na metodologia *Design Science*.

2.3.5 Conclusões

Acredita-se muito no potencial do projeto e almeja-se ser pioneiros nos conceitos de contato inteligente, *chatbots* e inteligência artificial no mercado imobiliário.

Atualmente a versão 1.0 do *chatbot* está rodando. E a versão 2.0 está em desenvolvimento, trabalhando integrações com outros sistemas da empresa e atuando em fluxos ativos de comunicação disparados por automações em estágios específicos do CRM da imobiliária. Já está aprovado pela direção da empresa a ampliação da equipe e pelo menos mais três pessoas serão contratadas para compor o time, o objetivo é futuramente esse *chatbot* virar um produto que possa ser utilizado por outras imobiliárias.

3 REVISÃO DA LITERATURA

Essa seção do trabalho é destinada à revisão bibliográfica das principais abordagens e conceitos utilizados neste projeto.

3.1 *CHATBOTS* E ASSISTENTES VIRTUAIS

O objetivo de um *chatbot* é de interagir com o usuário sobre determinado assunto simulando uma conversa em linguagem natural, utilizando técnicas de inteligência artificial para alcançar esse objetivo. No geral, a função do *chatbot* é a de responder perguntas frequentes dos usuários de maneira natural como uma pessoa real faria. A finalidade de se utilizar essa abordagem é de que o usuário esqueça que está conversando com um programa de computador (TEIXEIRA, 2005).

Neves (2005) separa os *chatbots* em três grupos de acordo com as técnicas utilizadas no desenvolvimento dos mesmos.

- Casamento de padrões e Regras Gramaticais, que se baseia na utilização de palavras-chaves retiradas das perguntas do usuário para encontrar respostas pré-programadas para responder o usuário;
- Inteligência artificial, que funciona com a utilização de redes neurais para identificar a intenção do usuário e encontrar a resposta mais adequada para a pergunta;
- Linguagem de marcação baseado em AIML, considerado o método mais completo no quesito de processamento de linguagem natural.

Segundo Laureano (1999), os *chatbots* podem ainda ser divididos em grupos de acordo com as funcionalidades que entregam.

- Educacional, onde o *chatbot* tem como objetivo responder dúvidas de alunos e professores e ainda apresentar diversos tipos de informações relevantes;
- Entretenimento: tem apenas o intuito de divertir e passar o tempo do usuário;
- F.A.Q. (*frequently asked questions*), tem o objetivo de responder as perguntas mais recorrente dos usuários para um dado assunto;

- Comercial, funciona como um atendente de clientes para dar suporte ou vender produtos e serviços;

- Propóslito Geral, *chatbots* que não se encaixam em nem uma outra categoria. Na literatura podemos encontrar diversas definições e descrições para assistentes virtuais, porém nem uma delas consegue abarcar todos os conceitos satisfatoriamente. Então normalmente se utiliza os requisitos do projeto para o descrever. Em geral, os requisitos mais comuns são flexibilidade, autonomia, adaptabilidade de contextos, integração com o usuário, ser adaptativo para as necessidades, ter um propósito ter integrações com sistemas e ser interoperável (ZAMBIASI; RABELO, 2012).

  Algumas das implementações descritas na literatura são descritas a seguir.

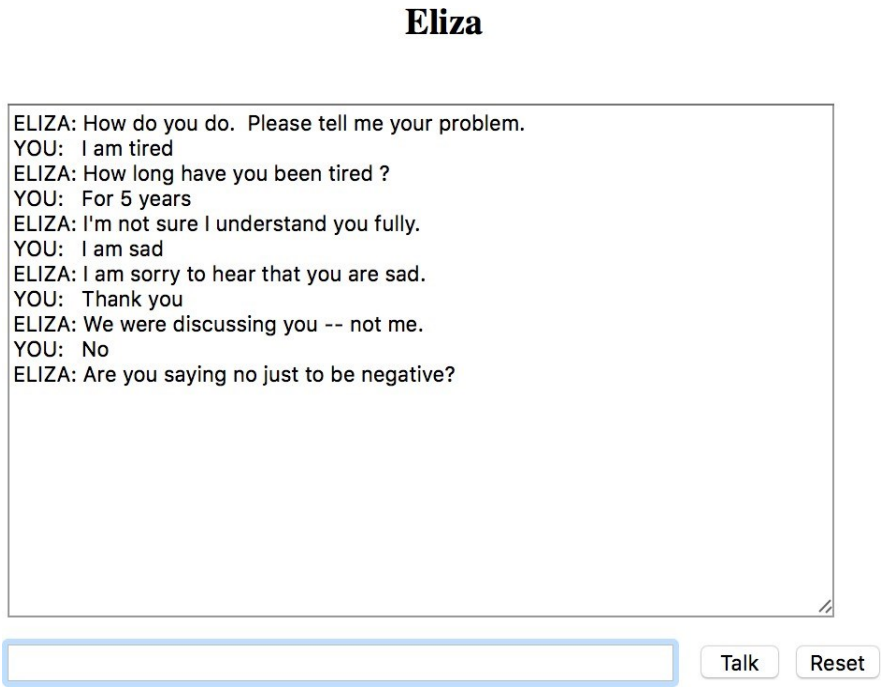
3.1.1 ELIZA

É preciso citar o primeiro *chatbot* criado, que foi o ELIZA, em 1966 o Joseph Wizenbaum terminou o desenvolvimento do ELIZA. A principal função ou objetivo do *chatbot* era se passar por uma psicoterapeuta, sendo considerado um dos primeiros programas de I.A. (Inteligência Artificial) desenvolvidos (WEIZENBAUM, 1966).

Em testes com usuários que desconheciam a existência de programas de inteligência artificial, obteve-se um resultado inusitado até mesmo para Wizenbaum, a maioria dos usuários pensava que estava falando com outra pessoa, mesmo falando horas seguidas com o programa.

O *chatbot* ELIZA teve uma implementação bem simples, o principal artificio para responder as perguntas do usuário e utilizar as próprias entradas do usuário e transformar em outras perguntas previamente elaboradas para casos em que não se conseguia utilizar a primeira abordagem. As respostas prontas e informações fornecidas pelo *chatbot* tinham a intenção de serem amigáveis (WEIZENBAUM, 1966). A seguir temos a Figura 3 que mostra um exemplo de diálogo com o *chatbot* ELIZA.

Figura 3 - Diálogo em uma versão alternativa do ELIZA.



Fonte: Adaptado de Weizenbaum.

3.2 ARTIFICIAL INTELLIGENCE MARKUP LANGUAGE (AIML)

A *Artificial Intelligence Markup Language* (AIML), fundamentada na linguagem *e Xtensible Markup Language* (XML) é uma linguagem de marcação. O objetivo da criação da mesma é de, por meio de computação, simular inteligência humana em uma conversa, dialogo com um ser humano (LEONHARDT, 2003).

Essa linguagem é utilizada para programar a base de conhecimento de *chatbot,* funcionando como um meio de selecionar uma resposta para o usuário de acordo com as entradas do mesmo. Essa linguagem é simples de se implementar por ser baseada em um conjunto de comandos e marcações simples (LEONHARDT, 2003).

A linguagem se baseia em objetos chamados AIML, que são formados por unidades *category* e *topic.* As unidades são descritas em detalhes a seguir.

3.2.1 Category

Segundo WALLACE (2008) a unidade *Category* é definida como uma unidade de conhecimento básico composta por uma entrada do usuário, uma resposta do *bot* e o contexto desse diálogo, o que pode ser opcional. As *tags* <*pattern*> e <*template*> definem, respectivamente, a entrada do usuário e a saída dada pelo

bot.

Dentro do objeto <*category*> o elemento <*pattern*> deve ser o primeiro bem como deve ser único para cada objeto <*category*>. Na Figura 4 temos um exemplo da linguagem AIML.

Figura 4 - Exemplo de AIML.

```
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern> WHO IS ABRAHAM LINCOLN </pattern>
    <template>Abraham Lincoln was the US President during American civil war.</template>
  </category>

  <category>
    <pattern>DO YOU KNOW WHO * IS</pattern>
    <template>
      <srai>WHO IS <star/></srai>
    </template>
  </category>
</aiml>
```

Fonte: Maruti Techlabs.

3.2.2 Topic

O elemento *Topic,* é um elemento opcional da linguagem AIML e é formado de elementos do tipo *Category.* Esse elemento basicamente serve para reunir elementos do tipo *Category* com assuntos relacionados em um único objeto. Ele precisa de um atributo identificador, como por exemplo, um nome. Segundo Wallace (2008) esse atributo pode ser definido por meio de qualquer *template*.

3.2.3 Outras *tags*

A linguagem conta ainda com outras *tags* que valem a menção:

- <*random*>: essa *tag* é utilizada para sortear aleatoriamente uma reposta tagueada com a marcação <*li*>;
- <*srai*>*:* é uma *tag* utilizada para se evitar a repetição da resposta, redirecionando a pergunta do usuário para outra resposta. Ela pode ser usada dentro da *tag* <*template*>;
- <*set*>: é utilizado quando se quer salvar alguma variável;
- <*think*>: tem a função de salvar uma variável sem que o usuário seja alertado.

4 PROPOSTA

Neste capítulo é abordado a proposta de modelagem e o design do projeto.

4.1 BASE DE CONHECIMENTO

A criação de uma base de conhecimento é fundamental para o desenvolvimento de um *chatbot* ou assistente virtual. Para se documentar e criar a mesma, foi utilizada a ferramenta miro, onde de forma visual e intuitiva se armazenou as informações para a base de conhecimento. A ferramenta miro é uma ferramenta visual de criação de fluxos, no caso foi utilizada para a criação de fluxos de conversas. Essa abordagem facilita o trabalho de coletar e visualizar as informações obtidas.

A base de conhecimento foi obtida por meio de mapeamento de processos de atendimento ao cliente feitos por atendentes humanos, bem como serviços que se pretendia oferecer no assistente virtual.

O mapeamento das dores e perguntas mais frequentes do usuário foram feitos em diversos pontos do atendimento dentro da imobiliária. O atendimento da imobiliária é dividido em setores: C.S. (*customer success*) que trata de assuntos relacionados aos locatários de imóveis; NPR (núcleo do proprietário) como o próprio nome referência, trata de assuntos relacionados ao proprietário; financeiro, trata de assuntos gerais financeiros; Jurídico, trata de assuntos jurídicos; Manutenção, trata de assuntos de manutenção. O foco da obtenção do conhecimento foi no de extrair principalmente perguntas mais recorrentes para desafogar o atendimento humano e melhorar o tempo de resposta ao usuário.

Os processos mapeados e projetados para o assistente virtual foram os mais recorrentes e com a melhor possibilidade de serem automatizados e disponibilizados sem a intervenção de um atendente humano para que fossem concluídos. As funcionalidades mapeadas foram as de abertura de chamados de manutenção, financeiros para proprietário, financeiros para locatário, segunda via de boleto, procura ativa por imóveis e agendamento de visitas aos imóveis.

4.1.1 Brognoli

A Brognoli é uma grande imobiliária de Florianópolis e região, que tem uma cartela de clientes gigantesca, com mais de 10 mil entre donos e locatários de imóveis. Os volumes

de atendimentos ao cliente são gigantescos, e por isso vinha encontrando dificuldades em manter a qualidade do atendimento e o baixo tempo de resposta. Nesse cenário, o gerente do setor de T.I., Gabriel Diederichsen proativamente iniciou o desenvolvimento de um *chatbot* destinado a funcionar como um F.A.Q. para desafogar o atendimento ao cliente. O *chatbot* apesar de muito simples, trouxe algum resultado, validando a ideia de se criar um assistente virtual para os clientes da imobiliária.

Em seguida, o engenheiro de controle e automação Ricardo Ventura foi contratado para desenvolver a primeira versão e primeiras integrações do assistente virtual com os serviços *web* da imobiliária. Essa primeira versão está em uso pelos clientes desde a metade do ano, trazendo resultados e dados para a segunda fase do desenvolvimento. Com essa primeira versão do assistente virtual real, o gerente do setor de T.I. Gabriel, conseguiu mais incentivo para a criação de um assistente mais completo e pensado especificamente para o cenário de digitalização do atendimento ao cliente. Para o desenvolvimento foi formada uma equipe multidisciplinar composta por: uma estagiária de *designer* com foco em UX, um engenheiro de Controle e Automação, uma estagiária de psicologia e um estagiário de Engenharia de Controle e automação (autor desse documento).

As funções que foram desenvolvidas pelo autor deste documento são:

- Atendimento de clientes interessados em alugar um imóvel específico: informando características do imóvel; mostrando a localização do imóvel no *google maps*; informando sobre as garantias que a imobiliária solicita para o aluguel; agendando uma visita diretamente pelo chat ao imóvel; apresentando no *google maps* a agência para retirar a chave antes da visita ao imóvel; e por fim encaminhar o cliente para atendimento humano caso necessário;
- Abertura de chamados de manutenção: abrir um chamado diretamente pelo *chat* pedindo as informações pertinentes para se preencher o formulário de abertura de chamados no CRM, incluindo descrição do problema bem como documentos em anexo;
- Abertura de chamados financeiros proprietário/locatário: funciona de maneira análoga ao chamado de manutenção.

4.1.2 Atendimento ao cliente

O atendimento ao cliente com interesse em alugar um imóvel é uma das principais funções desenvolvidas nesse projeto por se tratar do carro-chefe da empresa. Os clientes com a intenção de alugar um imóvel são chamados de *leads*. *Lead* se define como uma

pessoa que tem interesse em adquirir um bem ou um serviço fornecido por uma empresa, loja, estabelecimento e etc.

Os *leads* que são tratados pelo assistente virtual são provenientes tanto de pessoas que pedem o contato referente a algum imóvel anunciado no site da própria imobiliária como em outros sites de anúncio de imóveis. Esses leads recebem um contato ativo do assistente virtual por meio do App *whatsapp*, para apresentar mais informações e agendar uma visita ao imóvel.

Outro tipo de atendimento ao cliente que o *bot* realiza é o de pós-venda, que nada mais é que o atendimento ao cliente que já é locatário ou é proprietário de algum imóvel que a imobiliária administra. O objetivo dessa parte do atendimento do cliente é tirar as principais dúvidas de forma automática para diminuir a carga dos atendentes e para melhorar o tempo de resposta.

Com o avançar dos estudos sobre as áreas do NPR (núcleo do proprietário) e CS (*customer success*), observou a possibilidade de oferecer serviços que são facilmente automatizáveis, e no caso de problema não resolvido, ainda encaminhar o cliente para a área adequada de atendimento humano para a resolução do problema.

4.2 SELEÇÃO E DESENVOLVIMENTO DE SERVIÇOS WEB

Os serviços escolhidos foram baseados em demandas mais comuns que tinham a possibilidade de serem automatizadas. Como dito na seção anterior, existem duas grandes áreas que o *bot* deve atender, a área de atendimento de *leads* ou clientes que desejam alugar um imóvel e a outra área que é para o atendimento de donos e locatários de imóveis administrados pela imobiliária.

Para o primeiro setor, para entregar uma boa experiência no primeiro contato com clientes interessados em alugar um imóvel, é preciso integrar o sistema do *bot* com o *whatsapp*. Além da integração com o *whatsapp*, tem de se integrar de alguma maneira os e-mails que os sites parceiros enviam com as informações dos leads para, primeiramente, registrar o lead no CRM da imobiliária, em seguida disparar uma automação no CRM para realizar uma chamada de API que comunica os dados do *lead* para o *bot*, pôr fim, para o mesmo entrar em contato com o cliente sobre o assunto de alugar o imóvel registrado no lead.

Outras funcionalidades que demandam serviços *web* que foram selecionadas são, a abertura de chamados financeiros, abertura de chamados de manutenção e segunda via de

boletos. Os serviços *web* necessários para a abertura de chamados via *bot* são, basicamente integrações com os sistemas CRM e RP financeiros da empresa. O CRM para o registro do chamado e acionamento dos responsáveis com as informações passadas do cliente sobre o chamado, e o RP para a verificação dos dados do cliente referentes ao imóvel. O processo para a emissão da segunda via de boletos também é dependente do RP da empresa, que é integrado via chamadas de API no bot.

4.3 ESPECIFICAÇÃO DOS REQUISITOS

Os requisitos para o projeto foram levantados de acordo com as demandas e dores mais comuns dos clientes da imobiliária. Conforme o desenvolvimento do projeto e levantamento de informações nas áreas, as funcionalidades foram adaptadas e melhoradas para o projeto que está em produção. O projeto também visou explorar ao máximo a potencialidade de automatizar demandas comuns e repetitivas.

4.3.1 Requisitos funcionais

Segundo Vazquez (2013), requisitos funcionais (RF) capturam o que o software deve fazer em termos de tarefas e serviços, são requisitos relativos às práticas e procedimentos de negócio do usuário, particulares de determinada tarefa e serviço.

Os requisitos funcionais deste projeto são:

- RF 01: o Assistente Virtual da Brognoli deve funcionar como um F.A.Q. para os X contextos anteriormente estudados e definidos;
- RF 02: O Assistente Virtual deve identificar a fala natural humana e responder de acordo com o contexto da entrada do usuário;
- RF 03: O Assistente Virtual deve contatar os interessados por imóveis da imobiliária de forma ativa;
- RF 04: O Assistente Virtual deve fazer o encaminhamento do cliente para o atendimento humano no setor competente;
- RF 05: O Assistente Virtual deve conter serviços aos clientes de pré- venda e pós-venda.

4.3.2 Requisitos não funcionais

Os requisitos não funcionais (RNF) especificam ou restringem as propriedades emergentes do sistema. Portanto, eles geralmente são mais importantes do que os requisitos funcionais individuais (SOMMERVILLE, 2007 apud BENITTI & RHODEN, 2013). Por isso, esses requisitos necessitam ser capturados e analisados desde os primeiros momentos do desenvolvimento do software. Erros provocados por não se lidar adequadamente, ou simplesmente não se lidar com RNF, são apontados entre os mais caros e difíceis de corrigir (CYSNEIROS, 2001, apud BENITTI & RHODEN, 2013).

Os requisitos não funcionais deste projeto são:

- RNF 01: Levantar e mapear conhecimentos necessários para o F.A.Q. e colocar na plataforma BLIP;
- RNF 02: Escrever *scripts* na linguagem *javascript* para o tratamento lógico dos dados e programação geral do *bot*;
- RNF 03: Serviços web devem ser utilizados para realizar a comunicação do bot com outros serviços e programas internos da imobiliária;
- RNF 04: Os passos das funcionalidades devem ser registrados e atualizados no CRM da empresa para controle de fluxo via requisição de serviço web.

4.4 CASOS DE USO

Nesta seção os casos de uso do Assistente virtual são apresentados de maneira geral. Os casos de uso são divididos em:

- UC 01: Captação de imóveis para aluguel
- UC 02: Encaminhamento de clientes interessados em vender seu imóvel direto para os corretores.
- UC 03: Abertura de chamados de assuntos financeiros, tanto para locatário como para proprietário.
- UC 04: Abertura de chamados de assuntos de manutenção.
- UC 05: Emissão de segunda via de boletos.
- UC 06: Informações sobre imóveis para aluguel.
- UC 07: Agendamento de visitas em imóveis para aluguel.
- UC 08: Criação de vitrine de imóveis de acordo com os interesses

5 FERRAMENTAS E TECNOLOGIAS

5.1 Blip.ai

A plataforma ou ferramenta utilizada para a criação do assistente virtual foi a blip.ai da empresa Take. Nesta seção será apresentada como funciona a plataforma e como criar um assistente virtual na mesma.

5.1.1 Configuração do Assistente Virtual

Para a criação de um novo chatbot, deve-se primeiramente acessar o site https://blip.ai/, criar uma conta grátis. Após criada a conta, apertar no botão “Criar chatbot” (Figura 5), em seguida selecione a opção “Criar do zero” (Figura 6), finalmente escolha um nome para seu *chatbot* e finalize (Figura 7).

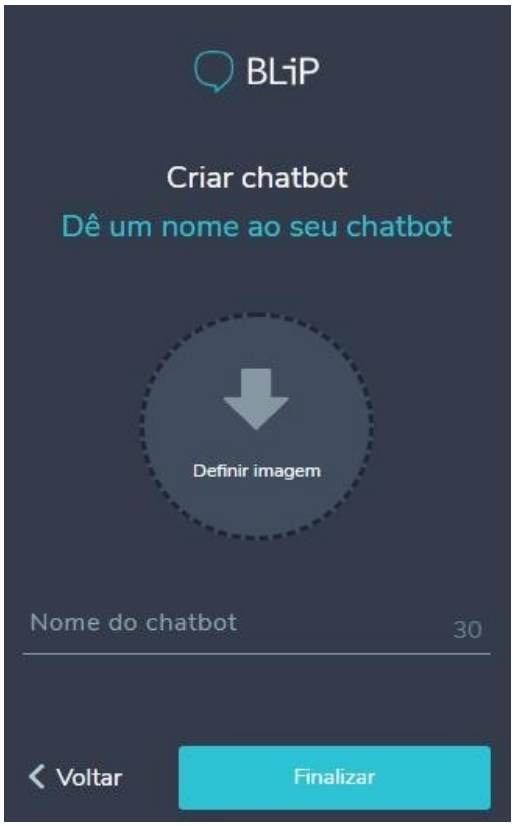
Figura 5 - Criar chatbot na BLiP.

Fonte: O autor, 2019.

Figura 6 - Criar chatbot na BLiP.

Fonte: O autor, 2019.

Figura 7- Nome do chatbot na BLiP



Fonte: O autor, 2019.

5.1.2 Builder

O sistema de criação e programação da plataforma BLiP é baseado em uma ferramenta visual mista com serviços e *scripts* chamada *builder*. No *builder* temos a presença dos blocos, que podem ser interpretados como estados do usuário. A forma como o *chatbot* é feita na plataforma BLiP é basicamente uma máquina de estado.

5.1.3 Blocos

Cada bloco tem um nome próprio que deve ser dado de acordo com as funções que ele exerce no *flow*. Além de um nome, ele é composto de 3 principais instâncias. A primeira é a de "Conteúdo", nesta seção podemos colocar conteúdos diversos para apresentar para o usuário. Como estamos criando um *chatbot* com o objetivo de se comunicar por meio do App *whatsapp*, temos algumas limitações quanto aos conteúdos suportados na plataforma. Na Figura 8 a seguir temos todos os conteúdos que podem ser adicionados.

Figura 8 - Conteúdos que podem ser adicionados.

Fonte: O autor, 2019.

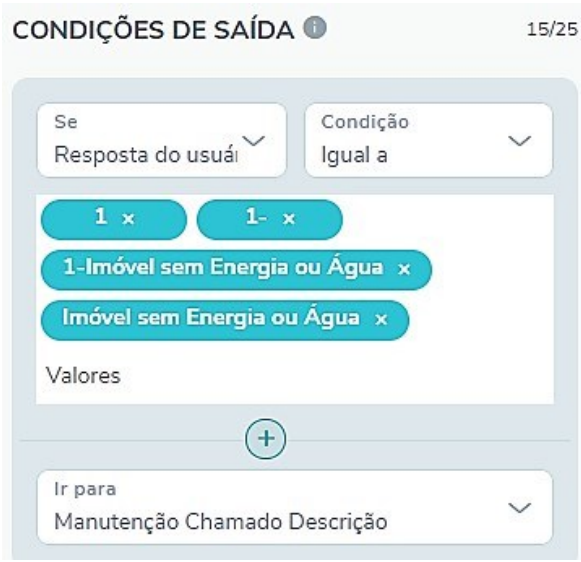
Além de entregar conteúdos para o usuário, nessa aba também podemos programar esse bloco para esperar uma entrada do usuário e ainda ter a possibilidade de salvar essa entrada para ser usada posteriormente. Essa informação salva pode ser usada em *scripts*, podem ser salvas como informações de contato, ou ainda podem ser usadas para decidir para qual bloco a conversa se encaminhará de acordo com as condições de saída do bloco.

5.1.4 Condições de saída

A aba "Condições de saída" é efetivamente onde a decisão de para onde o usuário vai no *flow*. A lógica de decisão é basicamente um *if* seguido de *else if* para as condições de

saída do bloco. Existem diversos tipos condições, elas são: Existe, Igual a, Diferente de, Contém, Começa com, Termina com, Maior que, Menor que, Maior igual a, Menor igual a, Precedido com, Corresponde à regex e não existe. Existem quatro tipos de entradas para o *if* que podem ser testados como condições de saída, Resposta do usuário, Variável, Intenção identificada e Entidade identificada. Como podemos ver, existem diversas maneiras de se encaminhar o usuário para o bloco que corresponde com a sua entrada ou intenção no *flow*. As intenções e Entidades são informações que a NLP integrada ao BLiP tira da entrada do usuário, e pode ser usado para entender para onde o rumo da conversa deve seguir para atender as dúvidas e desejos do usuário. As condições de saída tiradas de um dos blocos do *flow* de abertura de chamado, está ilustrada na Figura 9.

Figura 9 - condições de saída tiradas de um dos blocos do flow de abertura de chamado.



Fonte: O autor, 2019.

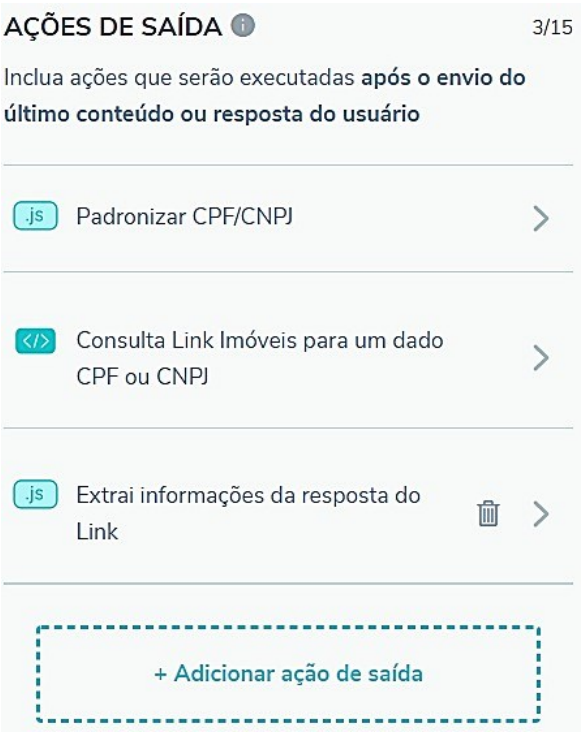
5.1.5 Ações

Na aba Ações se encontram todos os recursos que que podem ser executados para entregar um serviço ou tratar uma entrada do usuário. As ações disponíveis nesta aba são: Requisições HTTP, Registro de eventos, Definir contato, Redirecionar a um serviço, Gerenciar lista de distribuição, Executar script, Definir variável, Processar comando. Todas essas ações podem ser divididas em ações de entrada e de saída.

As ações de entrada são as primeiras coisas a serem executadas no bloco, elas são executadas com as informações vindas do bloco anterior, e seu resultado pode ser usado para entregar uma mensagem para o usuário ou um serviço, mais a frente será explicada mais sobre as principais ações que podem ser executadas.

As ações de saída são por penúltimo, pois ainda é executada toda a lógica de condição de saída do bloco, portanto, essas ações podem ser determinantes para determinar o destino da conversa. A Figura 10 mostra a aba de ações com alguns exemplos de sequência de ações em um bloco do *bot*.

Figura 10 - Imagem da aba de ações com alguns exemplos de sequência de ações em um bloco do bot.



Fonte: O autor, 2019.

5.1.6 Ação: Requisição HTTP

Dentre as ações possíveis de serem executadas na aba de "Ações", a requisição HTTP se destaca para a aplicação que o *chatbot* vai exercer, pois ela possibilita a integração do chatbot com diversos serviços *web* que são de suma importância para entregar os serviços que o usuário necessita.

A lista de requisições possíveis é: GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD, TRACE. Ainda é possível adicionar cabeçalhos, onde colocamos as informações necessárias para acessar as funcionalidades do serviço *web* escolhido. Além disso, temos campos para salvar a resposta do serviço *web*, e ainda um outro campo para salvar o status da resposta. Por fim, temos a condição de execução da requisição, que funciona de maneira semelhante às condições de saída, porém não é obrigatória.

5.1.7 Ação: Executar script

Esta ação é o grande coringa deste sistema de criação de *bots*. Esta ação consiste basicamente na execução de uma função *javascript* com entradas do tipo variável e uma única saída. Ela se torna o grande coringa, por possibilitar a execução de qualquer lógica de programação para o tratamento dos dados e para execução de lógicas mais complexas. São exemplos de algoritmos ou scripts:

- Extração de informações de objetos do tipo JSON, que normalmente são respostas de requisições HTTP executadas previamente, ou resposta de outros serviços integrados do BLiP;
- Algoritmos de Regex, para validar a entrada do usuário e garantir que os dados entregues são os desejados, além de poder tratar possíveis problemas que textos com formatações diversas podem causar na construção de objetos do tipo JSON utilizados como informações para a utilização de serviços *web* via requisição HTTP;
- Algoritmos para o aglomeramento das informações em objetos para serem utilizados para outras ações futuras;
- Algoritmos para controle de datas, horários, filas de atendimento.

No Anexo A temos o exemplo de um script bem como toda a interface da ação executar script no bloco.

5.2 BITRIX

Para a contextualização do trabalho, deve-se realizar uma breve introdução sobre o CRM utilizado na imobiliária, o Bitrix. Esse sistema se trata de uma plataforma Russa de serviços relacionados ao gerenciamento de empresas. O recurso mais utilizado nesse projeto é o do CRM, mas a seguir temos todos os recursos disponíveis na plataforma segundo o próprio site da aplicação:

- Colaboração (bate-papos, mensagens de fluxo, pesquisas, integração de telefonia, ligações & chamadas de vídeo);
- CRM (clientes potenciais, contatos, negociações, faturas, formulários da web, relatórios de vendas, automação de vendas, integração de canais abertos, widget do site);
- Gerenciamento de Projeto (tarefas, subtarefas, gráficos de Gantt, kanban,

dependências, tarefas recorrentes & acompanhamento do tempo da tarefa);

- Gerenciamento de Documento (armazenamento de arquivo & compartilhamento, sincronização com computador local, integrações, fluxos de trabalho para biblioteca de documentos);
- Gerenciamento do Tempo (cálculo automático do tempo de trabalho & resumo, relatórios de trabalho, calendários compartilhados);
- RH (diretório de funcionários, estrutura da empresa, gráfica de ausência & solicitação de férias fluxo de trabalho, anúncios & emblemas de reconhecimento).

Como mencionado anteriormente, a função de CRM é a mais importante para o projeto. Os motivos são:

- Acesso a lista de contatos de clientes, para realizar consultas para validação de informações ou apenas obtenção dos mesmos;
- Tratamento de leads, que são clientes em potencial, gerenciando as ações necessárias para contato e atendimento desses clientes em potencial;
- Criação e tratamento de negócios, que são ações que devem ser realizadas pela a empresa, e tem seus passos mapeados em *pipelines* que são gerenciados dentro do Bitrix, por meio de automações;
- Integrações com o RP financeiro da empresa;
- Consulta de produtos registrados no banco de dados.

Apesar da grande importância para o trabalho realizado, não será explicada em detalhes a implementação das lógicas dentro do bitrix, pois foram realizadas por outro setor da empresa antes mesmo da implementação do chatbot, e foram apenas adaptadas para o que se necessitava para o *bot*.

No Anexo B temos uma imagem da página inicial da plataforma para ilustrar melhor essa seção.

6 IMPLEMENTAÇÃO

Nesta seção será detalhada de como foi a implementação dos módulos que foram desenvolvidos para esse projeto. Os detalhes vão incluir interações, lógicas e informações usadas para a criação dos fluxos e suas funcionalidades. Serão apresentados exemplos de fluxos bem como tutoriais de como acessar as funcionalidades que estão disponíveis no *chatbot* oficial da imobiliária. Algumas funções são possíveis de ser acessadas apenas por clientes das imobiliárias.

São 3 módulos principais que serão apresentados a seguir, Chamados financeiros, Chamados Manutenção e Agendamento de Visitas.

6.1 MÓDULO DE ABERTURA DE CHAMADOS DE MANUTENÇÃO

Para demonstrar todo o desenvolvimento, será apresentado passo a passo do que foi feito por meio da apresentação dos blocos com os seus respectivos conteúdos. Alguns blocos serão passados de maneira mais rápida por trazerem conteúdos mais repetitivos, como menus e condições simples de saída. No Anexo C contém uma imagem geral dos blocos que formam essa seção do bot.

A nomenclatura dos blocos segue uma lógica simples, primeiro temos a tag do grupo ao qual ele pertence, e depois temos a sua função. O primeiro bloco é:

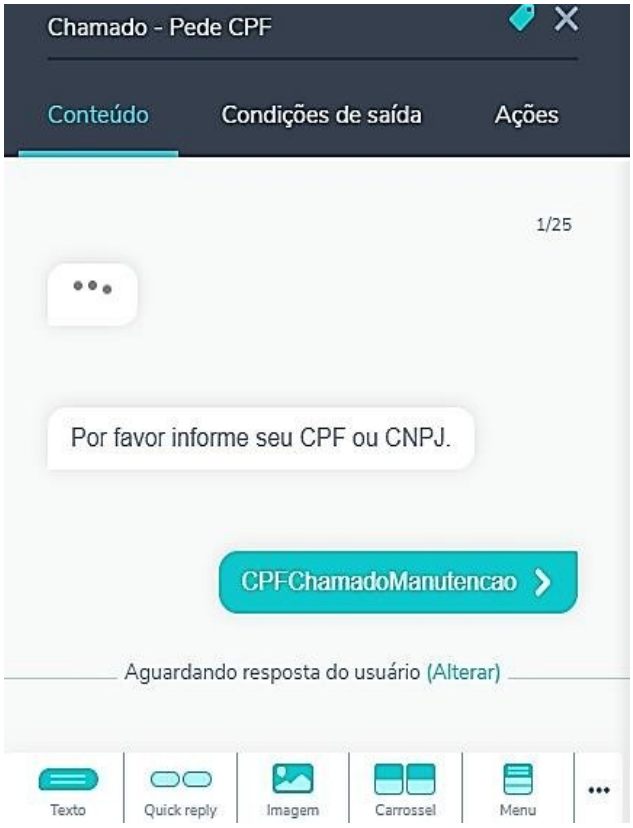
I. Chamado - Pede CPF. Na Figura 11 a aba Conteúdo do bloco.

Como podemos ver na imagem é pedido o CPF ou o CNPJ do cliente e é salvo na variável CPFChamadoManutencao. Essa informação será utilizada para se fazer uma chamada HTTP para acessar RP da imobiliária e retornar informações necessárias para o chamado. Optou-se por pedir o CPF do cliente ao invés do código do imóvel por se saber que a maior parte dos clientes não lembra ou não tem anotado o código do imóvel que aluga ou é dono.

Esse *flow* é acessado apenas por clientes que realmente querem abrir um chamado, portanto eles estão cientes do que estão fazendo.

Esse bloco não efetiva nem uma outra ação importante além de salvar a variável CPFChamadoManutencao, e em sua aba condições de saída encaminha a conversa para o próximo bloco no momento em que o usuário responde à pergunta.

Figura 11 - aba Conteúdo do bloco.



Fonte: O autor, 2019.

II. Chamado - Verifica CPF. A seguir temos a Figura 12 as Ações de Entrada do bloco.

Figura 12 - Ações de Entrada do bloco

Fonte: O autor, 2019.

Nesta aba temos 4 ações. A primeira ação, Definir Pergunta CPF é apenas para definir uma pergunta personalizada para os casos de erro em que precisamos pedir o CPF novamente.

A segunda e terceira ações, são para definir um contador de erros. A lógica é contar quantas vezes o usuário teve de responder à pergunta de qual é o CPF e caso tenha passado de três vezes, para evitar a frustração do cliente, ele é encaminhado para o atendimento humano.

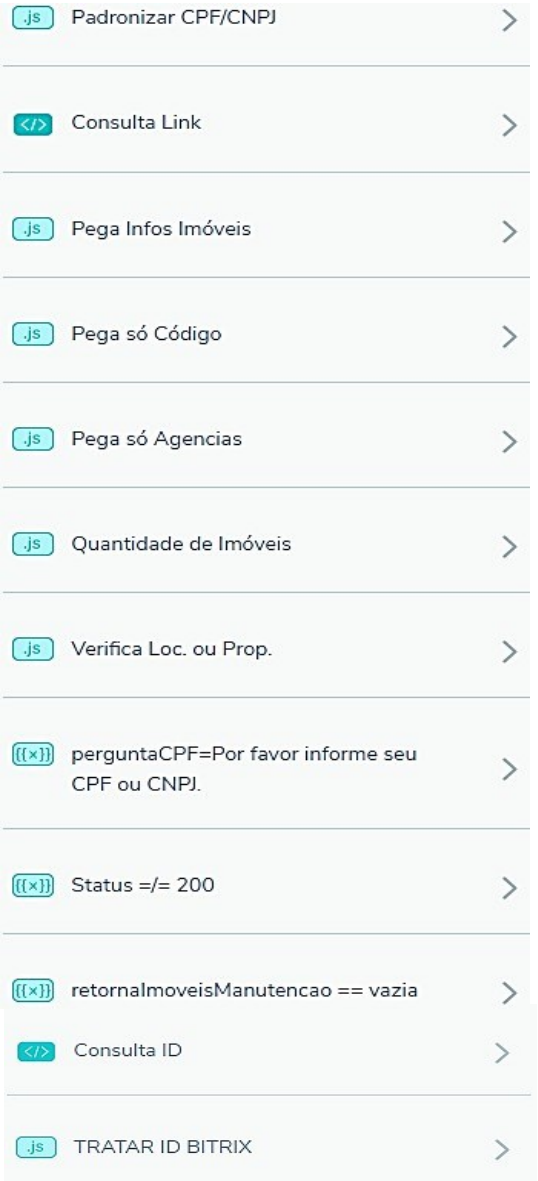
A quarta ação se trata de um registrador de eventos, que apenas registra o ID do usuário que chegou até esse menu, para se tirar um relatório de quantos e quais clientes estão usando essa funcionalidade de Abertura de chamados.

Na Figura 13, temos as ações de saída contidas neste bloco. Como pode ser visto, uma boa quantidade da lógica e dos algoritmos está neste bloco.

A primeira ação, Padronizar CPF/CNPJ, é um *script* para tratar a entrada do usuário. O *script* é bem simples e serve apenas para tirar tudo o que não é número de 0 até 9 da entrada do usuário. Isso se faz necessário pois a API consumida na próxima ação exige só números para a consulta. A variável de entrada, que é a mesma de saída, é a "CPFChamadoManutencao". Na Figura 14, mostra o código da função.

A segunda ação contida nesta aba é a Consulta Link. Essa ação se trata de uma requisição HTTP do método GET. A requisição é feita para a API do RP da imobiliária, que tem o nome de Link. A URL da requisição é http://api.brognoli.com.br/api/v2/imoveis/listar/{{CPFChamadoManutencao}}. Para que a requisição retorne as informações desejadas, que no caso são os imóveis relacionados ao CPF, é preciso mandar parâmetros no cabeçalho da requisição, além do CPF no fim da URL. O cabeçalho é composto por um "user" e sua respectiva "key" de acesso ao serviço da API. E na URL se pode observar a presença da variável já tratada que corresponde ao CPF ou CNPJ que o usuário informou no bloco anterior.

Figura 13 - Ações de saída contidas neste bloco.



Fonte: O autor, 2019

Figura 14 - Código da função.

```
function run(inputVariable1) {

   var cpf = inputVariable1

   cpf = cpf.replace(/[^0-9]/g, "")

   return cpf
}
```

Fonte: O autor, 2019.

A resposta dessa requisição retorna todos os imóveis relacionados com esse CPF e com mais algumas informações adicionais. Informações essas como se o CPF é Locatário ou Proprietário do imóvel, código do imóvel, agência responsável e etc., mais à frente iremos tratar essa resposta para a extração das informações. O formato da resposta é um objeto do tipo JSON. Um exemplo de uma resposta dessa requisição é ilustrado na Figura 15.

Figura 15 - Exemplo de resposta da consulta feita no link.


```
{
 "documento": "XX.XXX.XXX/XXXX-XX",
 "clienteProprietario": [
  {
    "codigo": XXX,
    "nome": "XXXXXXXXXXXXXXXX",
    "tipoCliente": "LOCATARIO",
    "imoveis": [
      {
        "codigo": XXX,
        "situacao": "ALUGADO",
        "agencia": X,
        "rua": "RUA XXXX XXXXX XXXX",
        "numero": "48",
        "lojaAp": "null",
        "cidade": "FLORIANÓPOLIS",
        "bairro": "BALNEÁRIO",
        "uf": "SC",
        "cep": "XXXXX-XXX"
             },
      {
        "codigo": XXX,
        "situacao": "ALUGADO",
        "agencia": 1,
        "rua": "RUA XXX XXXXXX XXX",
        "numero": "138",
        "lojaAp": "KIT 3",
        "cidade": "FLORIANÓPOLIS",
        "bairro": "BALNEÁRIO",
        "uf": "SC",
        "cep": "XXXXX-XXX"
             },
      {
        "codigo": XXXX,
        "situacao": "ALUGADO",
        "agencia": 1,
        "rua": "RUA XXXXXXXXX XXXXXX",
        "numero": "1725",
        "lojaAp": "202",
        "cidade": "FLORIANÓPOLIS",
        "bairro": "BALNEÁRIO",
        "uf": "SC",
        "cep": "XXXX-XXX"
```

```
                    },
                {
                  "codigo": XXXX,
                  "situacao": "ALUGADO",
                  "agencia": 1,
                  "rua": "RUA XXXXXX XXXXX XXX",
                  "numero": "38",
                  "lojaAp": "null",
                  "cidade": "FLORIANÓPOLIS",
                  "bairro": "ESTREITO",
                  "uf": "SC",
                  "cep": "XXXX-XXX"
                        },
                }
        }
```


Fonte: O autor, 2019.

Esse exemplo é de um caso atípico, em que o usuário é locatário de mais de um imóvel e também é dono de imóveis, informação que está oculta na imagem para não a deixar exageradamente grande. Além de encurtar o JSON de resposta, informações mais delicadas foram trocadas pela letra “X”.

Esse exemplo é o motivo pelo o qual existe um desafio maior em retornar o imóvel que o usuário deseja abrir o chamado, pois o CPF ou CNPJ pode estar relacionado a vários imóveis. Como queremos uma melhor experiência para o usuário, tem-se duas opções para o prosseguimento do f*low*, uma é que retornamos imediatamente o código do imóvel que está associado ao CPF ou CNPJ nos casos em que só existe um imóvel, ou criamos uma lista de código de imóveis e apresentamos ela para o usuário escolher para qual imóvel ele deseja abrir o chamado.

Na Figura 16, temos uma sequência de *scripts* que foram arquitetados para a extração das informações do JSON de resposta da requisição HTTP anterior. Estes *scripts* tem o objetivo de extrair, verificar e tratar as informações para se decidir o rumo do *flow*.

Figura 16 – Algoritmo de extração de informação dos imóveis.

```
function run(inputVariable1) {

   var result = JSON.parse(inputVariable1)
   var tamClienteProprietario = result.clienteProprietario.length
   var tamImoveis
   var u = 0
```

```
   var i = 0
   var uAnterior = 0
   var respostas = "Ótimo, localizei seu cadastro! Segue a lista dos imóveis com seus
respectivos endereços:\n"
   var rua = null

   for (; i < tamClienteProprietario; i++) {

       tamImoveis = result.clienteProprietario[i].imoveis.length

       for (u = 0; u < tamImoveis; u++) {

           rua = result.clienteProprietario[i].imoveis[u].rua

           respostas      =      respostas      +           "-      Código:      "      +
result.clienteProprietario[i].imoveis[u].codigo
           + ", endereço: " + rua + '\n'

       }
       uAnterior = uAnterior + u
   }

   if(respostas == "Ótimo, localizei seu cadastro! Segue a lista dos imóveis com seus
respectivos endereços:\n" ){

       respostas = "vazia"
   }

   return respostas;
}
```

Fonte: O autor, 2019.

O *script* Pega só Código, que é o *script* responsável por criar um vetor sequencial com os códigos dos imóveis relacionados com o CPF ou CNPJ dados (Figura 17). Essa informação será utilizada na sequência do fluxo para verificar se o código informado pelo usuário corresponde a algum código da lista de imóveis retiradas do JSON.

O pega só Agências é semelhante ao Pega só Código, ele cria um vetor sequencial só com os códigos das agências que cada imóvel é relacionado. Essa informação é utilizada mais à frente para a criação do Chamado no CRM Bitrix.

Por se tratar de um código muito extenso, e muito semelhante ao anterior, não será anexado neste documento.

O *script* seguinte, Quantidade de Imóveis (Figura 18), é apenas para medir o tamanho do vetor que contém os códigos de imóveis. Se caso o vetor tenha apenas um código de imóvel, a tag de retorno do script é “direto”, e caso tenha mais de um

imóvel é “indireto”. Essas tags são utilizadas nas condições de saída para encaminhar o usuário para o rumo certo no *flow*, ou seja, o resultado do primeiro script onde criamos a lista de imóveis será utilizado apenas se o CPF ou CNPJ tenha mais de 1 imóvel associado.

Figura 17 - Script Pega só Código, é responsável por criar um vetor sequencial com os códigos dos imóveis relacionados com o CPF ou CNPJ dados.

```
function run(inputVariable1) {

   var result = JSON.parse(inputVariable1)
   var tamClienteProprietario = result.clienteProprietario.length
   var tamImoveis
   var imoveis = []
   var u = 0
   var i = 0
   var uAnterior = 0

   for (; i < tamClienteProprietario; i++) {

       tamImoveis = result.clienteProprietario[i].imoveis.length

       for (u = 0; u < tamImoveis; u++) {

           imoveis[u + uAnterior] = result.clienteProprietario[i].imoveis[u].codigo

       }
       uAnterior = uAnterior + u
   }

   return imoveis
}
```

Fonte: O autor, 2019.

Semelhante ao Pega só Código e Pega só Agência, no Verifica Loc. ou Prop. é criado um vetor sequencial com a informação se o CPF ou CNPJ é dono ou locatário de cada imóvel da lista. Essa informação é relevante apenas nos Chamados Financeiros. Os Chamados Financeiros serão explicados mais adiante neste documento, e eles compartilham todos esses blocos em que é pedido o CPF ou CNPJ do usuário para se retornar um código de imóvel para a abertura do Chamado.

Figura 18 - Quantidade de Imóveis, mede o tamanho do vetor que cotem os códigos de imóveis.

```
function run(inputVariable1) {

   var codigoImovel = JSON.parse(inputVariable1)
   var tamanho = codigoImovel.length
   var vaiPara = "indireto"

   if(tamanho <= 1){

       vaiPara = "direto"
   }

   return vaiPara;
}
```

Fonte: O autor, 2019.

A Próxima ação executada é a retornaImoveisManutencao == vazia. Essa ação cria uma mensagem para avisar para o usuário que o CPF ou CNPJ informado não tem nem um imóvel associado a ele. Lembrando que é uma situação diferente da anterior, pois apesar de o status da resposta da requisição HTTP indicar sucesso, não existe nem um imóvel para retornar com a informação dada.

A penúltima ação é uma requisição HTTP (Figura 19). A ação Consulta ID é uma requisição do tipo POST para o CRM da imobiliária, o Bitrix, em que procuramos o ID de contato do usuário no CRM utilizando como chave de pesquisa o CPF ou CNPJ passado, essa informação é utilizada mais à frente na criação do chamado no próprio Bitrix.

Figura 19 – Consulta ID do usuário no Bitrix.

```
{
   "order": {
       "STATUS_ID": "ASC"
   },
   "filter": {
       "UF_CRM_1554913746": "{{respostaLinkImoveisCPF@documento}}"
   },
   "select": [
       "ID"
   ]
}
```

Fonte: O autor, 2019.

Por último temos uma ação bem simples, que extrai a informação ID no Bitrix da resposta da requisição HTTP anterior.

Figura 20 – Scrip que salva a informação retornada na requisição anterior

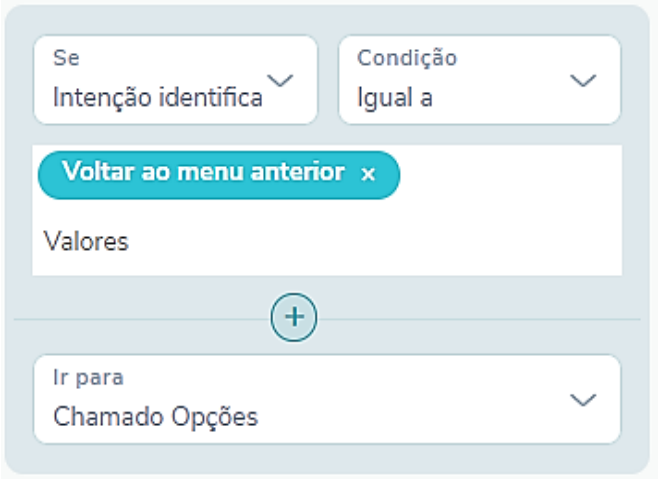
```
function run(inputVariable1) {

   var resultado = JSON.parse(inputVariable1)
   var id

   if(resultado.total != 0) {

       id = resultado.result[0].ID

   } else {

       id = ""

   }

   return id
}
```

Fonte: O autor, 2019.

Com isso temos o fim das ações executadas no bloco. Agora que se tem diversas informações que foram retiradas das requisições HTTP a partir da informação dada pelo usuário, pode-se encaminhar o usuário para o caminho certo no *flow*.

As condições de saída (Figura 21). serão explicadas brevemente para se entender a lógica utilizada para o encaminhamento do flow.

Figura 21 – Condição de saída 1 do bloco “Chamado - Pede CPF”.

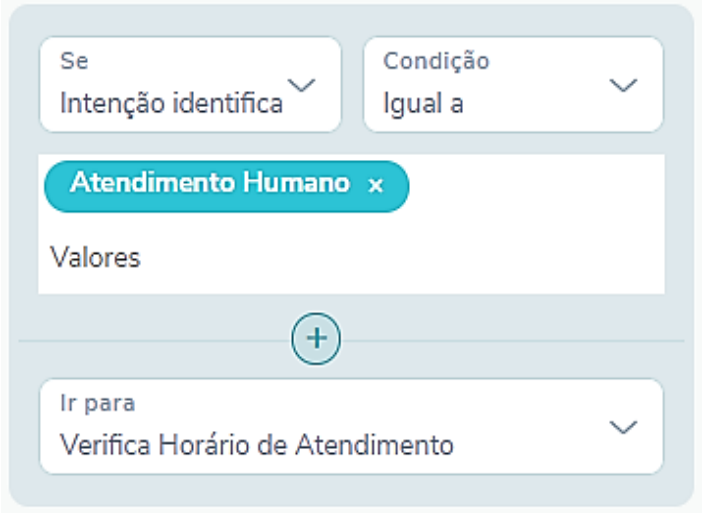


Fonte: O autor, 2019.

Essa condição de saída é baseada na identificação de intenção na entrada do usuário. No *bot*, existe uma NLP integrada que faz o reconhecimento de intenções dentro das conversas. Essa NLP foi treinada antes da realização deste módulo por outro membro da equipe, e por isso não será abordada mais a fundo nesse trabalho. A intenção que se deseja

identificar é a “Voltar menu anterior”, que seria uma intenção que indicaria que o usuário quer sair desse *flow* e voltar ao *flow* principal do bot. E essa condição de saída faz exatamente isso caso a condição seja atendida, como observado na Figura 22.

Figura 22 – Condição de saída 2 do bloco “Chamado - Pede CPF”.



Fonte: O autor, 2019.

Nessa condição de saída (Figura 23), analogicamente a anterior, se procura identificar uma intenção na entrada do usuário, que no caso é a de falar com um atendente. Caso a condição seja satisfeita, o usuário é encaminhado para o atendimento humano.

A lógica dessa condição de saída é verificar se o usuário está preso nesse bloco, tendo de responder mais de três vezes a mesma pergunta (Figura 24). Caso as condições sejam atendidas, o usuário é encaminhado para o atendimento humano para evitar maiores frustrações.

A Figura 25 mostra a condição de saída, onde é atendida quando a primeira requisição HTTP tem o status de sucesso e quando o CPF ou CNPJ tem mais de um imóvel associado ao mesmo. Ela encaminha a conversa para um bloco em que o usuário deve escolher para qual imóvel ele deseja abrir um chamado antes de continuar o fluxo de preenchimento do formulário.

Figura 23 - Condição de saída 3 do bloco “Chamado - Pede CPF”.
Fonte: O autor, 2019.

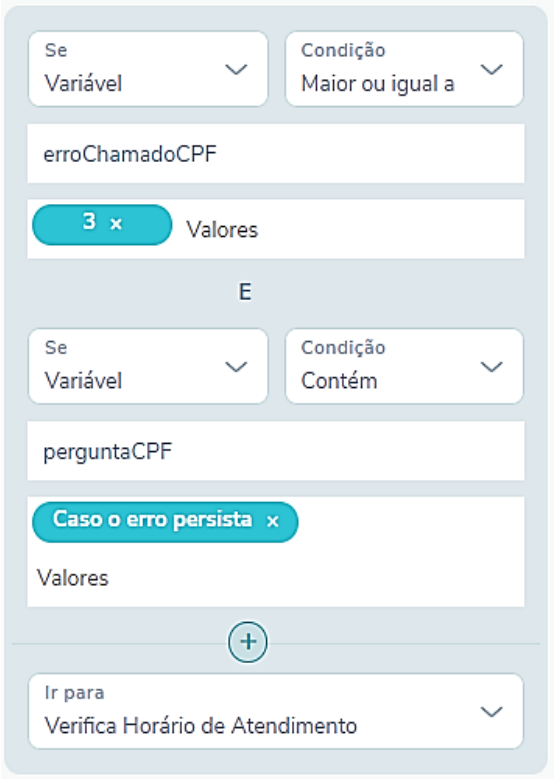
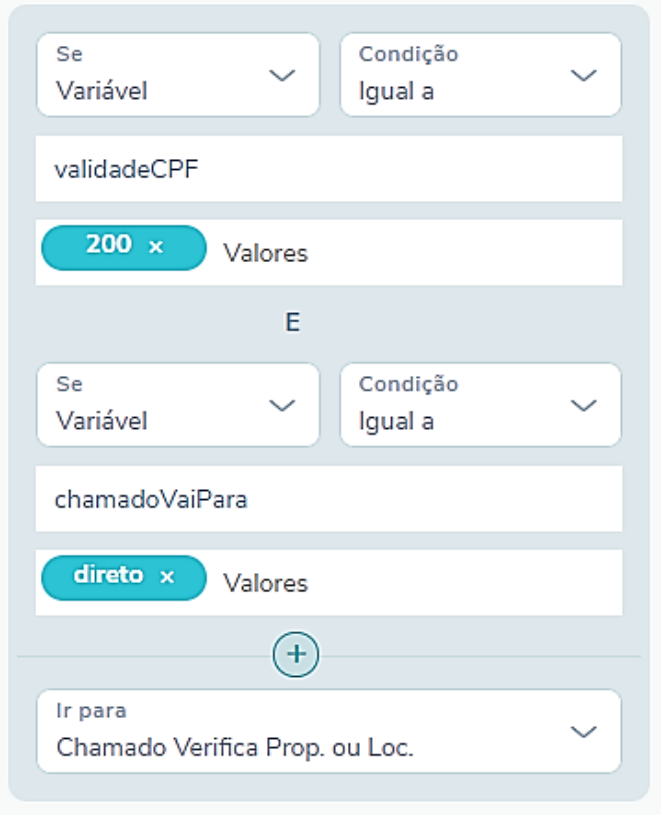


Figura 24 - Condição de saída 4 do bloco “Chamado - Pede CPF”.

Fonte: O autor, 2019.

Figura 25 - Condição de saída 5 do bloco “Chamado - Pede CPF”.
Fonte: O autor, 2019.



Analogamente a anterior, essa condição verifica se o status da resposta da requisição HTTP foi de sucesso e ainda verifica se o CPF ou CNPJ tem apenas um imóvel associado a ele. Caso as condições sejam cumpridas, o usuário é encaminhado para um bloco em que ele é informado o código do imóvel que ele irá abrir o chamado e apresentado ao primeiro menu de opções do formulário de abertura de chamado.

III. Chamado - Pega Código imóvel

Como alguns blocos na sequência do flow apresentam apenas informações para o usuário e salvam a resposta do mesmo para uso posterior, sem nem um tratamento muito profundo, será mostrado apenas o menu principal e 1 dos submenus do formulário de abertura de chamado de Manutenção.

O bloco “Chamado - Pega Código imóvel” apresenta um menu de opções para que o usuário escolha as categorias que melhor se enquadram com o seu chamado (Figura 26).

Figura 26 - Menu de opções para que o usuário escolha as categorias que melhor se enquadram com o seu chamado.

Fonte: O autor, 2019.

Como pode ser visto na Figura 27, a entrada do usuário é salva na variável "categoriaChamadoManutencao", essa informação é utilizada mais à frente no *flow* para a abertura do chamado. A entrada do usuário também determina para qual submenu ou bloco ele será encaminhado dependendo de condições de saída. Por se tratar de condições repetitivas, será mostrada apenas uma condição de saída.

Figura 27 - Condição de saída 1 do bloco “Chamado - Pega Código

Fonte: O autor, 2019.

Como pode ser visto, se dá algumas alternativas para que a entrada do usuário corresponda a algum item do menu. Mais à frente essa entrada será tratada para ser usada para a abertura do chamado no CRM Bitrix.

Outras condições de saída como detecção de intenção de voltar e intenção de falar com o atendente também estão presentes nos blocos com os menus e submenus de preenchimento do formulário.

IV. Manutenção Chamado Descrição

Após o usuário navegar pelos menus e submenus do formulário e encontrar a categoria ou categoria e subcategoria do seu chamado, ele é encaminhado para o bloco Manutenção Chamado Descrição. Este bloco, apesar de parecer bem simples contendo apenas uma pergunta “Em uma única mensagem, informe a descrição da sua solicitação.”, uma saída e um script, é de suma importância. A sua importância está na ação chamada Traduz para o Bitrix. como observado na Figura 28.

A ação traduz para o Bitrix tem a função de traduzir as respostas dadas pelo usuário no formulário para os códigos correspondentes dos itens nos campos da requisição HTTP para abertura de chamados no Bitrix.

As os itens deste objeto são utilizados para o preenchimento dos dados na requisição HTTP (Figura 29).

Figura 28 - Trecho do código com a lógica de substituição.

```
        function run(inputVariable1, inputVariable2) {

            var mCC = inputVariable1 //manutencaoChamadoCategoria
            var mCSC = inputVariable2 //manutencaoChamadoSubCategoria
            var r1 = ""
            var sr2 = ""
            var sr3 = ""
            var sr4 = ""
            var sr5 = ""
            var sr6 = ""
            var sr7 = ""

            if (mCC == "1" || mCC == "1-" || mCC == "1-Imóvel sem Energia ou Água" || mCC == "Imóvel
sem Energia ou Água") {

                r1 = "1003"

            } else if (mCC == "2-Eletro doméstico" || mCC == "2." || mCC == "2-" || mCC == "Eletro
doméstico") {

                r1 = "995"

                if (mCSC == "1-Geladeira" || mCSC == "1-" || mCSC == "1" || mCSC == "Geladeira") {

                    sr2 = "1015"

                } else if (mCSC == "2-Ar Condicionado" || mCSC == "2-" || mCSC == "2" || mCSC == "Ar
Condicionado") {

                    sr2 = "1013"

                } else if (mCSC == "3-Outros" || mCSC == "3-" || mCSC == "3" || mCSC == "Outros") {

                    sr2 = "1017"
```

Fonte: O autor, 2019.

Figura 29 - No fim deste script as informações são salvas em um objeto.

```
        var objManutencaoChamado = {
                "r1": r1,
                "sr2": sr2,
                "sr3": sr3,
                "sr4": sr4,
                "sr5": sr5,
                "sr6": sr6,
                "sr7": sr7
            }
```

Fonte: O autor, 2019.

V. Chamado Locatário Salva Arquivo

Este bloco é o responsável por salvar as URLS dos arquivos anexados ao chamado pelo usuário. A plataforma BLiP tem uma função nativa muito interessante, ela possibilita configurar um servidor para receber arquivos nas conversas com os usuários do bot. Essa função é utilizada neste bloco para se salvar todos os arquivos que o usuário desejar anexar ao chamado para ajudar a justificar a sua solicitação.

As URLS salvas são enviadas para o Bitrix na requisição HTTP de abertura de chamado. O script é bem simples, cria uma lista sequencial das URLS dos arquivos, como observado na Figura 30.

Figura 30 – Lista Sequencial das URLS dos arquivos.

```
function run(inputVariable1,inputVariable2) {

   var arquivo = JSON.parse(inputVariable1)
   var urls = inputVariable2

       urls = urls + '\\n'+ '\\n'+ arquivo.uri

   return urls;
}
```

Fonte: O autor, 2019.

VI. Chamado - Cria Deal

Este apesar de não ser o último bloco do *flow*, será o último a ser explicado. Isso se deve ao fato de os outros blocos são apenas complementares e não exercem funções novas ou diferentes das anteriores já explicadas.

O que é importante neste bloco é a requisição HTTP do tipo POST feita para o CRM Bitrix da imobiliária. A url da requisição não pode ser mostrada por conter informações críticas, mas a função utilizada é a "crm.deal.add". Essa função cria um "deal" com as informações do preenchimento do formulário de abertura de chamado no pipeline de chamados no CRM para ser tratado pelo setor responsável posteriormente.

No JSON podemos ver os nomes dos campos do formulário no bitrix, e os respectivos valores que foram adquiridos durante o *flow* da abertura do chamado (Figura 31).

Figura 31 - JSON com as informações enviadas para o Bitrix.

```
{
   "fields": {
       "CATEGORY_ID": "1",
       "CONTACT_ID": "{{IDContatoBitrix}}",
       "SOURCE_ID": "26",
       "TITLE": "{{manutencaoCodigoImovel}}",
       "UF_CRM_1554898210": "{{respostaLinkImoveisCPF@documento}}",
       "UF_CRM_5C6D511934A76": "{{manutencaoChamadoAgencia}}",
       "UF_CRM_5C6D511978E6E": "{{manutencaoCodigoImovel}}",
       "UF_CRM_5C6D51198ECAD": "{{objManutencaoChamado@r1}}",
```

```
            "UF_CRM_5C6D5119A3D88": "{{objManutencaoChamado@sr2}}",
            "UF_CRM_5C6D5119B63C8": "{{objManutencaoChamado@sr3}}",
            "UF_CRM_5C6D5119C012C": "{{objManutencaoChamado@sr4}}",
            "UF_CRM_5C6D51199A596": "{{objManutencaoChamado@sr5}}",
            "UF_CRM_5C6D5119ABEEC": "{{objManutencaoChamado@sr6}}",
            "UF_CRM_5C6D5119CA65E": "{{objManutencaoChamado@sr7}}",
            "UF_CRM_1569963809": "{{urls}}",
            "UF_CRM_5C6D511946B56": "{{manutencaoChamadoDescricao}}"
        },
        "params": {
            "REGISTER_SONET_EVENT": "N"
        }
    }
```

Fonte: O autor, 2019.

6.2 ABERTURA DE CHAMADOS FINANCEIROS LOCATÁRIO E PROPRIETÁRIO.

Essa função, apesar de ter suas particularidades, é bastante análoga a função de Abertura de Chamados de Manutenção, portanto para evitar repetição será apresentada apenas a particularidade dessa função.

A particularidade, que já foi mencionada anteriormente, é que durante o *flow* em que se retorna os imóveis para um dado CPF ou CNPJ, já se é verificado se o usuário é proprietário ou locatário. Essa informação é importante para se direcionar o usuário para o formulário correto, pois existem formulários distintos para a abertura de chamados financeiros para proprietários e locatários. A maneira que foi realizada essa verificação torna imperceptível para o usuário a existência de 2 formulários distintos, e torna a experi6encia mais fluida e intuitiva.

Nos exemplos será mostrado logs de interações reais do usuário com as funções, onde ficará mais clara as diferenças entre os formulários.

Outro aspecto dos fluxos de abertura de chamados, é que apesar de serem programados pensando na experiência do usuário, eles estão presos aos formulários previamente elaborados. Isso torna a experiência do usuário um pouco menos intuitiva.

6.1 AGENDAMENTO DE VISITAS.

Essa seção irá tratar de um aspecto de suma importância do chatbot, o contato ativo comercial com o cliente. O contexto deste assunto é o seguinte: Todos

os anúncios de imóveis em sites parceiros da imobiliária encaminham os leads (clientes com interesse em alugar um imóvel) para entrar em contato com o chatbot. Esse contato é via mensagem ativa no app Whatsapp.

A jornada desse lead até o contato ativo do chatbot foi criado anteriormente ao início do trabalho do autor deste documento na imobiliário, porém será explicado em um âmbito geral o caminho destes leads até chegarem no módulo de agendamento de visitas do chatbot.

O primeiro passo é um possível cliente, ou lead, peça para contatado a respeito de um imóvel que ele teve interesse em algum site de anúncio ou no próprio site da imobiliária. Quando o lead passa seus dados, e-mail, nome, telefone e aceita ser contactado a respeito do imóvel, o site parceiro envia essas informações para o e-mail da imobiliária. Estas informações são, automaticamente, extraídas dos e- mails pelo programa "mailparse". Este programa é especializado na extração de informações de e-mails. Com estas informações, por meio de integrações com o CRM Bitrix, é criado um Lead dentro do mesmo. Dentro do Bitrix existem automações que enviam as informações deste lead para uma plataforma de integração de sistemas, o integromat.

No integromat, por meio de algoritmos e requisições HTTP, os dados do lead são verificados e usados para se enviar uma mensagem ativa para o lead via Whatsapp. Dentre as ações presentes no integromat, duas são as mais importantes para o processo, a verificação se o número telefônico passado pelo lead está no Whatsapp, e a outra é se o lead não é duplicado de um mesmo usuário, ou seja, se ele não pediu múltiplas vezes para ser contactado a respeito de um imóvel.

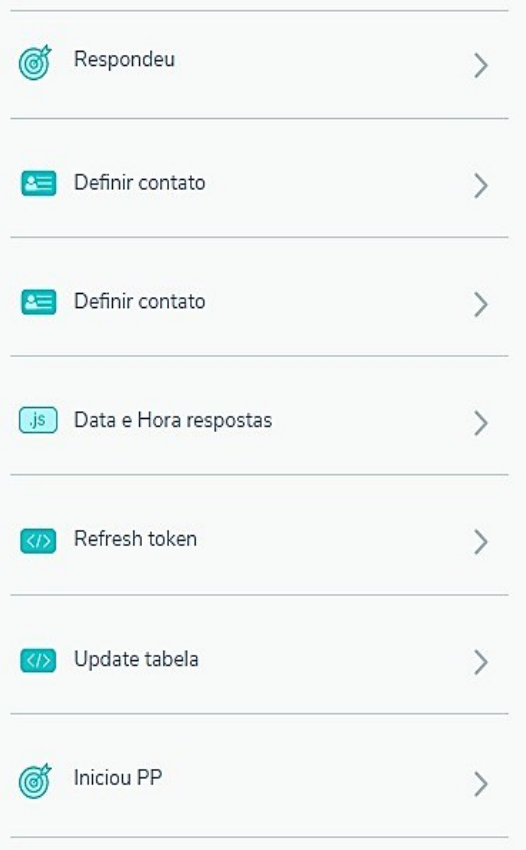
Com isso, com todas as verificações feitas e o usuário ou lead é contactado via Whatsapp. A partir deste momento se inicia a jornada de Agendamento de visita. A seguir será apresentado os principais blocos onde residem os scripts e integrações da jornada. O primeiro bloco é:

i. WhatsApp Ativo - Portais Parceiros

Este primeiro bloco é bastante importante no processo todo de agendamento de visitas, além de ser o bloco receptivo para os leads, também o bloco em que se detecta as intenções do lead no contato com o chatbot.

Primeiramente serão apresentadas e explicadas todas as Ações realizadas neste bloco (Figura 32).

Figura 32 – Ações realizadas no Bloco “WhatsApp Ativo - Portais



Fonte: O autor, 2019.

A primeira e a última ações, Respondeu e Iniciou PP, são ações de registro de eventos que são utilizadas para as métricas do chatbot, para quantificar e mapear o andamento do módulo de agendamentos.

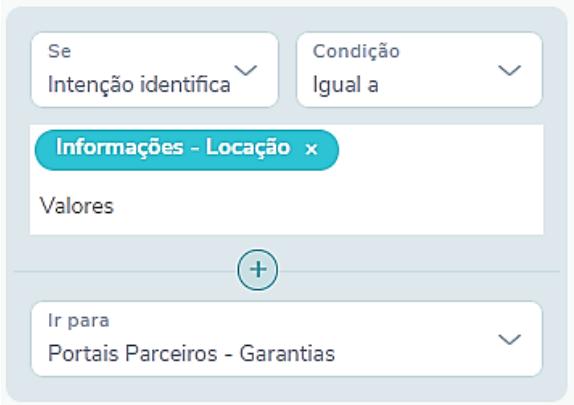
As ações Definir contato que estão na segunda e terceira posições são responsáveis por criar, caso o usuário não exista ainda, o usuário ou atualizar os dados do mesmo.

Data e Hora é um script simples para retornar a data e a hora do momento em que o usuário respondeu o chatbot pela primeira vez depois da mensagem ativa. A quinta ação, Refresh token, é uma requisição HTTP do tipo POST com URL https://accounts.google.com/o/oauth2/token. Esta requisição é um gerador de token para se poder consumir algumas apis do google. Esse token serão utilizados logo a seguir na ação seguinte.

A ação Update tabela é uma requisição HTTP do tipo POST para a alimentação de uma tabela do google sheets com as informações do lead que respondeu a mensagem ativa.

Agora que já está explicada a aba de ações deste bloco, será direcionada a atenção para a aba Condições de Saída (Figura 33).

Figura 33 – Condição de Saída 1 do bloco “WhatsApp Ativo - Portais

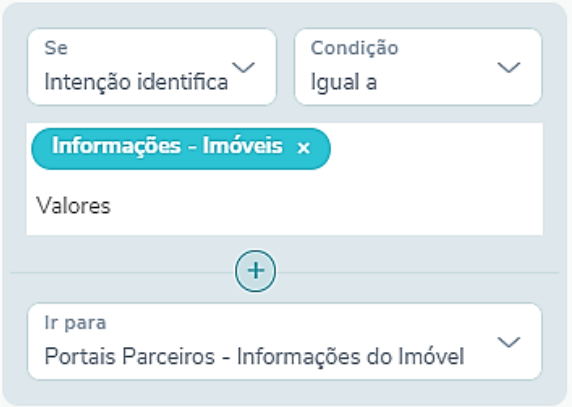


Fonte: O autor, 2019.

A primeira condição de saída tem o objetivo de identificar a intenção do usuário querer saber mais sobre as garantias e os requisitos para se alugar com um imóvel com a Brognoli. Caso seja detectada esta intenção, o usuário é encaminhado para o bloco “Portais Parceiros - Garantias”, onde ele receberá um link para uma página com todas as condições e requisitos de garantia para aluguel.

Esta condição de saída verifica a detecção de intenção na entrada do usuário de receber mais informações sobre o imóvel em que ele teve interesse (Figura 34). Se a condição for preenchida, o usuário será encaminhado para o bloco “Portais Parceiros - Informações do Imóvel”, onde ele receberá diversas informações sobre o imóvel, esse bloco por se tratar de um bloco com uma complexidade maior, será explicado mais adiante neste documento.

Figura 34 – Condição de Saída 2 do bloco “WhatsApp Ativo - Portais



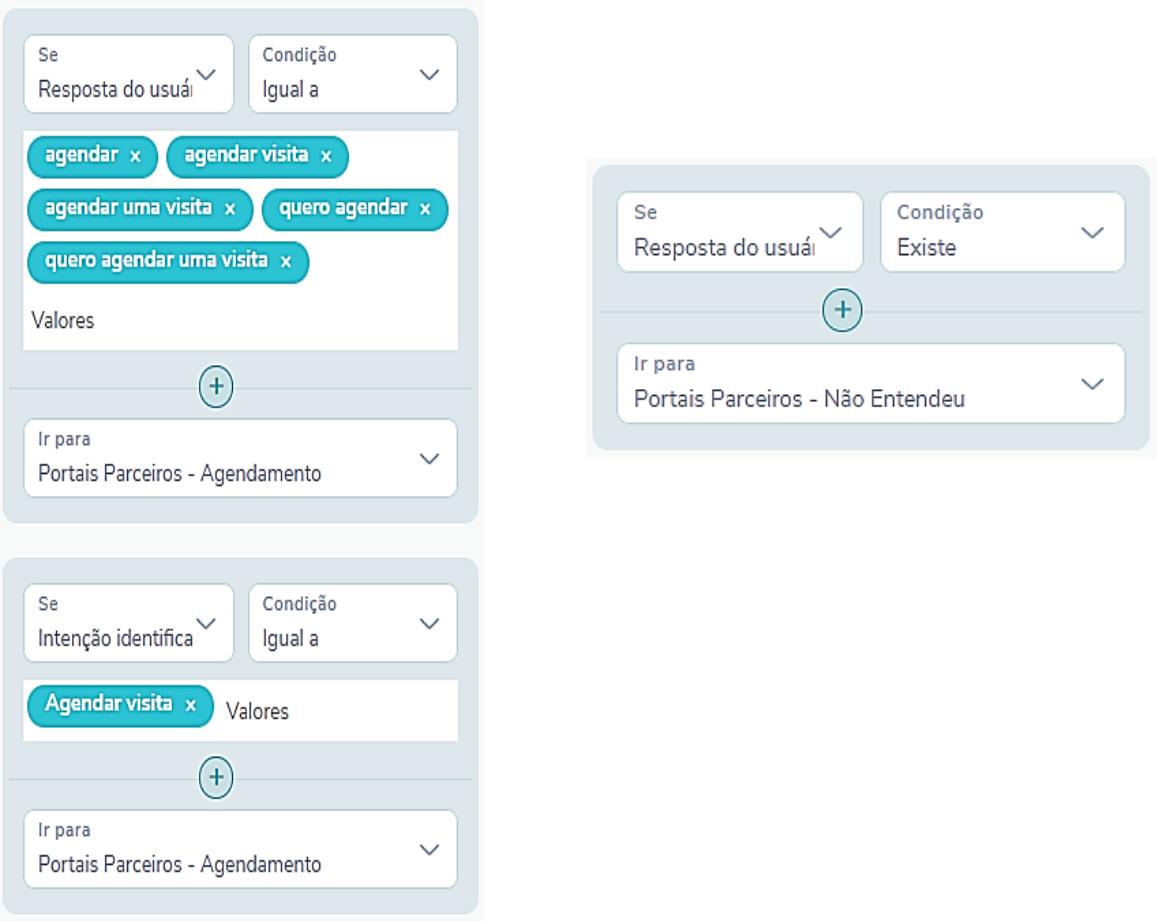
Fonte: O autor, 2019.

Estes dois blocos de condições de saída tem o mesmo objetivo, identificar a intenção na entrada do usuário de se agendar uma visita para o imóvel que ela demonstrou interesse (Figura 35). Estas duas condições de saída encaminham o usuário para o bloco

“Portais Parceiros - Agendamento”. Mais à frente este bloco será destrinchado em detalhes.

O último caminho de saída do bloco é um sem condição. Ele está por último para tratar o usuário que escreveu algo que não pode ser interpretado pela NLP. Esta saída encaminha o usuário para um menu de opções com as alternativas dos serviços disponíveis no módulo de agendamento de visitas. Este menu fica no bloco “Portais Parceiros - Não Entendeu”.

Figura 35- Condição de Saída 3 e 4, respectivamente do bloco “WhatsApp Ativo - Portais Parceiros”.



Fonte: O autor, 2019.

ii. Portais Parceiros - Informações do Imóvel.

Este bloco é destinado apenas para a execução de ações, a seguir temos uma imagem com todas as ações executadas no bloco (Figura 36).

A primeira ação do bloco, Consulta imóvel Bitrix, é uma requisição HTTP do tipo Post no CRM Bitrix. Esta requisição retorna informações de um imóvel de acordo com o código passado. No caso o código é do imóvel que o lead tem intenção de visitar. As diversas informações retornadas desta ação são tratadas mais a frente para serem apresentadas para o usuário.

Figura 36 - Condição de Saída 5 do bloco “WhatsApp Ativo - Portais Parceiros”.

Fonte: O autor, 2019.

Na Próxima ação, o script Trata dados do Imóvel, é tratada toda a informação vinda da requisição anterior. O código é bastante extenso e executa várias ações diferente, tendo mais de 290 linhas, portanto será apresentada apenas o JSON resultante desta ação (Figura 37).

Figura 37 – Objeto que contém as informações do produto.

```
var infoProduto = {

        "rua":  rua, "numero":
        numero, "bairro":
        bairro, "cidade":
        cidade, "agencia":
        agencia, "endereco":
        endereco, "apto": apto,
        "descricao": descricao,
        "bloco": bloco, "andar":
        andar,
        "agenciaLat": agenciaLat,
        "agenciaLng": agenciaLng,
        "valorAluguel": valorAluguel,
        "condominio": condominio,
        "iptu": iptu,
        "enderecoFormatado": enderecoFormatado,
        "codImovel": codImovel,
        "informacoes": informacoes,
        "agenciaEndereco": agenciaEndereco,
        "valorTotal": valorTotal,
        "agenciaFrase": agenciaFrase
    }
```

Fonte: O autor, 2019.

Estas informações serão utilizadas para montar respostas para o usuário e executar algumas ações para melhor informar o usuário.

Na ação seguinte, “Localização”, a informação “endereco” do JSON criado na ação anterior, é utilizada para se realizar uma requisição HTTP do tipo GET para a api do google maps para se pegar as informações de longitude e latitude do imóvel. Estas informações serão utilizadas mais a frente para se marcar a localização do imóvel no google maps para mandar para o usuário.

Como a API do google maps utilizada anteriormente retorna além das informações de latitude e longitude do imóvel, também o endereço formatado e padronizado, porém com “Brazil” com “Z” e a informação do país é irrelevante e óbvia, o script Endereço do Imóvel substitui a informação “Brazil” por “.” e a salva em uma variável para ser utilizado mais à frente no fluxo para ser apresentado para o cliente.

Figura 38 – Algoritmo de formatação de endereço

```
function run(inputVariable1) {

   var GPS = JSON.parse(inputVariable1);

   var enderecoFormatado = GPS.results[0].formatted_address

   enderecoFormatado = enderecoFormatado.replace(", Brazil", ".")

   return enderecoFormatado;
}
```

Fonte: O autor, 2019.

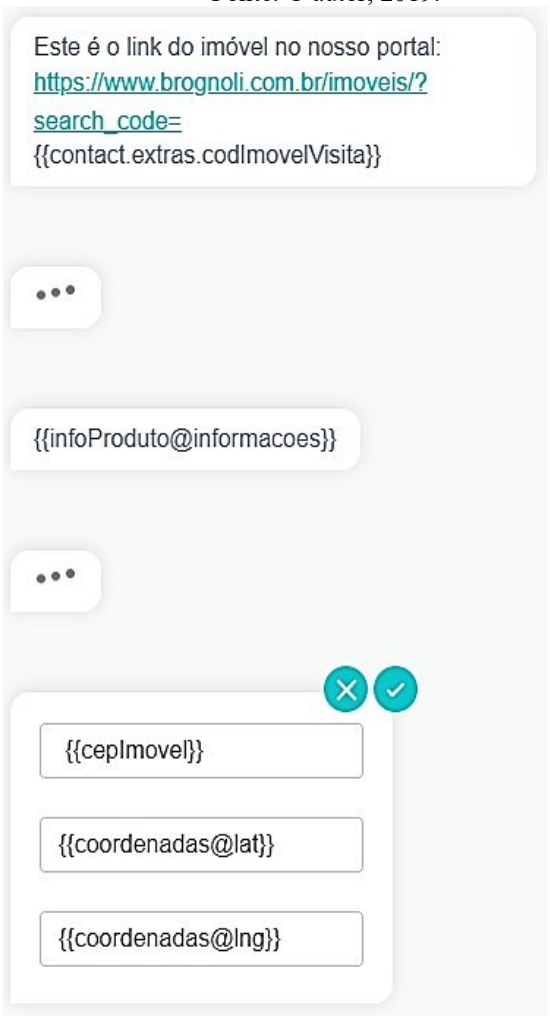
A última ação do bloco é apenas um registro de evento, que registra que o usuário pediu mais detalhes ou informações do imóvel.

iii. Portais Parceiro - Informações do Imóvel Sucesso

Este bloco não contém nem uma ação, porém ele entrega diversas informações diretamente para o usuário (Figura 39).

A primeira informação entregue é a URL do imóvel no site da Brognoli. A segunda informação é uma variável contendo "Este é o endereço do imóvel (" + codImovel + ") que você gostou.", que veio direto do script em que foram tratadas as informações do imóvel.

Figura 39 – Bloco que entrega diversas informações para o usuário.
Fonte: O autor, 2019.



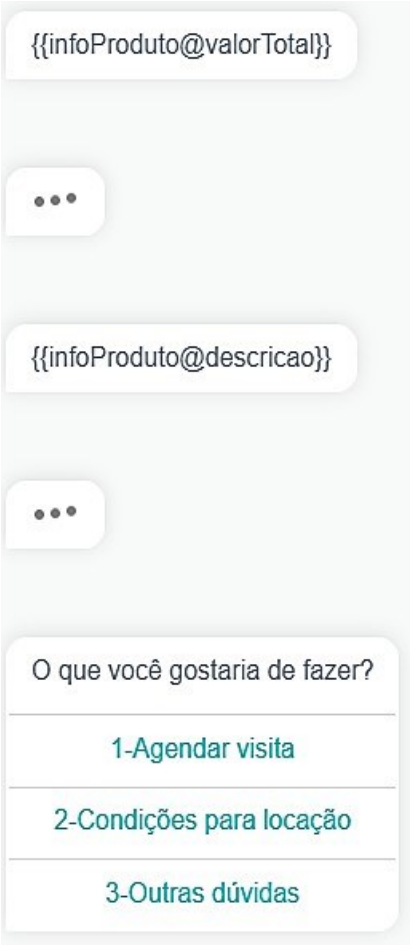
A terceira informação é a localização do imóvel no googlemaps. Esta função é uma função nativa do Blip, basta se passar três parâmetros: Legenda, que deve conter alguma explicação ou título para o mapa, que no caso é o próprio endereço formatada do imóvel e as coordenadas de latitude e longitude do imóvel. Com isso temos um mapa gerado automaticamente indicando o caminho do imóvel para o cliente.

A seguir ainda é apresentada uma frase personalizada de acordo com as informações disponíveis de valor de aluguel, condomínio e IPTU. Ela se adapta de acordo com as infos que tem, para tornar ela mais natural, ela é criada anteriormente no script de tratamento das informações do imóvel.

Por penúltimo se tem a variável contendo a descrição do imóvel registrada no Bitrix. E finalmente temos o menu de opções para encaminhar o usuário para 3 blocos diferentes (Figura 40). As duas primeiras alternativas são autoexplicativas e a última encaminha o usuário para o atendimento humano. Nas condições de saída do bloco a NLP é

usada de maneira semelhante ao bloco “WhatsApp Ativo Portais Parceiros” para auxiliar o menu de opções a encaminhar o usuário para o destino correto.

Figura 40 – Menu de opções para encaminhar o usuário para 3 blocos.



Fonte: O autor, 2019.

iv. Agendamento - Data e Hora

Este bloco serve apenas para pedir a data e a hora que o usuário deseja marcar a visita ao imóvel e salvar em uma variável (Figura 41). Esta informação será utilizada no próximo bloco para consultar a API do Dialogflow.

Figura 41 – Bloco para o usuário marcar visita no imóvel.

Fonte: O autor, 2019.

v. Agendamento - Tratamento DialogFlow

No desenvolvimento deste módulo, resolveu-se utilizar a NLP do google DialogFlow para se detectar a entidade de sistema @date-time, que é nativa da NLP. A API do google foi a escolhida para a tarefa pois foi a que respondeu melhor às entradas que foram dadas, que simulavam entradas reais dos usuários. Isto foi feito para tornar mais natural o ato de informar a data e a hora desejadas.

Figura 42 – Ações do bloco “Agendamento - Tratamento DialogFlow”.

Fonte: O autor, 2019.

A primeira ação já foi explicada anteriormente, é a requisição HTTP do google para a renovação de token de autenticação oauth2.0. Este token é utilizado na ação “Dialogflow request”.

Na Dialogflow request, que é uma requisição HTTP do tipo POST, que tem a função de identificar a entidade @date-time na entrada do usuário, retorna uma resposta em um JSON que é tratado mais à frente.

O script “Interpreta resposta DF” serve para interpretar as informações vindas no JSON de resposta do DialogFlow (Figura 43). No script é identificado se o usuário escreveu as duas informações, data e hora, ou só data, ou só hora, ou período, ou ainda se não mandou nem uma destas informações.

Figura 43 - O formato do objeto de saída do script “Interpreta resposta.

```
var objAgendamento = {

    "d": data,
    "h": hora,
    "r": resposta,
    "asw": answer,
    "qst": question

}
```

Fonte: O autor, 2019.

No script “Verifica Disponibilidade do Agendamento”, a data e hora (Figura 44), ou só a data é verificada de acordo com alguns parâmetros, para se criar respostas personalizadas de acordo com a situação. Os parâmetros de verificação são os horários de atendimento das agências, que são onde o cliente deve buscar a chave para realizar a visita ao imóvel. Existem respostas personalizadas para algumas situações como observado nas Figura 44, 45, 46, 47, 48,49 e 50.

Figura 44 - Data e hora dentro do horário de atendimento das agências.

```
answer = "Ótimo! Entendi que você quer agendar no dia " + ddmmhhmm + "."
question = "Posso confirmar?"
```

Fonte: O autor, 2019.

Figura 45 – Respostas personalizadas para o domingo.

```
answer = "Desculpe, mas agendamos visitas apenas de segunda a sábado, e a data informada é um domingo."
question = "Por favor, nos informe outro dia e horario de sua preferência."
```

Fonte: O autor, 2019.

Figura 46 – Respostas personalizadas para sábado fora de horário.

```
answer = "Desculpe, não conseguimos agendar uma visita nesse horário no sábado. Neste dia, realizamos visitas das 09:00 às 12:00. "
question = "Por favor, informe outro horário para esta mesma data, ou uma nova data e horário de agendamento."
```

Fonte: O autor, 2019.

Figura 47 – Respostas personalizadas para o dia de semana fora do

```
answer = "Desculpe, não conseguimos agendar neste horário. As visitas de segunda a sexta acontecem das 08:30 às 17:30."
question = "Por favor, informe outro horário para esta mesma data ou uma nova data e horário de agendamento."
```

Fonte: O autor, 2019.

Figura 48 – Respostas personalizadas para data que já passou.

```
answer = "Acredito que ocorreu um engano, a data informada já passou."
question = "Por favor, nos informe outro dia e horário de sua preferência."
```

Fonte: O autor, 2019.

Figura 49 – Respostas personalizadas quando o usuário informa só a data e é um dia de semana.

```
answer = "Entendi que você quer agendar sua visita no dia " + ddmm + ". De segunda a sexta as visitas podem ser agendadas das 08:30 às 17:30."
question = "Por favor, informe um horário para esta mesma data ou uma nova data e horário de agendamento."
```

Fonte: O autor, 2019.

Figura 50 – Respostas personalizadas quando o usuário informa só a data

```
answer = "Entendi que você quer agendar sua visita no dia " + ddmm + ", um sábado. Aos sábados o hoário de visita é das 09:00 às 12:00."
question = "Por favor, informe um horário para esta mesma data ou uma nova data e horário de agendamento."
```

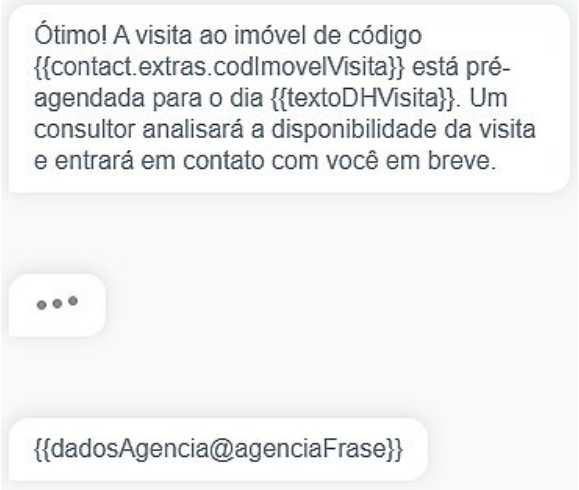
Fonte: O autor, 2019.

Estas respostas personalizadas são utilizadas para pedir novamente as informações de data e hora, ou só hora para o caso de o usuário ter colocado uma data válida.

vi. Portais Parceiros - Visita Agendada

Este bloco é responsável por registrar as informações do agendamento no Bitrix e informar o usuário de que a visita ao imóvel está pré-agendada. A seguir se tem as informações e textos contidos na aba Conteúdo do bloco (Figura 51).

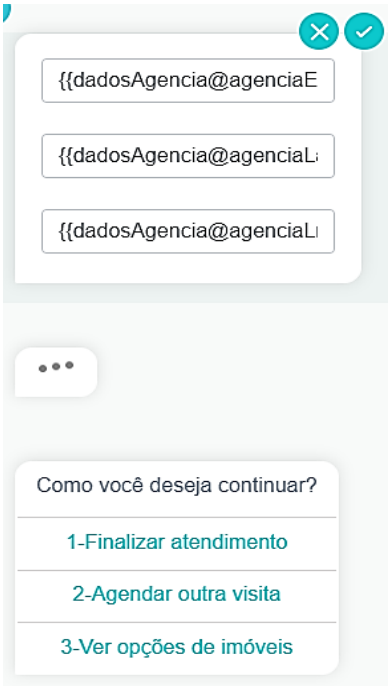
Figura 51 - Conteúdo 1 do bloco Portais Parceiros - Visita Agendada.



Fonte: O autor, 2019.

Primeiramente confirma os o imóvel, data e hora do agendamento, depois informa que será analisada agenda para se verificar a disponibilidade do imóvel para a visita no horário escolhido. A seguir na Figura 52 temos uma frase criada de acordo com a agência que o imóvel é administrado, informando que o usuário deve retirar as chaves na agência para realizar a visita.

Figura 52 - Conteúdo 2 do bloco Portais Parceiros - Visita Agendada.



Fonte: O autor, 2019.

6.1.1 Exemplo Abertura de Chamado

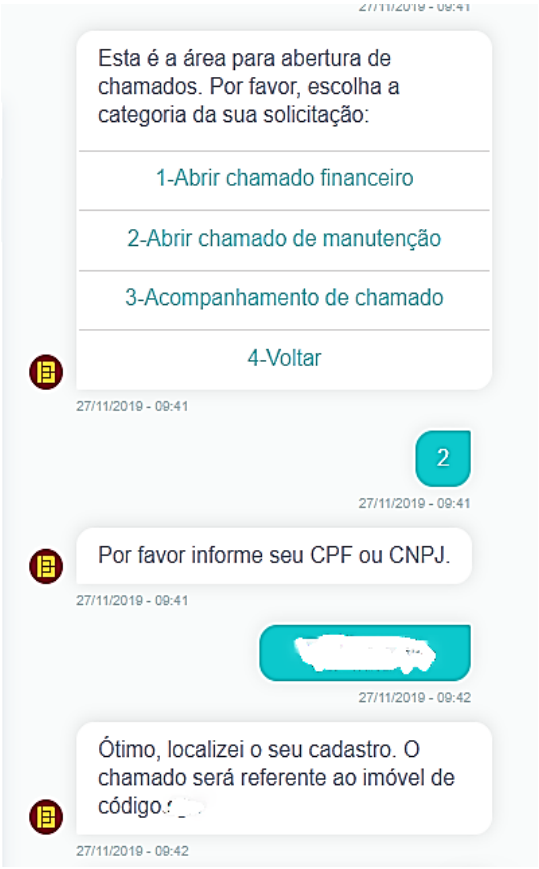
Será apresentada nesta seção um exemplo um exemplo de abertura de chamado. Será apresentada apenas para um dos módulos, pois os outros dois funcionam de maneira semelhante. A seguir temos imagens as imagens do log de uma interação real com o chatbot (Figura 53).

Figura 53 - Log exemplo abertura de chamado fig. 1.

Fonte: O autor, 2019.

Este menu de opções apresenta as funcionalidades que estão presentes no chatbot e o usuário selecionou a opção 4, abertura de chamados (Figura 54).

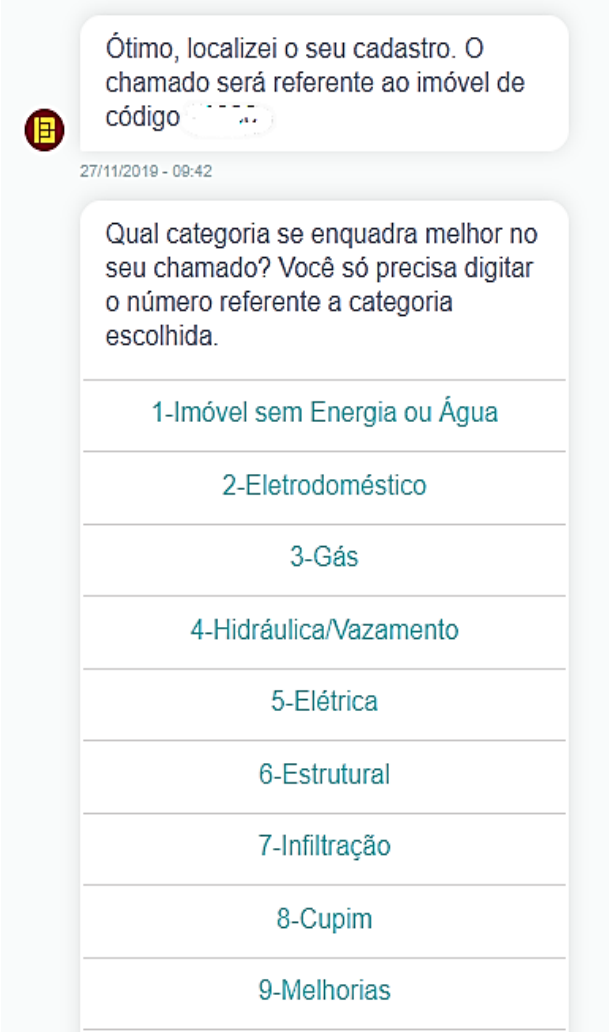
Figura 54 - Log exemplo abertura de chamado fig. 2.



Fonte: O autor, 2019.

Na sequência temos o início da jornada do usuário na abertura do chamado, a imagem foi alterada para esconder informações do usuário (Figura 55).

Figura 55 - Log exemplo abertura de chamado fig. 3.



Fonte: O autor, 2019.

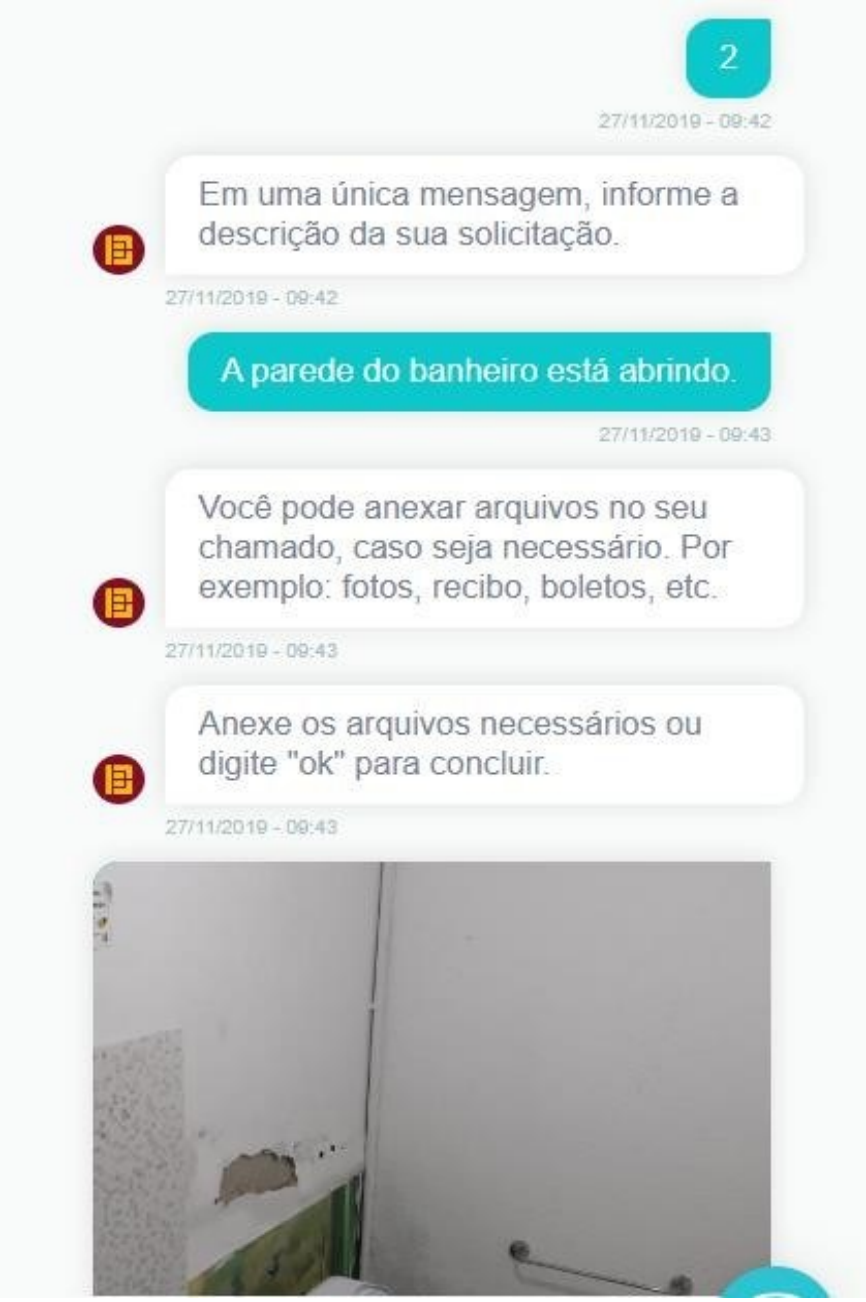
Como pode ser observado na Figura 56, o usuário tem apenas um imóvel. Essa possibilidade foi tratada na criação dos algoritmos para o *flow* e com isso podemos pular a etapa de o usuário ter de informar o código do imóvel para o qual o chamado será referente.

Figura 56 - Log exemplo abertura de chamado fig. 4

Fonte: O autor, 2019.

Nesta sequência vemos o usuário preenchendo o formulário, escolhendo categoria e subcategoria do chamado que deseja abrir (Figura 57).

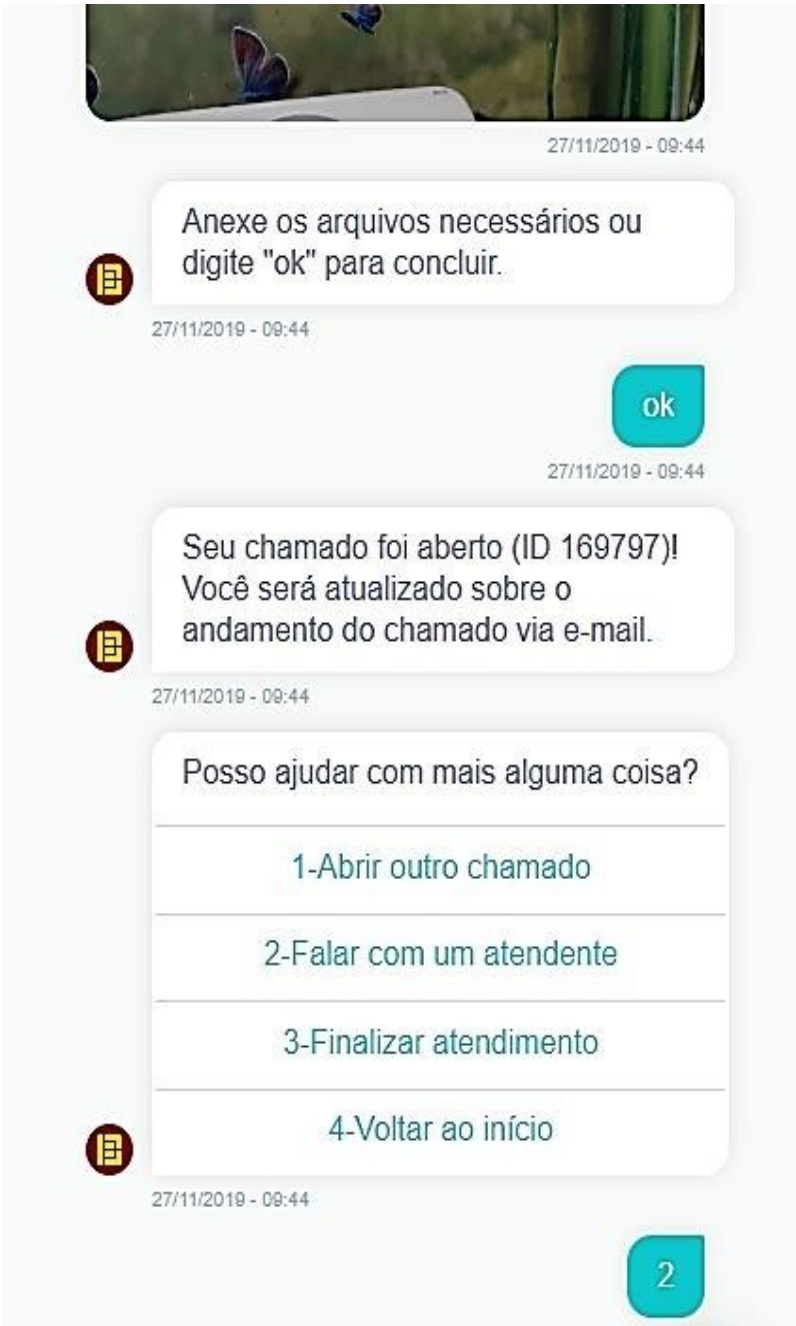
Figura 57 - Log exemplo abertura de chamado fig. 5



Fonte: O autor, 2019.

A Figura 58 apresenta na sequência os campos de descrição do chamado e de anexos. O usuário fez uma breve descrição e anexou algumas imagens para ilustrar sua solicitação.

Figura 58 - Log exemplo abertura de chamado fig. 6



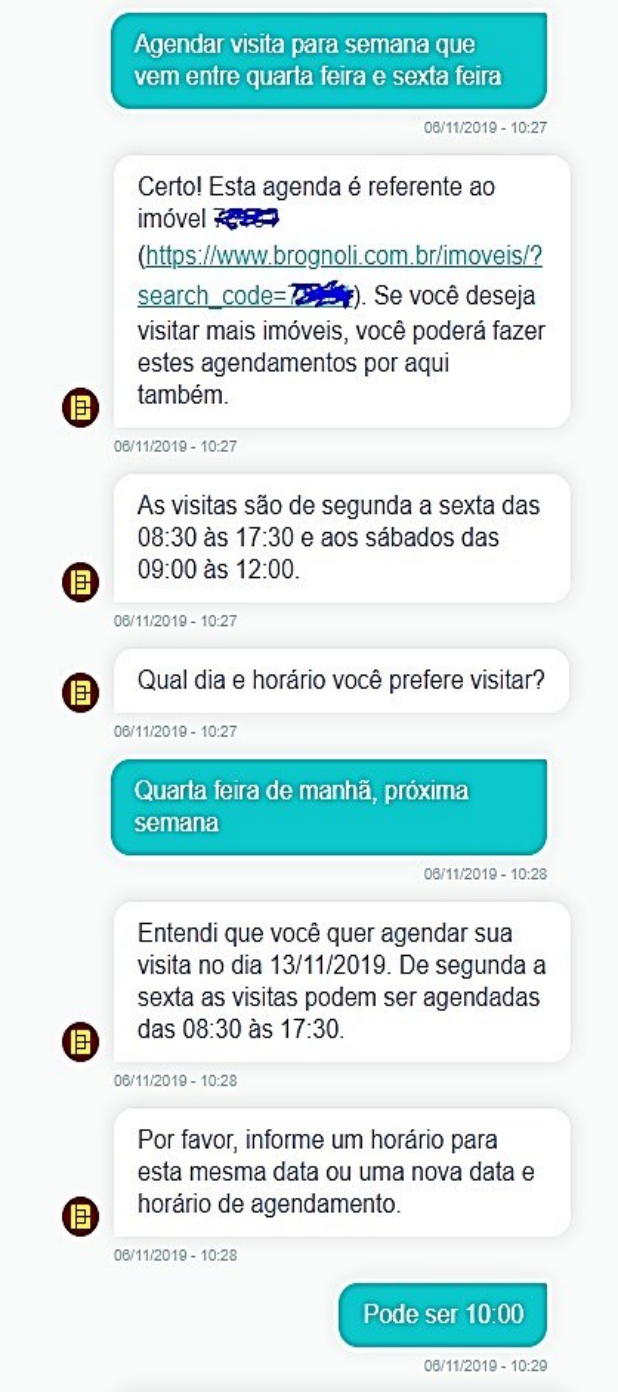
Fonte: O autor, 2019.

Por fim, apesar de ter conseguido abrir o chamado e ter em mãos o ID do chamado para consulta, o usuário ainda resolveu falar com um atendente. Na sequência o chatbot encaminhou o usuário para ser atendido por um colaborador humano. Como foge do escopo do objetivo do exemplo e muitas informações pessoais são passadas, a conversa com o atendente foi omitida neste documento.

6.1.2 Exemplo Agendamento de Visita

A seguir temos o exemplo real de um agendamento realizado totalmente no chatbot apresentado nas Figura 59 e 60. O usuário recebeu o contato ativo via whatsapp e seguiu o seguinte *flow*.

Figura 59 - Log exemplo de agendamento de visita fig. 1.



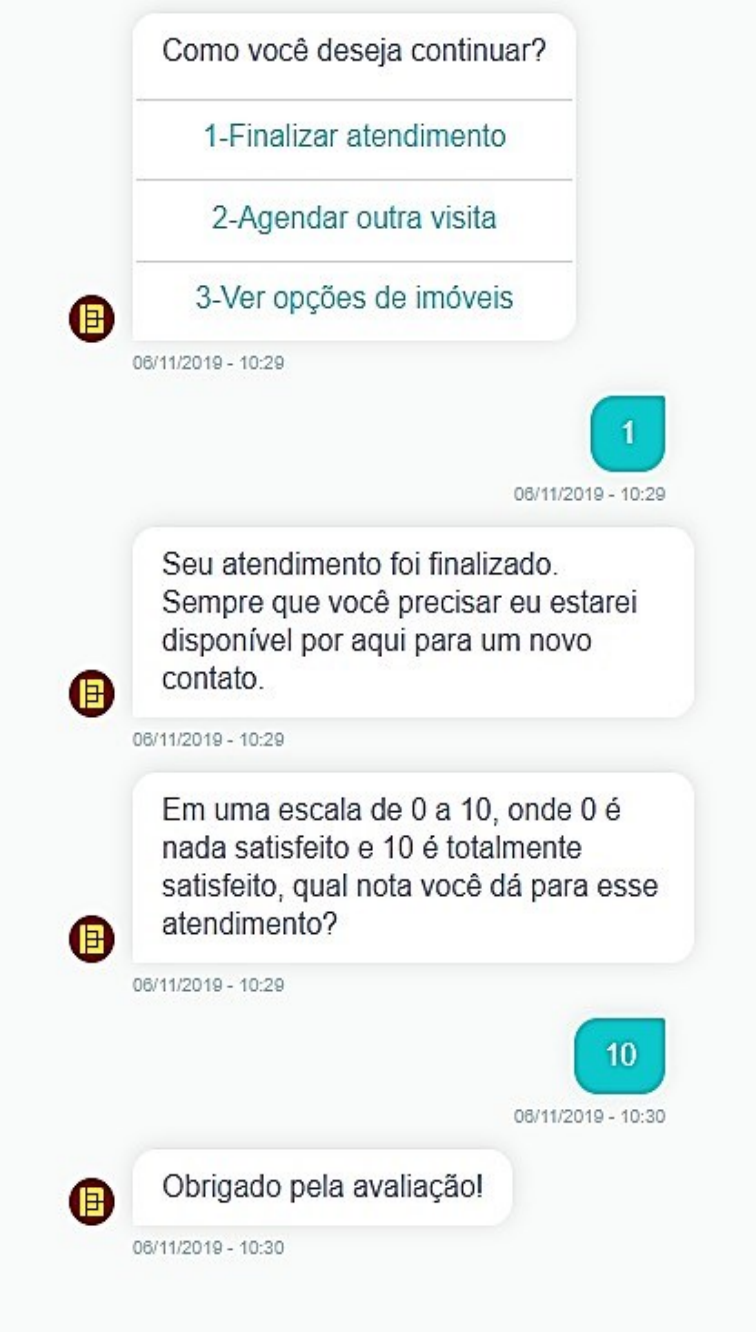
Fonte: O autor, 2019.

Figura 60 - Log exemplo de agendamento de visita fig. 2

Fonte: O autor, 2019.

Este exemplo se trata de um agendamento bem simples, onde o usuário estava focado em agendar a visita de maneira rápida. Existem outros exemplos com uma interação entre chatbot e usuário maiores, porém estenderia demasiadamente esta seção do documento.

Figura 61 - Log exemplo de agendamento de visita fig. 3



Fonte: O autor, 2019.

7 ANÁLISE DE RESULTADOS

Primeiramente será apresentada uma análise breve dos módulos de abertura de chamados de manutenção e financeiro.

Os números não são tão expressivos quanto os números apresentados no módulo de agendamento, mas para o contexto de ele não ter sido divulgado e depender de o usuário procurar essa funcionalidade dentro do bot, é de certa maneira impressionante.

Os números são de uma média de 10 aberturas de chamados por semana, totalizando 62 chamados abertos no último mês e meio, tempo que o módulo está no ar. Destes 62, 45 usuários ainda tinham dúvidas e quiseram falar com o atendente.

Analisando a eficiência e a quantidade de usuários que encontraram a funcionalidade sem divulgação, pode se considerar um sucesso a implementação do módulo. Um lado não tão eficiente do módulo é o fato de não passar confiança o suficiente para o cliente em certos casos, pois o mesmo ainda quer uma confirmação de um atendente.

Agora será analisado o grande sucesso deste projeto, o módulo de agendamento de visitas automático. A seguir temos números e estatísticas deste módulo (os números são referentes aos 2 meses e meio que o módulo está ativo).

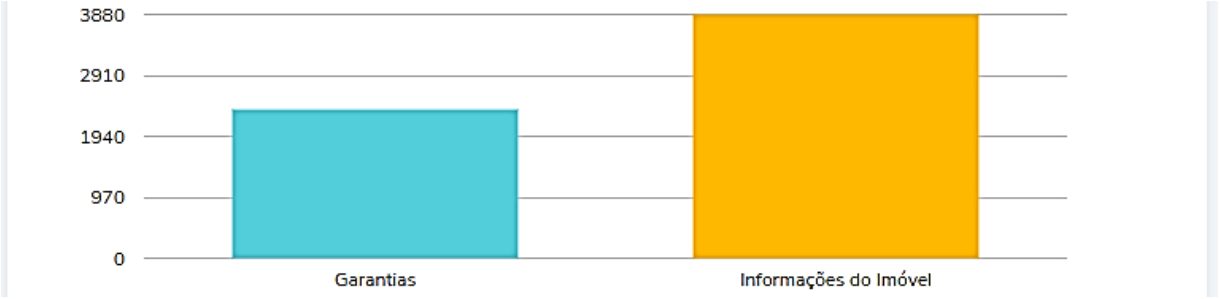
Na Figura 62 temos os números referentes aos contatos ativos com *leads* vindos dos portais parceiros, bem como a quantidade de leads respondeu à mensagem ativa. Podemos observar que é um número expressivo de clientes que responderam o chatbot.

Na sequência temos os números referentes aos tipos de perguntas que o chatbot recebeu dos usuários neste módulo (Figura 63). Vemos que mais da metade dos usuários que respondeu ao chatbot, pediu mais informações e teve uma interação com o mesmo.

Figura 62 - Gráfico de estatística de mensagens ativas.

Fonte: O autor, 2019.

Figura 63 - Gráfico de estatística de perguntas.



Fonte: O autor, 2019.

Na Figura 64 temos os números efetivos das três intenções mais importantes do módulo. Como podemos ver, um número expressivo de leads conversou com o chatbot com a intenção de agendar uma visita.

Figura 64 - Número e tipo de respostas.

Resposta Portais Parceiros

Ações	%	Total
Mais informações atendente	39.83%	1,962
Quer agendar	42.43%	2,090
Mais informações	17.74%	874

Fonte: O autor, 2019.

Por fim temos os números referentes aos agendamentos efetivos realizados com a ajuda ou totalmente pelo chatbot (Figura 65). Como podemos observar, cerca de 10% dos leads vindos dos sites parceiros se converteram em visitas, o que é um número expressivo para este tipo de contato.

Figura 65 - Número de agendamentos.

Portais Parceiros Agendamento

Ações	%	Total
Agendamento com atendente	52.63%	750
Agendamento com o bot	47.37%	675

Fonte: O autor, 2019.

8 CONSIDERAÇÕES FINAIS E PERSPECTIVAS

Este projeto que foi realizado para a imobiliária Brognoli e também para servir como Projeto de Fim de Curso do autor deste documento, pode ser considerado um sucesso. Isso se deve ao fato de estar ativo e funcionando, atendendo milhares de clientes todos os meses.

Com o sucesso do chatbot as perspectivas de crescimento do projeto são enormes. Já foi aprovada a criação de uma versão 2.0 do chatbot. Esta versão conta com uma equipe maior, e mais diversa funcionalmente, de desenvolvimento, que é composta agora por 7 membros. Com esta equipe, o foco passou a ser em refazer o bot do zero pensando além de funcionalmente, a parte mais de engenharia, pensar também no UX e no sentimento e personalidade do chatbot.

O módulo de agendamento de visitas teve um resultado tão expressivo que outras imobiliárias de outras regiões do país estão entrando em acordo com a Brognoli para o desenvolvimento de seus próprios chatbots de agendamento de visitas.

Por fim o autor deste documento conclui que este projeto e a contratação na imobiliária Brognoli foram uma grande oportunidade de se desenvolver profissionalmente e aplicar todo o conhecimento e lógica adquiridos durante o decorrer do curso de Engenharia de Controle e Automação. O Projeto Final de Curso realmente representa uma ótima oportunidade de experimentar como é estar inserido no mercado de trabalho como um engenheiro.

REFERÊNCIAS

BENITTI, Fabiane Barreto Vavassori; RHODEN, Jaqueline Sezra. UMA TAXONOMIA UNIFICADA PARA REQUISITOS NÃO FUNCIONAIS. Revista
Electronica de Sistemas de Informaçao**, v. 14, n. 3, p. 1, 2015.**

DEMO, Pedro. **Pesquisa e construção de conhecimento**. Rio de Janeiro: Tempo Brasileiro, 1996.

JÄRVINEN, P. **Action Research is Similar to Design Science**. Quality & Quantity, n. 41, Springer. 2007, p.37-54.

JÄRVINEN, P. **On Research Methods. Tampere**, Finlândia: Opinpajan Kirja 2004.

LEONHARDT, Michelle; NEISSE, Ricardo, TAROUCO; ROCKENBACH, Liane Margarida. Meara: **Um Chatterbot temático para uso em ambiente educacional**. Porto Alegre, 2003.

NEVES; André M. M.; BARROS, Flávia A. **iAIML: Um Mecanismo para Tratamento de Intenção em Chatterbots**. 2005. XVIII Encontro Nacional de Inteligência Artificial. São Leopoldo.

RABELO, R.J.; ROMERO, D; ZAMBIASI, S.P. **Softbots Supporting the Operator 4.0 at Smart Factory Environments**. 2018. Moon I., Lee G., Park J., Kiritsis D., von Cieminski G. (eds) Advances in Production Management Systems. Smart Manufacturing for Industry 4.0. APMS 2018. IFIP Advances in Information and Communication Technology, vol 536. Springer, Cham.

TECHLAB, Maruti. **How do Chatbots work? A Guide to the Chatbot Architecture.** Disponível em: <https://marutitech.com/chatbots-work-guide-chatbot-architecture/>. Acesso em 21 de novembro de 2019.

TEIXEIRA, S.; RAMIRO, T. B.; OLIVEIRA, E. de; MENEZES, C. S. de. **Chatterbots em ambientes de aprendizagem - uma proposta para a**

construção de bases de conhecimento. In: Anais do Workshop de Informática na Escola. [S.l.: s.n.], 2005. v. 1, n. 1.

VAVASSORI BENITTI, F. B.; RHODEN, J. S. **Uma Taxonomia Unificada Para Requisitos Não Funcionais**. Revista Eletrônica de Sistemas de Informação, [s. l.], v. 14, n. 3, p. 1–15, 2015. Disponível em: <https://search.ebscohost.com/login.aspx?direct=true&db=foh&AN=118916329&lang=pt-br&site=eds-live>. Acesso em: 24 nov. 2019.

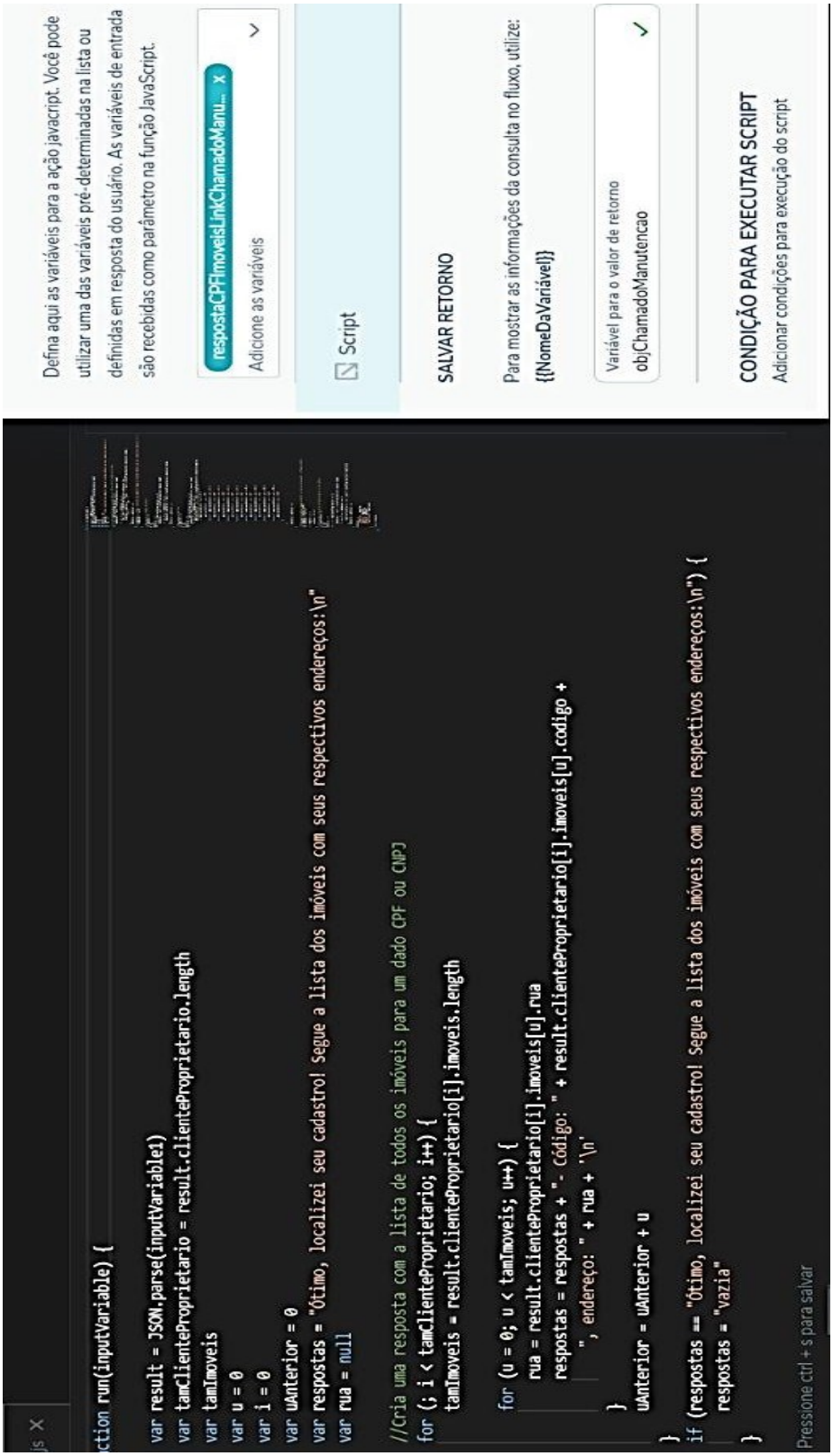
VAZQUEZ, C. E. Análise de pontos de função: medição, estimativas e gerenciamento de projetos de software. [s. l.], 2013.

WALLACE, Richard S. **AIML Overview**. 2008. Disponível em: <https://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html> Acesso em: 20 de novembro de 2019.

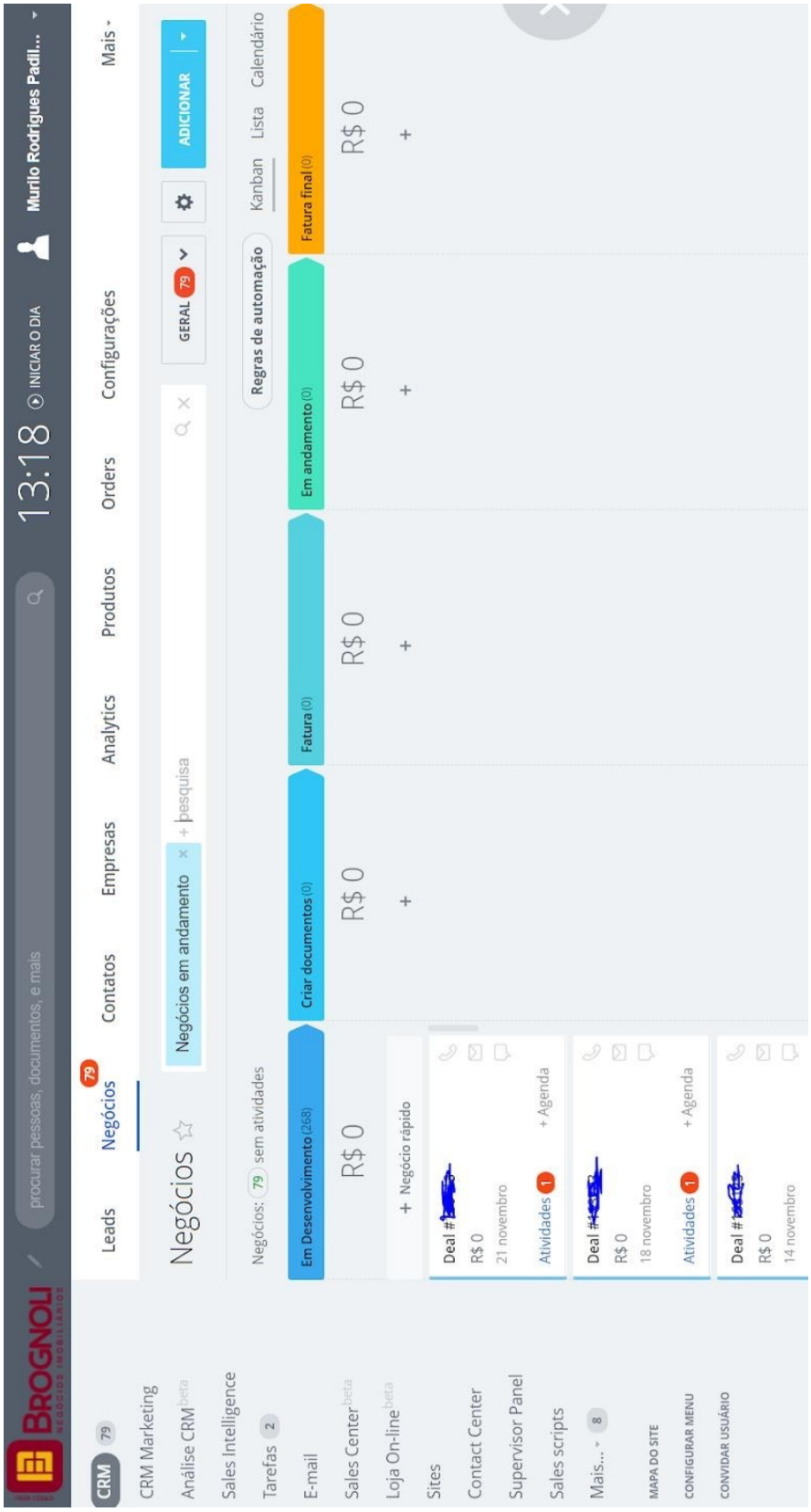
WEIZENBAUM, J. **Eliza - a computer program for the study of natural language communication between man and machine**. Communications of the ACM, ACM, v. 9, n. 1, p. 36–45, 1966.

ZAMBIASI, Saulo P.; RABELO, R. J. **A Proposal for Reference Architecture for Personal Assistant Software Based on SOA**. Revista IEEE América Latina, v. 10, p. 1227-1234, 2012.

ANEXO A - SCRIPT E SUA INTERFACE DA AÇÃO EXECUTAR SCRIPT NO BLOCO



ANEXO B – PÁGINA INICIAL DA PLATAFORMA BRITRIX



ANEXO C - IMAGEM DOS BLOCOS QUE FORMAM ESSA SEÇÃO DO BOT