Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Diretrizes para a construção de um agente conversacional: Estudo de caso do desenvolvimento da *chatbot* Tais

Autores: Bruna Pinos de Oliveira, Guilherme Guimarães Lacerda

Orientador: Prof. Dra. Carla Silva Rocha Aguiar

Brasília, DF

2020

Bruna Pinos de Oliveira, Guilherme Guimarães Lacerda

Diretrizes para a construção de um agente conversacional: Estudo de caso do desenvolvimento da *chatbot* Tais

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dra. Carla Silva Rocha Aguiar

Brasília, DF

2020

Bruna Pinos de Oliveira, Guilherme Guimarães Lacerda

Diretrizes para a construção de um agente conversacional: Estudo de caso do desenvolvimento da *chatbot* Tais/ Bruna Pinos de Oliveira, Guilherme Guimarães Lacerda. – Brasília, DF, 2020-

118 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dra. Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2020.

1. inovação. 2. chatbot. 3. diretrizes. I. Prof. Dra. Carla Silva Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Diretrizes para a construção de um agente conversacional: Estudo de caso do desenvolvimento da *chatbot* Tais

CDU 02:141:005.6

Errata

Bruna Pinos de Oliveira, Guilherme Guimarães Lacerda

Diretrizes para a construção de um agente conversacional: Estudo de caso do desenvolvimento da *chatbot* Tais

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 18 de dezembro de 2020:

Prof. Dra. Carla Silva Rocha Aguiar
Orientador

Prof. Dr. Renato Coral Sampaio
Convidado 1

M.e. Ricardo Augusto Poppi Martins
Convidado 2

Rodrigo Maia Pereira
Convidado 3

Brasília, DF
2020

Agradecimentos

Bruna Pinos de Oliveira

Não tem como deixar de agradecer em primeiro lugar minha família, a base e a parte mais preciosa da minha vida. Todo agradecimento a eles é pouco. Aos meus pais, Valéria e Jorge, agradeço por todo amor, carinho, empenho, e tudo o que me ensinaram durante toda a minha vida. Sempre me falaram que a educação me levaria aonde eu quisesse, e esse ciclo que fecho agora não seria possível sem o apoio deles.

Agradeço às minhas irmãs, Natália e Laura, por me escutarem, me apoiarem, e ajudarem a tomar decisões durante toda essa trajetória. Que um dia eu possa ajudá-las em suas graduações, como elas me ajudaram na minha. À minha avó Vanilda, por acreditar em todo meu potencial, me incentivando e me chamando de "minha futura doutora". Espero que ela me acompanhe nessa jornada acadêmica ainda mais.

Sou extremamente grata a todos os professores que tive o prazer de frequentar as aulas. Entrei no curso de Engenharia de Software não sabendo nada da área de tecnologia, apenas com a vontade de aprender, e foi com a ajuda deles que estou aqui concluindo o curso, e sabendo que o mundo ganhou mais um profissional competente. Muito obrigada por todo o conhecimento que tive a honra de receber de todos. Desejo um dia poder exercer o papel de lecionar tão bem quanto eles.

Aos amigos integrantes do LAPPIS, agradeço pela convivência durante os projetos, o conhecimento trocado em conversas informais, e ao meu primeiro contato com situações reais de projeto. Em seus momentos divertidos, essa equipe me mostrou como éramos amigos além do projeto que trabalhamos. Obrigada ao seniores do projeto por depositarem a confiança em mim e na minha capacidade de contribuir com o projeto.

Agradeço especialmente a minha orientadora, e professora Carla Rocha. Exemplo de mulher e de profissional, seu convite para participar do LAPPIS foi um marco na minha carreira e vida. Nunca vou conseguir agradecer por todo apoio, incentivo, carinho que me deu durante todo esse período. Com certeza uma pessoa que levarei para o resto da minha vida.

Por fim, agradeço aos meus amigos de graduação, àqueles que entraram comigo na universidade, mas seguiram caminhos diferentes, e os que estiveram comigo em trabalhos e projetos. Em particular, agradeço aos Guilherme Lacerda e Guilherme Augusto, cuja amizade me acompanha desde o inicio da graduação, e a Clarissa Borges pela companhia em todas as horas. Que eu possa contar com eles sempre em minha vida.

Saio desse ciclo com esperanças de que um dia estarei contribuindo com o apren-

dizado de pessoas, assim como cada um que fez parte do meu caminho até aqui.

Guilherme Guimarães Lacerda

Primeiramente agradeço imensamente aos meus pais, Nélia Gonçalves Guimarães e Wilmar Lacerda, e ao meu irmão, Luan Guimarães Lacerda, que tiveram um papel fundamental em toda minha trajetória, me servindo, diariamente, de base e inspiração para os caminhos, por mim, trilhados.

Deposito aqui minha gratidão a todos professores, que prestam o mais lindo serviço à população, o de ensinar. São grandes os desafios encarados todos os dias por esses guerreiros. Mesmo com todas barreiras impostas ainda desempenham um trabalho de excelência.

Agradeço também a todos meus amigos que estiveram comigo nesta fase, compartilhando momentos memoráveis e muito significativos para minha formação como pessoa e profissional, em especial aos meus grandes amigos Guilherme Augusto, Bruna Pinos, Vitor Falcão, Matheus Miranda e João Guilherme.

Por último, agradeço imensamente à minha orientadora e amiga, professora Carla Rocha, por todo seu jeito singular de ser e viver, sua empatia, paciência e amor pelo que faz. Me apoiando e guiando em diversos caminhos dentro e fora da Universidade.

É de extrema felicidade e com bastante responsabilidade que carrego comigo todos esses ensinamentos, de pessoas e profissionais tão especiais, compartilhando-os por onde passo.

Resumo

Chatbots, ou agentes conversacionais, são programas de computador que têm por sua finalidade a simulação de conversas humanas através de aplicações de mensageria. Podem ficar disponíveis 24 horas por dia e são capazes de responder uma gama de perguntas. Assim, é uma tecnologia que beneficia a governança digital. Seu foco é voltado para usuário, e simplifica as complexidades do serviço público.

Muitos elementos fazem parte para a construção de um *chatbot* capaz de manter uma conversa fluida com o usuário. Além da tecnologia de classificação de intenções, construir e testar fluxos de conversa, além de usar conhecimentos de especialistas em linguística são importantes para transformar o contexto do agente conversacional em diálogo. Um ponto a se observar é como o *chatbot* se comporta ao ser posto em produção, acompanhando os seus erros e comportamentos do usuário a fim de manter uma evolução constante.

O projeto de co-desenvolvimento Tais surge como solução para aplicação de governança digital na Secretaria Especial da Cultura, voltada à questões da Lei de Incentivo a Cultura. Além disso, é um exemplo de aplicabilidade dos elementos citados, presentes em suas 4 fases de desenvolvimento.

A partir deste projeto, o presente estudo tem como objetivo analisar o ciclo de desenvolvimento da *chatbot* Tais e elucidar diretrizes para construção de agentes conversacionais, que leva em consideração a sua manutenção e evolução.

Palavras-chave: *Chatbot*. Diretrizes. Governança Digital. e-Gov. Usabilidade. Rasa. Estudo de caso.

Abstract

Chatbots, or conversational agents, are computer programs designed to simulate human conversation by messaging applications. The chatbot is a service that has the characteristic of helping the user to perform some tasks through textual dialogues. Being able to be available 24 hours per day and is able to answer a range of questions. Thus, it is a technology that benefits digital governance. Its focus is user-oriented and simplifies the complexities of public service.

There are many elements to construct a chatbot capable of maintaining a fluid conversation with the user. In addition to the intent classification technology, building, and testing conversation flows, and using knowledge from linguistic experts are important to transform the context of the conversational agent into dialogue. We need to observe how the chatbot behaves when it is deployed to production, as well as its errors and user behavior in order to maintain a constant evolution.

The co-development project, Tais, emerges as a solution for the application of digital governance in the Brazilian Special Secretariat for Culture, which is focused on issues of the Culture Incentive Law. Also, the four stages of development of this project contain the elements mentioned.

Based on this project, the present study aims to analyze the development cycle of the chatbot Tais and elucidate guidelines for the construction of conversational agents, which takes into account its maintenance and evolution.

Key-words: Chatbot. Guidelines. Digital governance. e-Gov. Usability. Rasa. Case study.

Lista de ilustrações

Lista de tabelas

Lista de quadros

Lista de abreviaturas e siglas

ICT	Information and Communication Technologies
TIC	Tecnologia de Informação e Comunicação
PFIGP	Prêmio Federal de Inovação de Gestão Pública
CGS	Computer Generated Solutions
MIT	Massachusetts Institute of Technology (MIT) Artificial Intelligence Laboratory
NLP	Natural Language Process
NLU	Natural Language Understanding
DM	Dialog Management
GD	Gerenciador de diálogo
NLG	Natural Language Generation
SVM	Support-Vector Machine
TE	TensorFlow Embedding
MDP	Markov Decision Processes
DR	Dialog Register
AIML	Artificial Intellligence Markup Language
RNNs	Recurrente Neural Networks
LSTM	Long Short Term Memory Networks
TAIS	Tecnologia de Aprendizado Interativo do Salic
OSI	Open Source Initiative
FLOSS	Free/Libre and Open Source Software
SPB	Software Público Brasileiro
BI	Business Intelligence
FAQ	Frequently Asked Questions

LAPPIS	Laboratório Avançado de Pesquisa, Produção e Inovação em Software
Pronac	Programa Nacional de Apoio à Cultura
JSON	JavaScript Object Notation
XP	Extreme Programming
CRISP-DM	Cross-Industry Standard Process for Data Mining

Sumário

1 Introdução

O serviço público vem passando por um intenso processo de inovação tecnológica em variados campos e setores da sociedade, como aponta o estudo (BRASIL, 2017). Entende-se como inovação uma combinação de ferramentas de gestão existentes que impacte positivamente o serviço público e a sociedade, mudanças das práticas anteriores ou incorporação de novos elementos da administração pública (CAVALCANTE; CAMõES, 2017). Um estudo realizado no Brasil indicou que 70% das iniciativas finalistas do Prêmio Federal de Inovação de Gestão Pública (PFIGP), de 2007 a 2015, incluem em suas soluções, ferramentas tecnológicas para a inovação em seus processos e serviços, seguindo uma das 5 tendências internacionais da administração pública (CAVALCANTE; CAMõES, 2017).

Tecnologias de comunicação e informação (TICs) (ou Information and Communication Technologies - ICT) vem permitindo governos a crescerem e melhorarem, tanto na parte de gerenciamento, quanto seu serviço em si (Scholl, 2003). A adoção de TICs para auxiliar processo e fornecer serviços governamentais, além de engajar cidadãos é conhecido como Governo eletrônico, ou *Eletronic Governament (E-governament, e-Gov)* (Scholl, 2003).

Os benefícios do e-Gov descritos por Misra (2006, p. 4) permeiam vários grupos e são diversos. Para os cidadãos, promove serviços 24 horas por dia, é rápido e prático, sem a necessidade de locomoção para resolução de assuntos, e também traz transparência de gastos e processos. Para negócios, melhora a conformidade com relação as normas e leis do governo. E os benefícios para o governo são melhora na gestão de processo, criação de normas e sua divulgação, melhor performance no setor social e financeiro, além de dar a imagem de governo moderno e progressista (Misra, 2006).

Atualizando a definição de e-gov, exite o conceito de Governança Eletrônica que é a união dos cidadãos, pessoas-chaves e representantes legais para participarem, por meios eletrônicos do governo de comunidade (FERGUSON, 2002). A Governança eletrônica, vem como o resultado da contribuição do e-gov, para um contexto em que o governo constrói e implementa suas políticas com a participação da sociedade, considerando o papel das tecnologias de informação e comunicação (GUIMARãES; MEDEIROS, 2005).

O governo brasileiro, traz essa união de recursos eletrônicos e ideias da população para formular suas políticas, através da governança digital. A governança digital é a utilização de TICs pelo setor público para melhorar a prestação de serviços, disponibilizar informação, e incentivar a participação da sociedade nos processos de tomada de decisão (BRASIL, 2016).

Fazem parte das tecnologias produzidas para implementar o governo digital brasi-

leiro, o Software Público Brasileiro (SPB). Eles são serviços que seguem diretrizes e estão alinhados à filosofia de software livre, onde são disponibilizados para acompanhamento da população. O SPB é uma forma de promover o desenvolvimento colaborativo, compartilhando ideias de soluções de software em todo o Governo, e trazendo economia de recursos de desenvolvimento.

Partindo da ideia de governança digital e SPB, desenhou-se então o projeto de co-desenvolvimento entre o Laboratório Avançado de Pesquisa, Produção e Inovação em Software (LAPPIS) e a Secretaria Especial da Cultura para acompanhar e participar ativamente do plano de inovação do Governo Federal, especialmente no tópico de Economia e Sociedade Digital (BRASIL, 2018). O projeto de chatbot Tais surgiu para auxiliar e tirar dúvidas sobre a Lei de Incentivo à Cultura (BRASIL, 2019c), através da interação do proponente, ou seja, pessoa ou entidade que submete uma proposta de incentivo à Lei, com o *chatbot*.

Chatbots, ou agentes conversacionais[1], são programas de computador que têm por sua finalidade a simulação de conversas humanas através de aplicações de mensageria. Há uma demanda crescente pela utilização desta tecnologia em diversos setores da sociedade (INTELLIGENCE, 2016). Em 2018, o Facebook anunciou que somente em sua plataforma de mensagens, o Messenger, havia mais de 300 mil *bots* ativos mensalmente (FACEBOOK, 2018). Uma pesquisa realizada em 2018 pela Computer Generated Solutions - CGS, revelou que 32% dos internautas americanos acham que atendentes humanos ficam pouco tempo disponíveis e levam bastante tempo para responder sobre as dúvidas (SOLUTIONS, 2018). Sendo assim, um dos grandes benefícios dos *chatbots* é que eles trabalham 24 horas por dia e respondem instantaneamente os questionamentos dos usuários.

O projeto abarca vários usuários, em um contexto específico de conversa, logo, a Tais deve ser capaz de responder adequadamente sobre a lei, de modo a não gerar dúvidas e desafogar o atendimento da ouvidoria da Secretaria. Como o cerne do funcionamento de um *chatbot* é o diálogo com o usuário, tratar este problema é necessário para o sucesso deste serviço. Logo, este estudo tem como objetivo utilizar o caso da Tais, que esteve em funcionamento ativo, passou por todas etapas de desenvolvimento e manutenção e já atendeu em torno de 6 mil usuários, para comparar com levantamentos da academia para a construção de agente conversacional, e assim chegar em diretrizes de desenvolvimento de *chatbot*, e se tornar uma base para inciantes desta tecnologia.

O objetivo principal deste projeto é apresentar o estudo de caso da *chatbot* Tais, e com ele traçar diretrizes para a construção de um agente conversacional. Assim, para alcançar este objetivo, temos os seguintes objetivos específicos:

- Descrever o processo de construção do *chatbot* Tais;

[1] Neste trabalho, adotamos a palavra *bot*, e agente conversacional, como sinônimos de *chatbot*.

- Analisar as entregas de cada etapa de construção deste projeto;
- Discutir estas entregas em relação as propostas acadêmicas;
- Fornecer diretrizes para construção de *chatbots*.

1.1 Organização do Trabalho

As definições de um agente conversacional se encontram no capítulo 2. Essas informações são a base para o entendimento do estudo de caso. Detalha-se a estrutura de *chatbot*, sua arquitetura, e tecnologias para sua construção.

Somados a esta descrição, no capítulo 3, encontra-se o porquê de um agente conversacional ser considerado uma inovação, e como construir um produto de *chatbot*. Pontos a se considerar como a usabilidade de um *bot*, como usar fluxos para elaborar seu conteúdo antes de desenvolver seu código, e de testar seu design, além do uso de ferramentas de Business Intelligence para auxiliar a evolução do *chatbot*. Associados a isso, é abordado também o que é um software livre e como faz parte das decisões tomadas durante o desenvolvimento do projeto.

No capítulo 4 se encontra descrição do Projeto Tais, seus requisitos e premissas, descrição e análise dos passos para a construção dessa agente conversacional. E ao final, no capítulo 5, esta presente as diretrizes levantadas, com base nos marcos do projeto, e que podem ser aplicados em ciclo de desenvolvimento de *chatbot*, independente do contexto. No mesmo capítulo, encontra-se a conclusão do trabalho e sugestões de trabalhos futuros.

2 Chatbots - Referencial Teórico

Um *chatbot* é um agente conversacional que interage com usuários acerca de um domínio ou tópico utilizando linguagem natural (HUANG; ZHOU; YANG, 2007). Assim, de acordo com uma pergunta ou fala do usuário, o sistema responde um texto relacionado ao contexto. Outra definição usada no campo de negócios, é *chatbot* como agentes virtuais que se engajam em uma interação verbal com humanos (PRZEGALINSKA et al., 2019).

O exemplo mais antigo deste sistema é o ELIZA (WEIZENBAUM, 1966), programa criado entre 1964 e 1966 no Massachusetts Institute of Technology (MIT) Artificial Intelligence Laboratory, cujo objetivo era simular um psicoterapeuta, e que trouxe como resultado a possibilidade de conversa entre humano e máquina. Seu funcionamento tinha como base a identificação de palavras-chave na fala do usuário, escolher uma regra de transformação, e aplicá-la para identificar a melhor resposta.

Este tipo de sistema de interação pode ter como foco a simulação de um ser humano, ou auxiliar em processos de negócios, atendendo os consumidores de forma mais acessível e de baixo custo (PRZEGALINSKA et al., 2019). A exemplo, PARRY é um modelo computacional que simula paranoias artificialmente (COLBY; WEBER; HILF, 1971). Este é um caso famoso na literatura, pois o que caracteriza o sucesso da conversa humano-máquina, e assim, o sucesso na interpretação da paranoia, é a banca de psiquiatras não conseguir identificar se o comportamento do sistema é de uma máquina ou de um ser humano.

Com avanços tecnológicos, principalmente na área de processamento de linguagem natural (*Natural Language Process* - NLP), tornou-se possível a realização de tarefas mais complexas, tal como as conversas orientadas a objetivos. Um bom exemplo é a realização de operações bancárias diretamente com um *chatbot* (BRASIL, 2019).

2.1 Estrutura

Considerando a sustentação de diálogo pelo sistema, é preciso estruturá-lo tecnicamente nos seguintes passos:

1. Receber a entrada do usuário, através de uma interface de comunicação com o usuário;

2. Identificar o significado da entrada;

3. Classificar a melhor resposta de acordo com o significado;

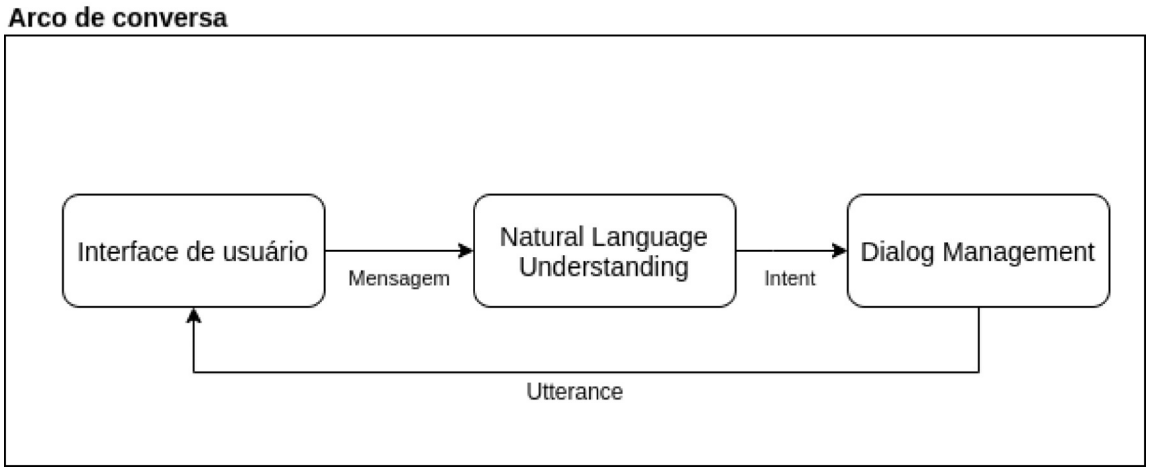
4. Enviar a resposta.

O agente conversacional precisa de uma **interface de usuário** para receber e enviar as mensagens da conversa, de uma **estrutura** para **identificar o que foi dito** e outra para **escolher a melhor resposta**.

Como dito, a **Interface de usuário** é responsável por receber e exibir as entradas e respostas (RAMESH et al., 2017). As plataforma que podem fazer o papel de interface, usualmente são mensageiros como Slack[1], Telegram[2], Rocket.chat[3], ou Facebook Messenger[4]. Estas possuem suportes para hospedagem de *chatbot*s, e no caso do Rocket.chat, o armazenamento das conversas ocorridas. Além disso, tem suporte para envio de fotos, vídeos e *emoticon*, para dar fluidez a conversa.

Para uma conversa fluir entre um usuário e um *chatbot* é necessário, inicialmente, que o *chatbot* consiga entender, a partir do uso de algoritmos de *Natural Language Understanding* (NLU), qual é a **intenção** da pergunta do usuário, denominada como ***intent***. Ademais, é esperado que após o entendimento por parte do *bot*, o usuário receba a resposta melhor qualificada para a sua pergunta, definida a partir de uma classificação do **gerenciador de diálogo**, chamado ***Dialog Management*** (GD). Chamamos essa ação de **resposta** como ***utterance*** (HARMS et al., 2018). Quando o usuário atinge o seu objetivo específico, ou seja, conseguindo executar um fluxo e obtendo sucesso, é denominado como **arco de conversa** (FALK et al., 2018), como mostra a Figura 2.1.

Figura 1 – Estrutura básica de um agente conversacional



O gerenciador de diálogo é um sistema composto por alguns elementos que trabalham desde o controle da conversa até a resposta recebida pelo usuário. É responsável por manter o contexto da conversa, fazendo com que tudo flua mais naturalmente. Além

[1] Slack: https://api.slack.com/bot-users
[2] Telegram: https://core.telegram.org/bots
[3] Rocket.chat: https://rocket.chat/bots
[4] https://developers.facebook.com/docs/workplace/integrations/custom-integrations/bots

disso, gerencia todo diálogo, criando uma base de informação, a partir da interpretação da conversa, para prever a próxima ação mais adequada do *bot*, sendo elas internas ou externas. O gerenciador também pode fazer requisições para serviços externos, melhorando a experiência de quem está utilizando do *bot*. Essa ação de diálogo corresponde à mensagem final que será encaminhada para o usuário (HARMS et al., 2018).

Por conta dos termos específicos da estrutura de um *chatbot* serem conhecidos principalmente na língua inglesa, neste trabalho traremos para alguns termos as suas traduções imediatas, facilitando a leitura do texto. No Quadro 1, sintetizamos os termos usados frequentemente neste trabalho, e que julgamos importante saber seus significados e nomenclatura em inglês.

Quadro 1 – Termos específicos de uma estrutura de *chatbot*.

Termo	**Tradução**	**Descrição**
Intent	Intenção	Intenção extraída da mensagem do usuário.
Utterance (ou utter)	Resposta	Resposta dada pelo *chatbot*, seja ela textual ou não.
Entity	Entidade	Parâmetro extraído da mensagem do usuário que auxilia o *chatbot* na classificação da resposta.
Dialog Management (GD)	Gerenciador de diálogo (GD)	Ferramenta, composta por componentes de identificação intenções e classificação de respostas do *chatbot*.
Dialog Policy (ou Policy)	Política de diálogo (ou política)	Componente responsável pela decisão da próxima ação no diálogo.

2.2 Arquitetura Geral

Chatbots são programas de computador, ou seja, não entendem a língua natural, como o português e o inglês. Dessa forma, é necessário que haja uma interpretação da mensagem enviada pelo usuário para uma linguagem que o *bot* entenda, possibilitando classificar a sua intenção em uma ação pretendida. Este trabalho de classificação de intenção é realizado pelo componente de Compreensão de Linguagem Natural, o NLU (*Natural Language Understanding*). Por exemplo, uma mensagem como "Ajude-me a encontrar um restaurante japonês no Gama" deve ser mapeada como uma ação chamada, por exemplo, de *busca_de_restaurante*. No qual terá como parâmetros de busca, definidos como entidades (ou, em inglês, ***entities***), *chinês* e *Gama*, sendo, respectivamente, o tipo do restaurante e a localidade do restaurante.

Para o NLU identificar as entidades da mensagem, é necessário ter uma base de dados pré-treinada, podendo conter exemplos de mensagens com textos, sinônimos, regex e tabelas de pesquisa. Algumas tecnologias fazem essa classificação, como por exemplo os classificadores de inteção Sklearn e o TensorFlow Embedding (GMBH, 2019). O primeiro

utiliza a biblioteca *spaCy* para carregar modelos de linguagem pré-treinados, específicos de cada idioma (não fornece suporte para todas as línguas), que separa cada palavra em um *word embedding*, significando que cada palavra é convertida em um vetor numérico denso (*dense numeric vector*) (MIKOLOV et al., 2013).

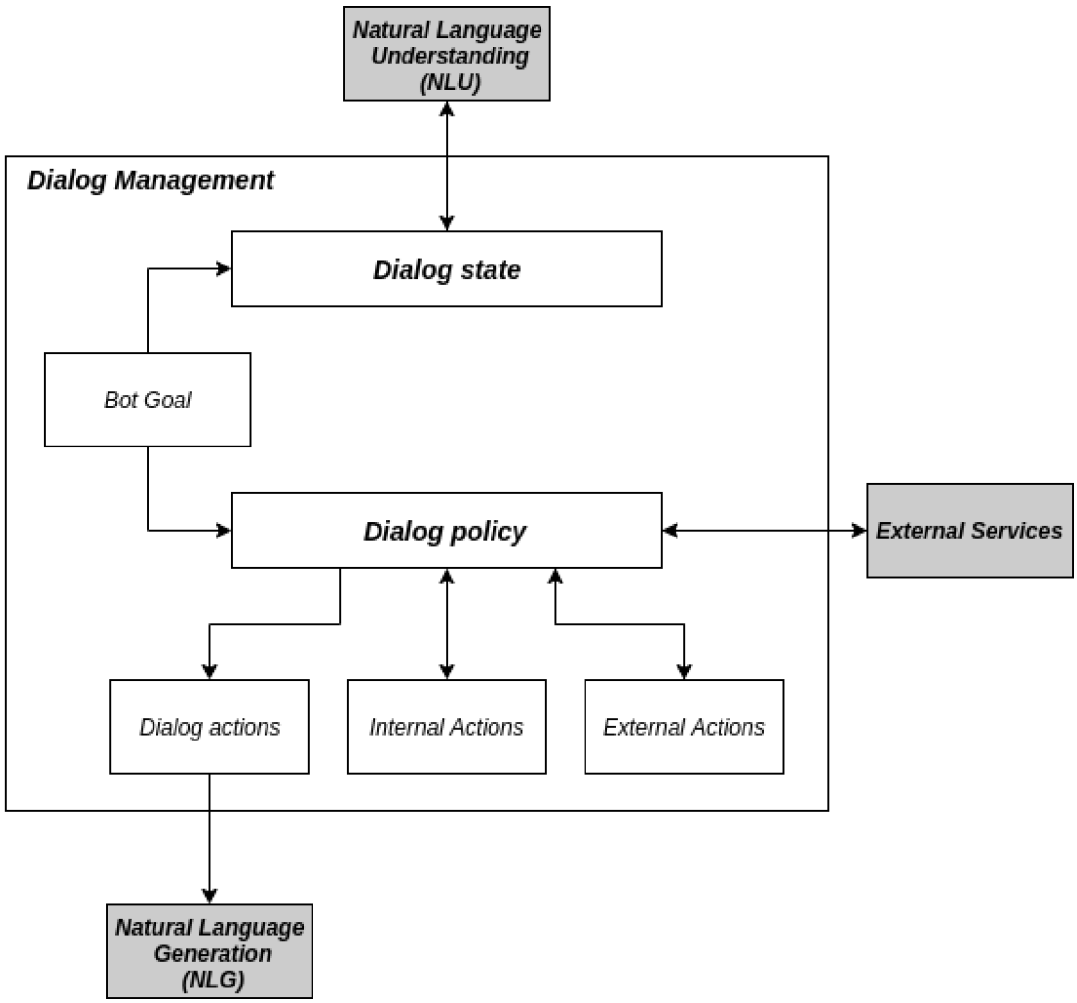
O Sklearn é bastante utilizado quando está na fase inicial do desenvolvimento de um *chatbot*, que não possui uma base de dados muito grande, tendo em vista a utilização do *Support-Vector Machine*, SVM, e requer apenas um pequeno treinamento para fazer a predição das intenções. O TensorFlow Embedding, TE, utiliza de uma outra estratégia para a classificação das intenções. Realiza uma contagem da frequência com que cada palavra dos dados de treinamento aparece em uma mensagem, armazenando cada ocorrência separadamente em um vetor. E no final fornece como entrada para o classificador de intenções (GMBH, 2019). Uma vantagem do TE é que possui suporte para mensagens com múltiplas intenções. Porém, como é um classificador que treina do zero a partir da base de treinamento, é necessário que possua mais exemplos.

A classificação da intenção e a identificação de entidades da mensagem são enviadas para o gerenciador de diálogos (GD), componente responsável por atualizar o estado do diálogo e selecionar a melhor resposta de acordo com a intenção do usuário. A utilização de um bom algoritmo de NLU torna o GD capaz de encurtar as conversas. Por exemplo, o usuário encaminha uma mensagem sobre viagem aérea que é composta pelo destino pretendido. Caso o NLU não consiga extrair essa entidade, *cidade destino*, corretamente, o gerenciador terá que fazer uma confirmação através de uma outra ação, na qual ele perguntaria sobre o local de partida do usuário, encurtando o diálogo.

O gerenciador de diálogos é uma ferramenta composta por outros componentes: ***dialog state***, ***dialog policy*** (**política de diálogo**), ***bot goal***, ***dialog actions***, ***internal actions*** e ***external actions***, como mostra a Figura 2. O primeiro componente é responsável por receber todas as informações do saída do NLU e formatá-las em uma linguagem que seja mais compreensível no nível da máquina, para então a **política de diálogo** decidir qual deve ser a próxima ação executada na conversa. Porém, antes dessa decisão, o objetivo do próprio *chatbot* pode influenciar nessa decisão. Sigamos no exemplo da passagem aérea. Suponha que o objetivo do *bot* que o usuário está utilizando seja vender uma passagem na primeira classe, logo uma próxima ação provável seja informá-lo sobre uma promoção na primeira classe. Tendo em vista isso, a informação adquirida do *dialog state* pode ser influenciada pelo seu objetivo (HARMS et al., 2018).

Após a informação ser processada pelo *dialog state* e o seu estado ser atualizado, a Política de Diálogos é acionado para decidir a próxima ação da conversa. Servindo como uma ponte de comunicação entre o contexto da conversa, os serviços externos e a resposta do agente de diálogo. Dois conceitos surgem quando abordado o tema de contexto do diálogo, *grounding* e *initiative*. O primeiro se dá pela identificação do contexto

Figura 2 – Arquitetura do gerenciador de diálogos (HARMS et al., 2018)



a partir da última mensagem do usuário, por meio de uma confirmação explícita ou implícita do próprio usuário. Ou seja, o *chatbot* poderá questionar o usuário se o fluxo de conversa o satisfaz, isso acontece caso as entidades existentes na última mensagem forem insuficientes para o prosseguimento de um diálogo. Já o *initiative* pode ser denominado como orientado ao usuário (o usuário lidera a conversa, fazendo perguntas ao *bot*), ou orientado ao sistema (o sistema orienta o diálogo, requisitando informações à pessoa). No chatbot Misto, qualquer parte pode orientar a conversa (simulação de uma conversa mais real e fluida) (HARMS et al., 2018).

Depois de contextualizar a conversa, e a partir das entidades extraídas e da intenção classificada pelo NLU, a política deve decidir entre utilizar ações internas ou externas, no primeiro caso seria utilizar informações da sua base já treinada, não precisando buscar informação em serviço de terceiros. Ou então, fazer requisições para serviços externos, caso haja necessidade de procurar por alguma informação que não esteja presente na sua base de treinamento, satisfazendo a solicitação do usuário e melhorando a sua experiência. Essas ações são retornadas para o *dialog actions* que as condensa na saída, o formato quase ideal para encaminhar ao usuário, por exemplo *"O restaurante encerra às*

{agenda_do_restaurante}". Essa informação é enviada ao *Natural Language Generation - NLG* que a formata na mensagem final que o usuário irá receber (HARMS et al., 2018).

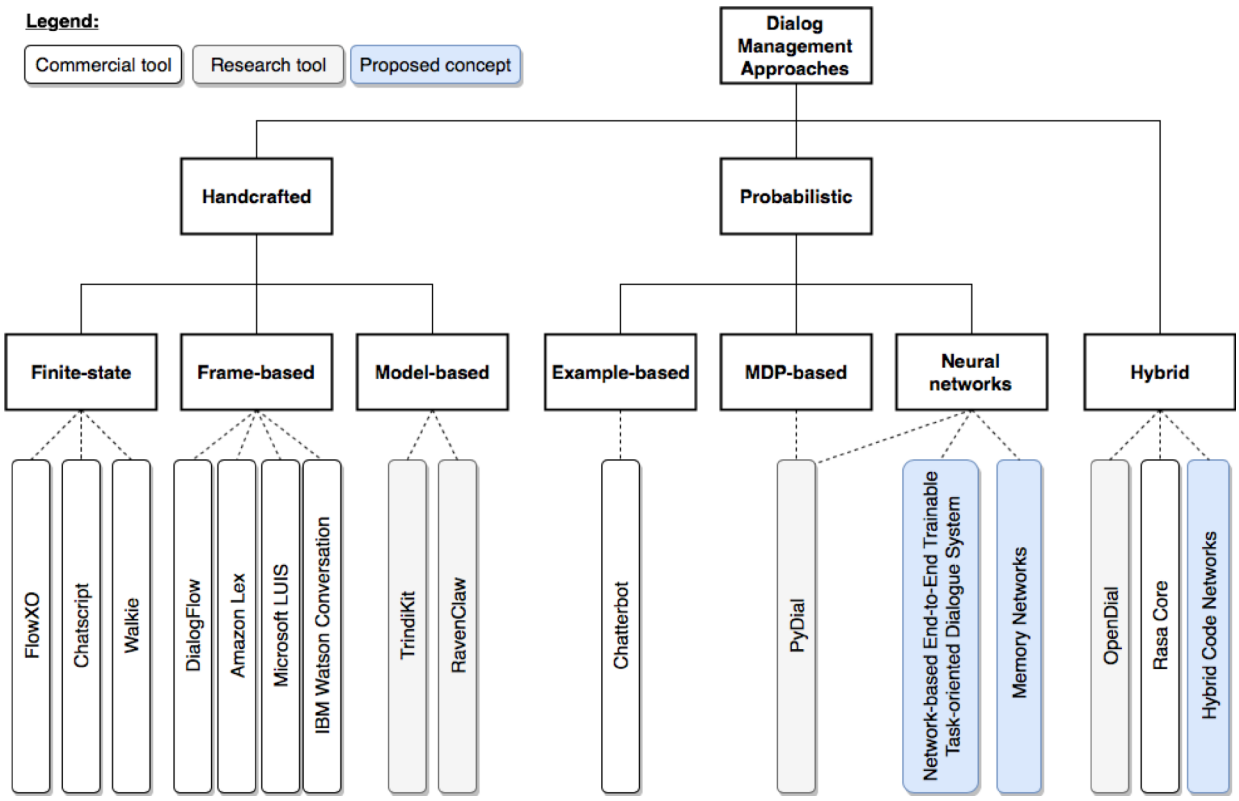
2.2.1 Gerenciador de Diálogos - Dialog Management

O gerenciador de diálogos é apenas um dos componentes de toda arquitetura de *chatbot*, mas é um dos mais importantes. Tendo em vista que é responsável por encapsular toda lógica de fala do *chatbot* e decidir qual ação deverá ser executada após o recebimento de uma mensagem do usuário. Porém, possui algumas variáveis que influenciam na decisão para selecionar uma ação em específico, a saída do NLU (contendo a intenção e as entidades), os turnos pertinentes ao diálogo em execução, o objetivo da conversa e todos os serviços de terceiros (MCTEAR; CALLEJAS; GRIOL, 2016). O GD possui uma grande responsabilidade em todo processo de conversação, sendo, basicamente, o componente que agrega todas as informações necessárias para formatar a resposta final que sera encaminhada para o usuário, impactando diretamente em sua satisfação.

O componente de extração de entidade e identificação de intenção, o NLU, comete alguns erros, tornando o entendimento do agente conversacional menos preciso do que o de um humano. Portanto, é preciso utilizar técnicas para estabelecer um nível de confiança e decidir quando solicitar confirmações sobre o que o *chatbot* entendeu, para tornar o NLU mais eficaz e diminuir a propagação de erro para o GD. Para conseguir executar essas técnicas, o gerenciador de diálogos deve acompanhar o histórico da conversa e atualizar o seu estado. Além disso, é necessário uma estratégia de diálogo que define quando e quem deve tomar a iniciativa conversacional, *Interaction strategies*, ou quando estabelecer um terreno em comum, através de validações, utilizando do tratamento de erros e estratégias de confirmação, como explicitado no *grounding* (MCTEAR; CALLEJAS; GRIOL, 2016). Essas ações são definidas através das políticas definidas na configuração do *chatbot*.

O gerenciador tem algumas abordagens arquiteturais diferentes, depende do contexto aonde o *chatbot* está inserido para então definir qual deverá ser utilizada. As abordagens definidas como Artesanais (ou do termo em inglês, *handcrafted*), surge de regras previamente estabelecidas. Geralmente utilizada para *chatbots* mais simples e que possuem iniciativas de conversa orientadas ao sistema. A Estatística é a estratégia que utiliza algoritmos de aprendizado de máquina para implementar o GD, comumente encontrada em agentes conversacionais mais complexos. E por último, o Híbrido, o caso onde o gerenciador de diálogos utiliza das duas estratégias anteriores. (MCTEAR; CALLEJAS; GRIOL, 2016). Nos subcapítulos 2.2.2.1, 2.2.2.2 e 2.2.2.3 discutiremos mais sobre cada uma das abordagens.

Figura 3 – Classificação das abordagens do *Dialog Management* (HARMS et al., 2018).



2.2.2 Abordagens para o *Dialog Management*

2.2.2.1 *Artesanal*

Uma das abordagens de desenvolvimento para o gerenciador de diálogo é a Artesanal, sendo uma solução mais simples na qual é definido a política de diálogo e um conjunto de regras, pelos desenvolvedores, que deverão orientar todo o diálogo entre o *chatbot* e o usuário. Essa abordagem geralmente é modelada como sendo tarefas altamente estruturadas, como autômatos finitos (MCTEAR; CALLEJAS; GRIOL, 2016), na qual a conversa encontra-se sempre em um estado definido, tendo possibilidades finitas de continuação. Tornando o *chatbot* orientado ao sistema, que fornece opções restritas de respostas ao usuário a cada turno, não permitindo-o fugir do fluxo definido pelo GD (HARMS et al., 2018).

Para se diferenciar em alguns aspectos do autômato finito, foi idealizada uma nova arquitetura, dentro dessa própria abordagem, para o Gerenciador de Diálogo, o *frame-based dialog*. Este novo modelo não possui um diálogo pré definido pelo sistema. Utilizam uma estrutura de *frames* que possui espaços (*slots*) de informações capturados da entrada do usuário. Podendo então, de uma mesma mensagem, coletar diversos dados relevantes daquele turno de conversa, e as fornecer para o GD em qualquer ordem (MCTEAR; CALLEJAS; GRIOL, 2016). Com essa evolução, próxima à solução de um modelo de dados, é

possível manter um modelo mais orientado ao usuário, armazenando seus objetivos, fornecendo ao desenvolvedor um leque maior de conteúdo para aprimorar o *chatbot* (HARMS et al., 2018).

A abordagem Artesanal para o GD, diálogo orientado ao sistema, possui alguns benefícios, como por exemplo o baixo custo para o desenvolvimento e também a eficiência em termos de precisão para identificar corretamente o que o usuário deseja, tendo em vista que possui um pequeno leque de opções de fluxo de conversa e não permite um diálogo orientado ao usuário. Contudo, a implementação mais rápida e menos elaborada, mesmo possuindo componentes adaptáveis, possui uma baixa versatilidade do sistema e também uma difícil adaptação do *chatbot* para compreender outras línguas naturais. Além de que é difícil e caro prever todas as entradas possíveis do usuário, requerendo tempo para fazer um refinamento do diálogo (MCTEAR; CALLEJAS; GRIOL, 2016).

2.2.2.2 Estatística

A abordagem estatística do Gerenciador de Diálogo surge em contrapartida à Artesanal para reduzir o esforço e tempo necessário para desenvolver a base de conhecimento do *chatbot* e criar uma estratégia de gerenciamento de diálogo. Com isso, algumas abordagens são possíveis, como a *Example-Based*, que após receber um valor do NLU (seja uma entidade ou uma intenção) irá combinar com um exemplo do conjunto de dados pré treinados, utilizando a resposta do mesmo conjunto. Por exemplo, temos um conjunto de treinamento sendo [usuário: "Tchau", sistema: "Tchau amigo, estarei sempre aqui."]. Logo, quando o usuário se despedir com a palavra "Tchau" ou alguma similar, o GD irá identificar em sua base de treinamento e responder a ação do sistema correspondente (HARMS et al., 2018).

Uma outra abordagem estatística para o Gerenciador de Diálogo é o Aprendizagem por Reforço, baseado no Processos de decisão de Markov (MDP), na qual determina a próxima ação do sistema otimizando a *reward function*, descrita pelo MDP, considerando o estado atual do diálogo (Noh et al., 2012). A interação ocorre por meio de estados-ações, na qual os estados determinam o número de ações que podem ser executadas. Os estados representam os possíveis contextos do diálogo (por exemplo, as entidades extraídas) e as ações são as próprias do *chatbot*, as respostas. Tendo uma probabilidade de transição que está associada a cada estado-ação. E esta mesma transição atrelada à um Sinal de Reforço, ou simplesmente, Recompensa, no qual descreve o quão bom foi o resultado da ação para aquele estado (FRAMPTON; LEMON, 2009).

Ademais, é possível abordar o GD como *corpus-based*. No qual define a próxima ação por meio de uma classificação que analisa todo histórico da conversa. Os dados extraídos da intenção do usuário, as entidades, são armazenados em uma estrutura de dados, *Dialog Register*, DR. Guardando todas as informações obtidas da conversa completa do

usuário, incluindo-as no processo de classificação (MCTEAR; CALLEJAS; GRIOL, 2016). O GD representa o diálogo como uma sequência de pares que armazenam as respostas do diálogo e as saídas do NLU, referente à intenção do usuário. O DR é alimentado a cada turno do usuário, ou seja, quando recebido uma intenção e extraído todas as suas informações, começa o processo de classificação, a partir do histórico já armazenado na estrutura de dados.

2.2.2.3 Híbrido

A abordagem híbrida do Gerenciador de Diálogo foi idealizada com o intuito de mesclar as abordagens Estatística e Artesanal. Esse modelo de GD busca reduzir a quantidade de dados necessários para a base de treinamento da abordagem estatística, que os utiliza para estimar a próxima ação do sistema. Além de tentar facilitar a inserção de conhecimentos específicos diretamente nos modelos de diálogos, sendo este fator um ponto positivo da abordagem Artesanal (LISON, 2015). A arquitetura híbrida do Gerenciador de Diálogo é utilizada em alguns *frameworks* de desenvolvimento de *chatbots*, por exemplo o *Rasa*, que utiliza hierarquia de *policies* para definir qual ação deve ser tomada. Dessa maneira, escolhendo entre a opção que seja mais viável, Artesanal, ou então uma abordagem Estatística. Além de utilizar de redes neurais (BOCKLISCH et al., 2017b).

2.3 *Tecnologias de desenvolvimento de Chatbot*

Frameworks no contexto de desenvolvimento de *software*, refere-se a um conjunto predefinido de soluções que resolve problemas comuns de desenvolvimento. Deste modo, não é necessário "recriar a roda"toda vez que vai desenvolver certa funcionalidade, fazer uma nova configuração, e muito mais. Assim, economiza-se tempo no desenvolvimento e torna-o, também, mais seguro, pois, estes, fornecem recursos de ponta para implementação de funcionalidades, como por exemplo autenticação.

Existem vários exemplos de *frameworks*, nas mais diversas áreas de programação, como por exemplo o *Django* (FOUNDATION, 2021) para desenvolvimento *back-end*, o React para desenvolvimento *front-end* (INC., 2021b), o *React Native* para aplicações móveis (INC., 2021a), dentre outros. Todos esses fornecem recursos para um desenvolvimento mais dinâmico, ágil e seguro.

O desenvolvimento de *chatbots* segue o mesmo caminho dessas outras áreas. Com arquiteturas bem definidas para os gerenciadores de diálogo, diversos *frameworks* surgiram para auxiliar no seu desenvolvimento. Os recursos disponíveis são vários e diversificados, como por exemplo, uma facilitação na integração com bibliotecas e componentes de NLU, plataformas de mensageria, definição/comportamento das políticas de diálogo, na definição das abordagens do Gerenciador de Diálogo. Temos como exemplos para o contexto

de *frameworks* para o desenvolvimento de *chatbots* o *DialogFlow* (GOOGLE, 2021), *Rasa* (BOCKLISCH et al., 2017b), Botflow (BOTFLOW, 2021), *Hubot Natural* (ROCKETCHAT, 2017), e outros.

Estes *frameworks* se diferem em vários aspectos, na linguagem de programação suportada, o *Botflow* permite por padrão o *Javascript*, já o *Rasa* define por padrão o *Python*. Nas abordagens do Gerenciador de Diálogo, como por exemplo o *Rasa*, que dar um suporte maior para a abordagem Híbrida enquanto o foco maior do *Botflow* é para a abordagem Artesanal. Assim como essas expostas, existem diversas diferenças entre cada *framework*.

O projeto Tais tinha como restrição o uso de softwares livres para seu desenvolvimento, além de necessitar de uma tecnologia capaz de interpretar diferentes perguntas, e manter o contexto nas interações com os usuários. O laboratório LAPPIS realizou uma pesquisa para saber qual tecnologia seria adequada para o uso no projeto, destacando suas vantagens e desvantagens (UNIVERSIDADE DE BRASÍLIA, 2018f).

Neste trabalho, daremos destaque e descreveremos sobre as tecnologias *Hubot Natural* e *Rasa*, pois essas foram usadas durante o projeto Tais e também, por permitirem o desenvolvimento focado na construção um Gerenciador de Diálogo que seguem os conceitos da abordagem Híbrida ou Estatística. Essa abordagem promove um agente conversacional que permite o diálogo ser orientado ao usuário.

2.3.1 *Hubot Natural*

Hubot Natural é um *framework* de código aberto (*open source*) de criação de *chatbots*, que usa arquivos de treinamento .yml para descrever os exemplos de entradas e suas respostas. Teve como base o *framework* similar chamado Hubot, e a biblioteca de processamento de linguagem natural feita em *Node.js* chamada Natural (ROCKETCHAT, 2017).

Hubot que possui a finalidade de construir *chatbot*, foi escrito em *CofeeScript* e já tem em sua composição scripts reutilizáveis para tradução, postar imagens e também integrar *Google Maps*, e a possibilidade da adição de novos scripts feita pela comunidade em seu repositório (HUBOITO, 2011).

Já a biblioteca Natural possui algoritmos para processamento de linguagem natural escritos em Node.js, e a maior parte deles baseados na língua inglesa. Alguns exemplos que ele suporta são tokenização, classificação, fonéticas, similaridade de string, entre outros (NATURALNODE, 2011). Ela também possui dois tipos de classificadores como o *BayersClassifiers* e *LogisticRegressionClassifier*, ou de regressão logística, onde o Hubot Natural utiliza esta última como padrão (ROCKETCHAT, 2017).

O design do HubotNatural permite construir o *chatbot* utilizando somente a nota-

ção YAML. As interações em YAML podem ter parâmetros específicos, processados por classes chamadas *Events* (ROCKETCHAT, 2017).

Os arquivos YAML descritos presentes em *scripts/index.js* são carregados e passados para o cerne do *chatbot*, no arquivo *scripts/bot/index.js*, onde todo o fluxo de informação e controle são programados (ROCKETCHAT, 2017). Já os exemplos de treinamento se encontram no arquivo *training_data/corpus.yml*, e tem como estrutura básica o exemplo a seguir.

Figura 4 – Exemplo de arquivo de treinamento do *framework HubotNatural* (ROCKETCHAT, 2017)

```
trust: .85
interactions:
  - name: salutation
    expect:
      - hi there
      - hello everyone
      - what's up bot
      - good morning
    answer:
      - - Hello there $user, how are you?
        - Glad to be here...
      - Hey there, nice to see you!
    event: respond
```

O significado da sintaxe de cada item do exemplo são (ROCKETCHAT, 2017):

- *trust*: o nível mínimo de certeza que o classificador deve retornar para que a ação aconteça. Os valores variam de 0 a 1 sendo o último 100% de certeza. Caso o classificador retorne um valor abaixo do estipulado, o *chatbot* retornar um erro.
- *interactions*: é um vetor de interações a serem analisadas. Toda interação do chatbot deve estar dentro de uma estrutura de *interactions*.
- *name*: nome único dado a cada *interaction*.
- *expect*: exemplos de entradas que o usuário pode fornecer, e que são usadas para o treinamento do *bot*.
- *answer*: a resposta dada pelo o *bot* caso o classificador identifique a *interaction*.
- *event*: nome da classe em *CoffeeScript* ou em *JavaScrip* dentro do arquivo *script/events* que possui uma ação a ser executada caso o classificador identifique a *interaction*.

Destrinchando um pouco mais as *Events*, estas recebem o objeto *interaction* e com base neste, analisa e executa uma ação que pode ser uma simples resposta, à uma chamada de API (ROCKETCHAT, 2017).

2.3.2 *Rasa*

Uma opção de ferramenta para construção de um *chatbot* é o *Rasa*. Esta ferramenta de código aberto (*open source*) (capítulo 3.4.1) é divida em *Rasa* NLU, responsável pela *natural language understanding*, e *Rasa* Core, como *dialog manager* (BOCKLISCH et al., 2017a).

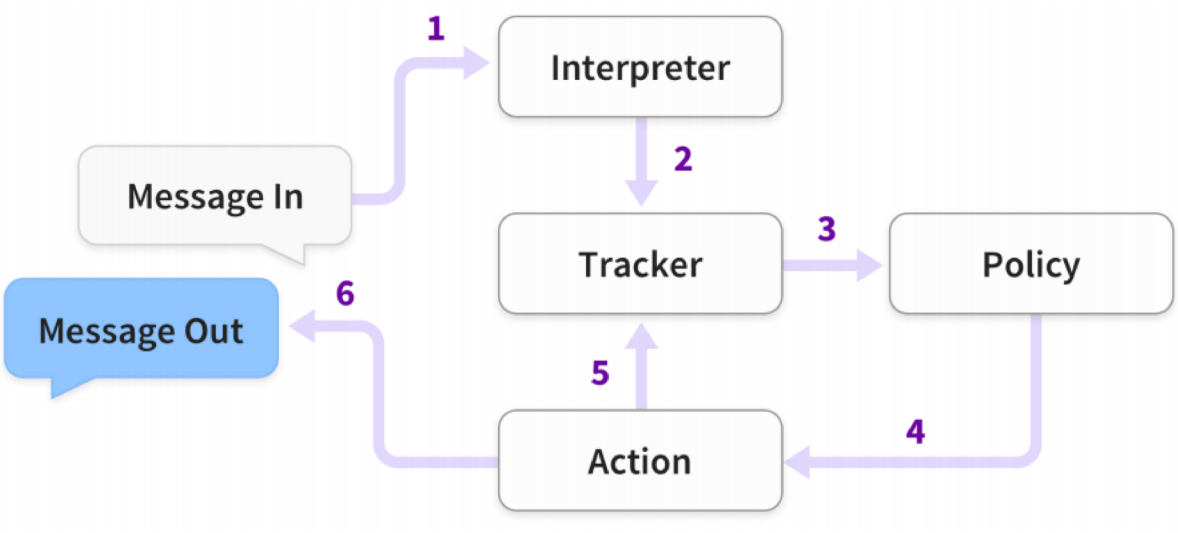
2.3.2.1 Arquitetura

A arquitetura do *Rasa*, presente na Figura 5, é uma adaptação da arquitetura genérica apresentada na Capítulo 2.2.1 e na Figura 2. Possui o componente de extração de *entities* e de identificação de *intent*, o *Rasa* NLU, denominado como *Interpreter*. Também está presente em sua arquitetura o *Rasa* Core, formado pelo conjunto de componentes denominados como *Tracker, Policy* e *Actions*, que formam o *Dialog Management*. Contudo, é visível na Figura 5 que o *Rasa* não tem em sua arquitetura a representação do componente de geração de resposta em linguagem natural, o *Natural Language Generation*, pois utiliza de uma estratégia baseada em *template*, ou seja, o próprio desenvolvedor formata a resposta previamente, chamada de *utterance*, relacionando-a com a *intent* correspondente. Então, quando selecionada a ação seguinte, o GD busca pela *utterance* já formatada.

O *Tracker* é a representação do *dialog state*, abordado na arquitetura genérica, Figura 2. É o componente responsável por atualizar e manter o estado do diálogo, e formatar as informações advindas do *Rasa* NLU em um modelo que o sistema consiga entender e processar em sua rede neural. Possibilita duas formatações diferentes, a primeira é uma representação vetorial binária, que atribui o valor 1 para a *intent* e as *entities* presentes na messagem, e com o valor 0 para todas as outras que estão em sua base de treinamento (BOCKLISCH et al., 2017c). A segunda é denominada *bag-of-words*, na qual transforma cada palavra da *intent* e das *entities* em *tokens* e faz uma contagem, armazenando em um vetor.

O *Rasa* possui um arquivo de configuração de *policies* que é a descrição de quais políticas devem ser aplicadas na seleção da melhor ação, a partir das informações armazenadas e o estado do diálogo atualizado pelo *Tracker*. Por padrão, são disponibilizadas oito *policies*: *Keras, Embedding, Mapping, Memoization, Augmented Memoization, Fallback, Two-Stage Fallback* e a *Form Policy* (RASA, 2019b). Além dessas *policies*, é permitido desenvolver a sua própria *policy* e usá-la neste arquivo de configuração. Então, quando recebido as informações do *Tracker*, todas as *policies* vão efetuar o cálculo de confiança paralelamente e o componente vai utilizar a que possuir o maior índice. Contudo, caso

Figura 5 – Fluxo da Arquitetura *Rasa* (*Rasa* NLU e *Rasa* Core) (BOCKLISCH et al., 2017a)



ocorra de duas ou mais apresentarem o mesmo valor, é escolhido então a partir da hierarquia das *policies*, definidas pela prioridade de cada uma, como demonstrado no Quadro 2, aonde quanto maior o nível, maior a prioridade.

Quadro 2 – Hierarquia das *policies* (RASA, 2019b).

Prioridade	**Policy(ies)**
5	FormPolicy
4	FallbackPolicy e TwoStageFallbackPolicy
3	MemoizationPolicy e AugmenteGDemoizationPolicy
2	MappingPolicy
1	EmbeddingPolicy e KerasPolicy

São três ações possíveis que a *policy* pode selecionar, *Utterance, Custom* e *Default actions*. A Utterance é a ação definida previamente pelo desenvolvedor em formato de *template*, suportando somente texto, por exemplo, o *chatbot* responderá o cumprimento do usuário, então a ação de resposta, "utter_cumprimentar", deve ter seu texto já especificado, podendo ser "Olá, eu sou um *chatbot*". Diferentemente da Utterance, a Custom Action pode ser definida a partir de um *script* Python, o qual pode fazer requisições externas, disparar eventos, formatação de dados e o que for necessário para a estruturação da resposta para o usuário, que pode ser em formato de texto, arquivo ou imagem (RASA, 2019a). A Default Action são ações padrões implementadas pelo próprio *Rasa* e que tem umas funções pré definidas. Como por exemplo a "action_listen", ela para de prever mais ações do sistema e espera pela próxima mensagem do usuário (RASA, 2019a), para então prosseguir com o fluxo normal de predição.

Logo após a execução da ação, o *Tracker* é atualizado e enquanto não receber uma

Default Action de *listen*, ele continua executando o ciclo a partir do passo 3, ilustrado na Figura 5.

2.3.2.2 Base de treinamento

Para treinar a classificação do *Interpreter* e as *Policies*, o *Rasa* é composto por arquivos de treinamento, onde coloca-se exemplos de *intents*, *utterances*, *actions*, *entitiese* *slots*.

O arquivo para alimentar a base do *Rasa* NLU deve ser do tipo *markdown*, ou *JavaScript Object Notation - JSON*. Abordando o formato *markdown*, temos a definição do nome da *intent* usando duas cerquilhas (##) seguido de *intent*, dois ponto e nome dado pelo desenvolvedor do *bot*, e após uma lista de entradas do usuário como itens, usando hífen(-). Por exemplo:

```
## intent:cumprimentar
- oi
- olá
- como vai?
- olá Tais
- Olá
```

No caso dos arquivos de treinamento do *Rasa* Core, as utterances são mapeadas em um arquivo tipo .yml e as ***stories*** em um arquivo *markdown*. Um exemplo de *story* é formado pelo nome desta (mesmo formato de uma *intent*), seguido de *intents* indicadas por asterisco (*) e sua *utterance* correlata, sinalizada por hífen (-).

```
## story_o_que_e_a_lei <!-- o nome é usado somente para debugar -->
* cumprimentar
    - utter_cumprimentar
* o_que_e_a_lei
    - utter_o_que_e_a_lei
```

Nota-se que o nome da *utterance* é relacionada a *intent* que ela responde. Isto não é obrigatório, é relacionado somente a convenção adotada pelo projeto, porém, é obrigatório no *Rasa* o nome iniciar com *utter_*.

3 Chatbot como produto - Referencial Teórico

3.1 Construção de um produto de chatbot

Desenvolver um *chatbot* requer um conjunto de conhecimento que abrange diversas áreas como a computação, linguística, *design*, estatística e outras mais. Deste modo é possível desenvolver um melhor trabalho, descentralizando as atividades, em busca do sucesso do projeto (presente no capítulo 2). Com uma equipe multidisciplinar é possível contornar esse problema através do direcionamento do foco de cada indivíduo para sua respectiva área (LACERDA; AGUIAR, 2019) almejando o produto final.

Um dos grandes desafios do desenvolvimento de um *chatbot* é construir falas claras, coesas e objetivas. Além disso, automatizar e melhorar o seu conhecimento, tornando-o capaz de entender toda diversidade linguística e cultural trazida na mensagem do usuário (AVERBUG, 2007). Sendo assim, é necessário um profissional com foco no estudo e análise do contexto aonde está inserido o *chatbot* (designer de conversação) para então incrementar e melhorar sua base de conhecimento.

O processo de análise do contexto e do conhecimento já existente do *chatbot* é realizado através das mensagens enviadas pelos usuários e respostas do agente conversacional. Este trabalho pode ser facilitado, pela equipe técnica de *Business Intelligence* (BI), por meio da construção de visualizações, gráficos e *dashboards* que disponibilizarão os dados estruturados e tratados (LACERDA; AGUIAR, 2019).

Figura 6 – Esquema simplificado de uma arquitetura de *chatbot* (UNIVERSIDADE DE BRASÍLIA, 2019b)

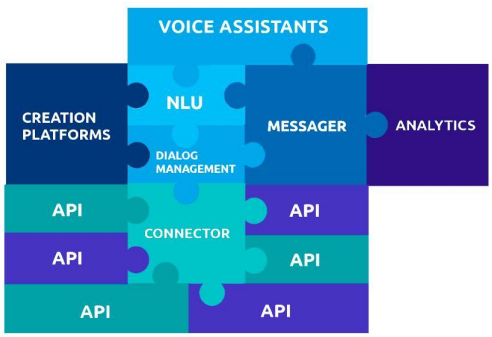
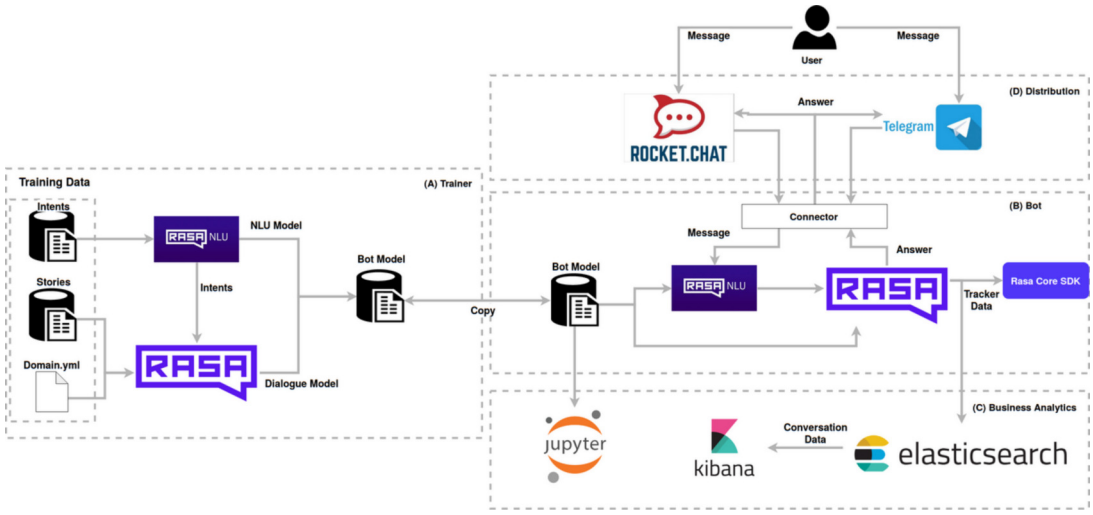


Figura 7 – Arquitetura e componentes do projeto Tais (UNIVERSIDADE DE BRASÍLIA, 2018k)



Para tornar possível e desvincular o trabalho do time de design de conversação do código do *chatbot* é necessário um trabalho de automação, desenvolvimento, manutenção e monitoramento das ferramentas e componentes do produto geral (como exemplificado na Figura 6), atividades realizadas por profissionais voltados para área de computação e/ou programação. Observa-se na Figura 7 um exemplo de aplicação de uma arquitetura que integra estes componentes.

3.1.1 Inovação e Chatbot

O Governo Federal do Brasil publicou em 2018 a Estratégia Brasileira para a Transformação Digital, o E-Digital, que abriu portas para projetos voltados para a comunicação e inovação (BRASIL, 2018). O E-Digital foi um dos eixos base para a aprimoramento da Estratégia de Governança Digital (EGD) que foi publicada em 2016 e revisada nos anos seguintes (BRASIL, 2016). O EGD define desafios, oportunidades, objetivos, iniciativas, indicadores e metas para implementar a Política de Governança Digital e a transformação digital de governo, norteando os investimentos do governo federal (BRASIL, 2016).

De acordo com o E-Digital as tecnologias de informação e comunicação da esfera pública são o vetor econômico e social da atualidade. Entende-se que o investimento em Pesquisa, Desenvolvimento e Inovação (PD&I) são fundamentais para promover empregos, aumento nos níveis de renda e atividade econômica e garantir a seus cidadãos acesso à informação, dentre outros (BRASIL, 2018). Compreendendo as diretrizes do PD&I, o EGD busca desburocratizar, modernizar, fortalecer e simplificar a relação do Estado com a sociedade, tornando o Governo federal mais acessível à população e mais eficiente em prover serviços ao cidadão utilizando tecnologias digitais (BRASIL, 2016).

Alguns dos princípios para a transformação da Governança Digital são identifica-

dos como foco nas necessidades da sociedade, compartilhamento de dados, simplicidade, priorização de serviços públicos disponibilizados em meio digital e inovação (BRASIL, 2016). Com base no conteúdo exposto no Capítulo 2 é possível através da tecnologia de *chatbot* contribuir, seguindo estes princípios, para um dos desafios da transformação da Governança Digital descrita no EGD, o de conceder amplo acesso à informação para a sociedade (BRASIL, 2016).

Chatbot é uma tecnologia que permite a implantação em diversos canais de comunicação digital com a mesma base de conhecimento, construída a partir da análise de todo o contexto em que está inserido. Isto torna mais simples e fácil a comunicação e a obtenção de informação para com a sociedade de um conteúdo específico. Além de estar disponível 24 horas por dia e conseguir atender de acordo com a demanda necessária através da ampliação dos recursos de sua infraestrutura.

3.2 Usabilidade de *chatbot*

Moore et al. (2017, p. 2) destaca que, apesar de a construção de um agente conversacional ser vista como fácil, a interação com este ainda é confusa, limitada e possui problemas para entendimento dos interlocutores. Os elementos da interface conversacional é restrita à elementos gráficos como o histórico de conversa, caixa de texto, e a interação com o usuário é conduzida principalmente por palavras (MOORE et al., 2017).

Existem algumas maneiras de determinar o sucesso de um *chatbot*, como no exemplo citado na seção 2, o *chatbot* PARRY tinha como definição de sucesso a banca não conseguir diferenciar seu comportamento com o de um ser humano com paranoia (COLBY; WEBER; HILF, 1971).

Como é observado em (NASS; STEUER; TAUBER, 1994), em uma conversa com um agente conversacional, apesar da consciência de que o usuário está tratando com um computador, este tende a ser educado e polido, identificar a máquina como amigável ou não, e até mesmo se tratando da mesma máquina, se esta emitir duas vozes diferentes, o humano a enxerga como indivíduos sociais diferentes. Este estudo mostra que pessoas aplicam regras sociais a máquinas em situação de diálogo.

Mesmo assim, a diferenciação humano-máquina, não garante o cumprimento da tarefa por parte do *chatbot*, nem mesmo a facilidade que o usuário pode ter em realizá-la.

A área de pesquisa de Usabilidade Conversacional envolve especialistas de diversas áreas de estudo de linguagem como sociólogos, comunicação e linguística, psicólogos, que oferecem modelos formais de como uma conversa natural é estruturada (MOORE et al., 2017).

Moore et al. (2017, p. 2) listam, assim, as melhores práticas para construção de

um *chatbot* levando em consideração a usabilidade conversacional, que estão disposta no Quadro 3.2.

Quadro 3 – Melhores práticas para Usabilidade Conversacional (MOORE et al., 2017).

Recepção de Usuário	O *chatbot* deve sempre falar o sobre o que ele pode ou não responder.
Comunicação Progressiva	Separar as informações em sequências para não sobrecarregar os usuários, provendo as próximas etapas em pedaços menores.
Histórico	Considerar dividir o histórico da conversa por tópicos para facilitar o usuário revisitar a conversa.
Artefatos	Sempre que possível fornecer outras mídias para facilitar a conversa, como vídeos e imagens para complementar a resposta do *chatbot.*
Feedback Multimodal	Dê *feedbacks* durante a conversa para mostrar se a pergunta foi visualizada, ou se alguma ação vai ser tomada. A exemplo, o *chatbot* mandar um "só um instante", para mostra ao usuário que o sistema está em funcionamento.
Falha "Graciosa"	O *chatbot* deve mostrar ao usuário que não entendeu a intenção desse. Projete o que o agente entendeu para que o usuário consiga observar e contornar o problema.
Personalidade	Construa uma *persona* para o agente conversacional, adequando a linguagem (formal ou informa), e prove respostas com humor e emoção para o usuário.

Daremos destaque também para uma ferramenta de construção de um diálogo de *chatbot* e uma forma de teste prévio da interação agente-humano nas Seções 3.2.2 e 3.2.1.

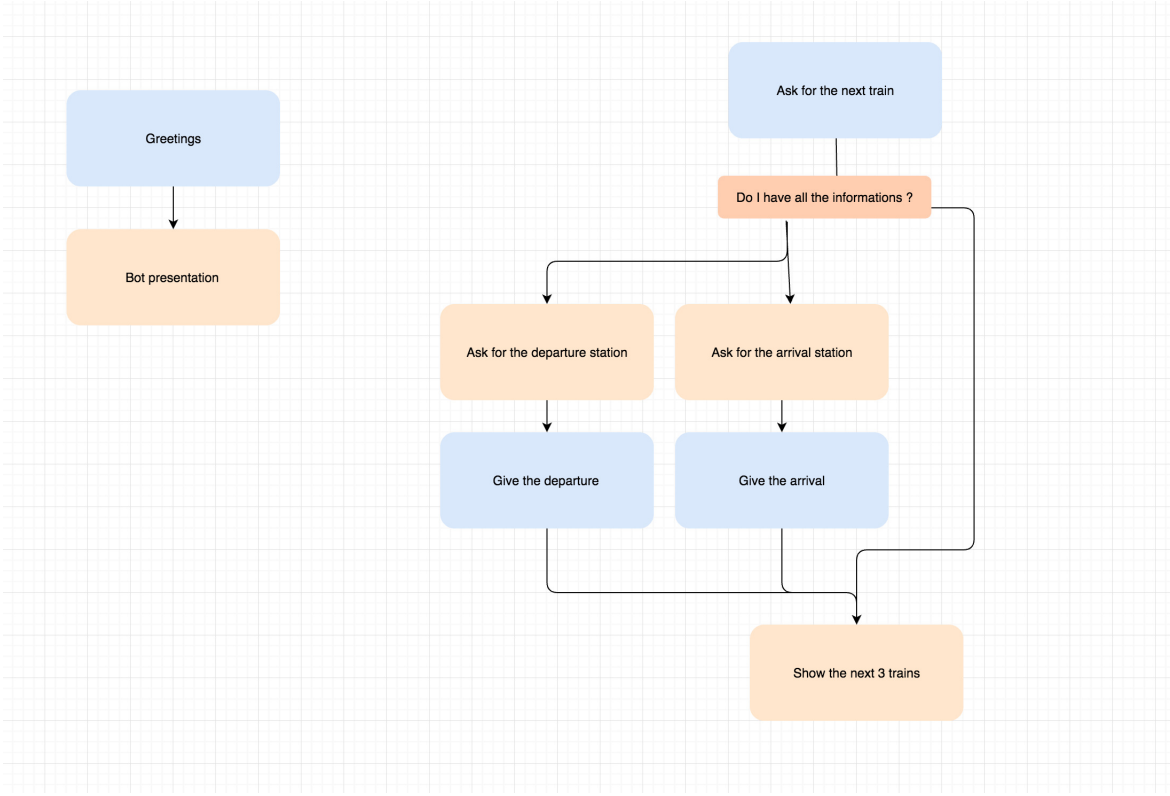
3.2.1 Fluxo de *chatbot*

Normalmente, a construção de um agente conversacional se dá no levantamento de exemplos de entradas do usuário e respostas à essas, gerando um parte <entrada, resposta> (HUANG; ZHOU; YANG, 2007). Com a tendência do usuário em considerar o *chatbot* como um agente social (NASS; STEUER; TAUBER, 1994), é possível que este pergunte assuntos diferentes do programado.

O que é chamado de fluxo, é o caminho pelo qual o usuário percorre para atingir a tarefa utilizando um *chatbot*. Comumente, este caminho é composto pelo **cumprimento**, **tarefa**, e **despedida**. Nos intervalos entre a tarefa e cumprimento e despedida, podem surgir um desvio de escopo.

Para ilustrar a modelagem de um fluxo de *chatbot*, temos a Figura 8 presente em (FOLLOW, 2017). É afirmado que para a construção de um bom fluxo, é importante manter a entre duas a três interações, além de complementar a modelagem com possíveis fluxos alternativos, e de erro (onde o *chatbot* não possui a resposta para o questionamento do usuário).

Figura 8 – Exemplo de fluxo simples (FOLLOW, 2017)



Um exemplo deste desvio de escopo de um agente conversacional é a Siri, assistente virtual presente em produtos da marca Apple, que tem como objetivo auxiliar em tarefas cotidianas, com comandos como ”E aí Siri, liga pra minha mãe no viva-voz”, ou “E aí Siri, acende as luzes da sala” (Apple Inc., 2019). Porém, existe abertura para perguntar a Siri algo como "Ei Siri, qual é o seu signo?"ou "Ei Siri, você é mulher?", o que não são tarefas cotidianas, mas o usuário tem a curiosidade de saber.

Para tratar erros como a má modelagem do fluxo, ou em casos da falta de conteúdo para um entrada específica, são criados fluxos de excessão, que podem ser nativos do próprio *framework* ou desenvolvido pela própria equipe técnica. O *Rasa* possui a *policy* chamada **Fallback** para o tratamento desses erros (Rasa Technologies GmbH, 2017). Esta *policy* leva em consideração o nível de confiança gerado pelo *Rasa* NLU na análise da *intent*, e caso a porcentagem dessa confiança for menor que a considerada pelo desenvolvedor como ideal, ele responde uma *action* padrão, com um texto como "Não entendi o que você falou, pode repetir?".

3.2.2 Testando de Design de Chatbot - Teste Mágico de Oz

Apesar da tecnologia de *chatbot* ser uma alternativa mais barata ao atendimento de clientes, o desenvolvimento de um software pode ser custoso, bem como o retrabalho caso alguma funcionalidade não seja a esperada. Como um *chatbot* não possui uma interface

de interação como aplicações Web ou aplicativos móveis, a única forma de contato com o usuário é a fala deste. Criar todas as possíveis respostas as intenções e verificar, a tempo de execução, sua precisão, é trabalhoso, logo para fazer um teste prévio a implementação e validar se a personalidade e respostas correspondem ao esperado do público alvo, o teste chamado **Mágico de Oz** é bastante adequado.

O objetivo deste teste é trazer algo para o usuário ter a experiência do sistema, possibilitando explorar conceitos de design e aspectos mais cedo no processo de construção do produto (BUXTON, 2007). Esse algo, não precisa ser um design inteiro, mas uma simulação do cerne do sistema. Aqui, a última coisa que deve-se fazer para realizar um sistema interativo é escrever código (BUXTON, 2007).

Diferentemente de protótipos de alta-fidelidade, o teste de **Mágico de Oz** consiste em um humano (normalmente chamado de mágico, ou condutor), que assume o papel da máquina e reproduz a interação humano-computador com um voluntário (FRASER; GILBERT, 1991). E para realizá-lo, precisa seguir os requisitos do Quadro 4:

Quadro 4 – Requisitos para execução do Mágico de Oz (FRASER; GILBERT, 1991).

Requisitos
Deve ser possível simular o sistema, dado a limitação humana
Deve ser possível especificar o comportamento do sistema
A simulação deve ser convincente

Um exemplo de aplicação do **Mágico de Oz** em *chatbots* é o *SearchBots*, que são *bots* que realizam buscas especificas na plataforma Slack (AVULA et al., 2018). Assim, como um teste de usabilidade, eles definiram tarefas para os voluntários cumprirem, o escopo do *bot*, e como o mágico agiria, e ao final, aplicaram um questionário sobre as impressões durante a realização do teste.

Para o começo do teste, os pesquisadores explicaram o domínio do estudo, o que são e as funcionalidades básicas de um *SearchBot*. Esses são agentes que interagem, e que podem intervir em uma conversa na plataforma Slack para prover resultados de pesquisa depois, ou não, de fazer algumas perguntas (AVULA et al., 2018).

Foram organizados 3 cenários que pares de participantes deveriam realizar. Um cenário era uma conversa para achar recomendações de restaurantes, outro era sobre achar atrações de lazer em um local, e o último sobre recomendações de livros (AVULA et al., 2018). Todos esses tinham associados duas preferências pessoais dentro desse cenário, que eram escolhidas pelos participantes dentro de uma lista fornecida. Por exemplo, para o cenário de restaurante, o participante poderia fornecer, dentro da lista dada para o cenário, o local e a preferência alimentar (AVULA et al., 2018).

Os cenários foram rodados em 3 condições:

- O *bot* não presente (*no_bot*), ou seja, em dado momento os participantes teriam que usar o ferramentas de pesquisa foram da conversa, para cumprir o cenário.

- A condição usando o (*bot_q*), onde o *bot*, para fornecer o resultado da pesquisa, intervem na conversa e pergunta sobre as duas preferências dos participantes usando um diálogo pré-determinado.

- E *bot_auto*, que intervem na conversa e fornece automaticamente a pesquisa, pois extraiu do contexto da conversa as preferências necessárias.

As possíveis respostas foram escritas previamente. Os participantes poderiam, ou não, ativar o *chatbot*, caso não fornece as duas preferências necessárias (AVULA et al., 2018).

A conversa ocorria em uma canal com os participantes e o *chatbot*. Esse *chatbot* era operado pelo "Mágico", que era responsável por intervir sempre no mesmo ponto da conversa, quando os participantes forneciam as preferências, e mostrava as respostas esperadas como o **chatbot** faria (AVULA et al., 2018).

Ao final das tarefas, os participantes preencheram um questionário sobre sua colaboração e sua interação com o *Searchbot*, por exemplo, se a intervenção do *chatbot* causou distração, se a resposta dada foi útil e salvou tempo dos participantes (AVULA et al., 2018).

3.3 Evolução Orientada a Dados

Os *chatbots* no geral, principalmente aqueles que usam redes neurais em seu treinamento, necessitam de quantidades consideráveis exemplos de entradas para melhor classificação da resposta. Comumente, criar a base de conhecimento desses agentes é feita a mão, consumindo tempo e sendo difícil adaptar a novos domínios de conteúdo (HUANG; ZHOU; YANG, 2007).

As estratégias para levar o conteúdo podem ser feitas desde consumo de informações em sites, como feito em (HUANG; ZHOU; YANG, 2007), onde ele usa fórum de discussões para recolher pares de <entrada, saída>, como titulo da discussão e a resposta melhor classificada de acordo com seu modelo, ou transcrevendo as perguntas frequentes de um domínio, caso o *chatbot* seja voltado a *Frequently Asked Questions* - FAQ.

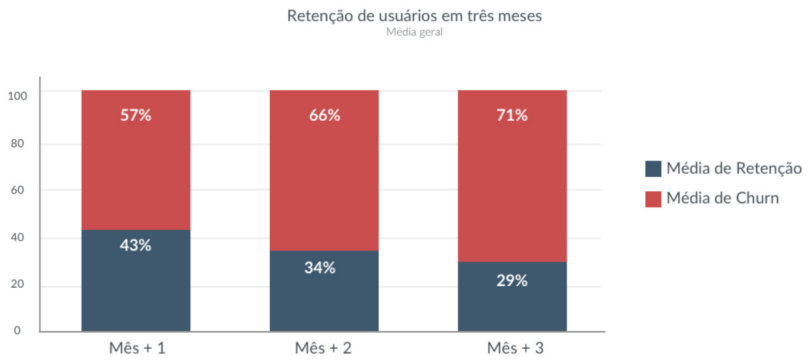
Mas quando se trata de um planejamento de alto volume de conteúdo ou da evolução de um *chatbot*, inserido em um contexto com uma grande variedade linguística (AVERBUG, 2007), é necessário abordagens e ferramentas para auxiliar na construção da base de conhecimento.

3.3.1 Business Intelligence - BI

O sistema de BI permite a fácil interpretação dos dados em grande volume, onde a análise desses dados irá permitir a identificação de novas oportunidades e novas estratégias, auxiliando em tomadas de decisão mais rapidamente. O BI fornece histórico, tempo real e previsões das operações de negócios (CHAUDHURI; DAYAL; NARASAYYA, 2011).

Imagine que o gráfico da figura 9 faz parte de um *dashboard*, conjunto de dados e gráficos, relacionado à análise de comportamento de usuário. A figura 9 representa a retenção de usuários por mês, durante um período de três meses. A partir deste dado é possível aferir se as decisões de negócio foram satisfatórias ao usuário ou não. No exemplo abordado, a retenção está diminuindo mensalmente, desta maneira é necessário fazer uma análise de contexto e produto para saber o porquê da perda significativa de usuários.

Figura 9 – Retenção de usuários em três meses (INNGAGE, 2021)



O BI é utilizado para análises de dados como identificar padrões nas vendas de uma empresa, detecção de fraudes, este possui a típica arquitetura para conseguir lidar com quantidade de consulta dos dados.

Como os dados advêm de vários lugares, é necessário a integração, refinamento e padronização e tudo isso é feito por ferramentas *Extract-Transform-Load* (ETL). Como a atividades de BI são em tempo real, temos motores (ou ***engines***) para tratar essas decisões. Estas são chamadas de *Complex Event Processing* (CEP) (CHAUDHURI; DAYAL; NARASAYYA, 2011).

As tarefas realizadas pelo BI são armazenadas em repositórios chamados ***data warehouse***, compostas normalmente por *engines* baseadas em bancos de dados relacionais (*relational database management systems* - RDBMS). Estas são complementadas por camadas intermediarias que permitem operações de BI como filtros, buscas, agregação e pivoteamento. Um tipo de camada intermediaria é o servidor *Online Analytic Processing* (OLAP) (CHAUDHURI; DAYAL; NARASAYYA, 2011).

Alguns RDMSs implementam estruturas de dados diferentes para facilitar as operações de BI, uma dessas estruturas são os ***indeces*** ou índices. Com a verticalização de índices, recuperar ou filtrar dados de vendas de uma coluna se torna uma atividade mais rápida. Em alguns casos elimina a necessidade de acessar tabelas, já que os índices dispõem de colunas com o conteúdo (CHAUDHURI; DAYAL; NARASAYYA, 2011).

E como interface destes dados tratados, planilhas, *dashboards* ou *ad hoc query*, mostrando gráficos e informações relevantes para tomadas de decisão (CHAUDHURI; DAYAL; NARASAYYA, 2011).

3.3.2 Stack Elastic

Conjuntos de ferramentas utilizado no projeto Tais, a Elastic *stack* é composta pelas ferramentas, *open sources* ElasticSearch, Kibana, Beats e Logstash. Este conjunto de componentes permite a execução completa de uma arquitetura voltada para o Business Intelligence (ELASTICSEARCH B.V., 2019).

O ElasticSearch, o coração da *stack*, age como ETL, CEP e RDMS. É responsável por fornecer recursos que o transformam em um banco de dados orientado à documentos e em uma *engine* de busca e análise RESTful distribuída em tempo real. Os dados previamente processados e tratados (pelo Logstash) fluem até ele, para então, serem indexados (ELASTICSEARCH B.V., 2019). Os índices do Elasticsearch são conjuntos de documentos *JSON*, todos eles relacionados entre si (ELASTICSEARCH B.V., 2019).

Realizada a indexação dos documentos em sua base de dados. O Elasticsearch fornece uma DSL (*Domain Specific Language*) de consulta completa, baseada em JSON (ELASTIC, 2020). Sendo possível realizar as pesquisas através de sua API disponibilizada ou por meio da ferramenta Kibana.

O Kibana é o componente da Elastic *stack* que fornece recursos para visualização e análise dos dados armazenados no ElasticSearch. Permitindo a criação de visualizações personalizadas, mapas, gráficos, grafos, aplicação de algoritmos de aprendizado de máquina, dentre outros recursos (ELASTIC, 2019). Sendo possível disponibilizar os dados analisados em *dashboards*, exportáveis via CSV ou iframe.

3.4 Software Público Brasileiro

O software Tais, objeto de estudo desse trabalho, produzido no projeto de codesenvolvimento como o laboratório LAPPIS, foi realizado em parceria da Secretaria Especial da Cultura do Governo Federal. Em meio ao movimento de governança digital do governo brasileiro, a Tais foi construída como um SPB, que leva em sua base, os conceitos de software livre.

O Software Público Brasileiro - SPB, é um software livre que atende às necessidades de modernização da administração pública de qualquer dos Poderes da União, dos Estados, do Distrito Federal e dos Municípios e é compartilhado sem ônus no Portal do Software Público Brasileiro, resultando na economia de recursos públicos e constituindo um recurso benéfico para a administração pública e para a sociedade (BRASIL, 2016).

Este começou com o Decreto de 29 de Outubro de 2003, onde institui em Comitês Técnicos para a implementação de softwares livres no governo brasileiro (BRASIL, 2019). Este comitê levantou diretrizes para a implementação que tem como exemplo, priorizar essas soluções, programas e serviços baseados em software livre que promovam a otimização de recursos e investimentos em tecnologia da informação (BRASIL, 2003).

Como resultado, a PORTARIA Nº 46, a definição de um SPB, como também suas características, como aquelas que se alinham à filosofia do software livre, a exemplo a definição da licença padrão do SPB como GNU GPL (Licença Pública Geral), que garante o cumprimento de todas as 4 liberdades essenciais de um software livre. Da mesma forma, determina a disposição dos códigos fontes desses software no Portal do Software Público (BRASIL, 2016).

Aplicação do SPB tem como benefícios a economia de recursos públicos, independência de fornecedores, segurança e compartilhamento de conhecimento (BRASIL, 2019a). O presente estudo tem como caso um software que se encaixa como SPB, pois é produzido para uso da Secretaria Especial da Cultura, e necessita que seus componentes sejam software livre também.

3.4.1 FLOSS

O movimento de software livre começou em 1983 com Richard Stallman através do Projeto GNU (RED HAT, 2020). Entende-se por software livre, aqueles que permitem o usuário ter as liberdades **para executar o programa (0)**, **estudar e mudar o código-fonte do programa (1)**, **redistribuir cópias exatas (2)** e **distribuir versões modificadas (3)** (GNU PROJECT, 2009), e como pré-requisito para que isso aconteça, seu código fonte deve estar disponível para todos. Essas liberdades garantem o senso de comunidade aos usuários, e dá o controle à este sobre o software, diferentemente de software proprietários, onde somente o seu desenvolvedor/proprietário controla seu comportamento, geralmente seguindo seus interesses (STALLMAN, 2015).

Em inglês, o termo software livre (*free software*) causa confusão, pois o fato do software ser livre não quer dizer que não pode ser comercializado. A palavra *free* atrelado ao software livre vem de liberdade, ou seja, a prática, se o software cumpre todas as liberdades, o usuário pode copiar e distribuir de forma comercial (GNU PROJECT, 2012).

Outro ponto a se diferenciar é o código aberto e software livre. Os requisitos para

o código ser código aberto (*open-source*) vem da *Open Source Initiative* (OSI) (INICIATIVE, 2018), e são similares aos do software livre, porém são mais amplos e flexíveis. Nem todo código aberto segue as liberdades do software livre, o seu código podem ter licenças permissivas, onde seus executáveis podem conter dependências não livres, ou restritivas, não permitindo sus redistribuição, ou modificação, por exemplo (STALLMAN, 2012). Isso então não segue a filosofia do software livre em dar o controle do software ao seu usuário.

Logo, para se referir a ambos os tipos de software, sem levar em consideração suas filosofias e evitar a preferência entre os dois, utiliza-se o termo neutro FLOSS - Free/Libre and Open Source Software, onde a palavra "Libre" é usada para mitigar a confusão sobre software livre ser gratuito, e afirmar a percepção de "free" como liberdade (STALLMAN, 2016).

4 Projeto TAIS

4.1 Informações Gerais

O desenvolvimento da *chatbot* Tais surgiu dentro do projeto de pesquisa **Ecossistemas de Software Livre**, advindo de uma parceria de co-desenvolvimento entre a Secretaria Especial da Cultura, parte do Mistério da Cidadania, e o Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS.

Este projeto concebido em 2017, com duração de dois anos, teve como um dos objetivos gerais realizar pesquisas aplicadas no contexto de software do antigo Ministério da Cultural, atual Secretaria Especial da Cultura, para investigar uma estrutura computacional para análise de dados de sistemas de software culturais, de modo a apoiar decisões técnicas, gerenciais e de negócio da Secretaria (AGUIAR; MENDES; NERI, 2017).

Voltado à produção de solução em software livre, um dos objetivos específicos dessa parceria é o *Processamento de linguagem natural dos dados extraídos dos diferentes sistemas de software culturais* (AGUIAR; MENDES; NERI, 2017), onde a Tais se encaixa como solução.

Este agente conversacional é voltado ao esclarecimento de dúvidas relacionadas a Lei de Incentivo à Cultura (Lei 8.313/91). Esta lei instituiu o Programa Nacional de Apoio à Cultura - Pronac, com o objetivo de ampliar o acesso e produção cultural (BRASIL, 2019c).

Para um produtor cultural conseguir o incentivo a proposta, o **proponente** (pessoa/entidade responsável pelo projeto), insere esta no Sistema de Apoio às Leis de Incentivo à Cultura - Salic, para assim, o Ministério da Cidadania fazer análise e admissão. Caso passe pela aprovação, o proponente encontra pessoas ou empresas para patrocinar sua ideia, e se captado 10% do valor definido no projeto, este é encaminhado para análise técnica. Aprovado e homologado, caso o proponente atinja 20% do valor total, este pode executar o projeto. E, ao final, todo o projeto é avaliado em relação a resultados alcançados, e prestação de conta, para que assim, os patrocinadores recebam dedução de impostos por ajudarem na realização deste projeto (BRASIL, 2019b).

Uma diagrama de fluxo do funcionamento da submissão de proposta encontra-se na Figura 10.

Neste contexto, a Tais surge como uma solução de software livre desenvolvido pelo LAPPIS, usando a ferramenta *Rasa* para atender dúvidas sobre a submissão de propostas culturais, o que é a Lei de Incentivo à Cultura, como se tornar um proponente

Figura 10 – Funcionamento geral da lei de incentivo à cultura (BRASIL, 2019b)

ou incentivador, e também alguns questionamentos sobre o processo, seus prazos e análises.

O time desta parceria foi composto por alunos, professores e profissionais formados com experiencia em construção de produtos. Informações sobre quantidade de membros, e resultados do desenvolvimento desse projeto se encontra no Quadro 5.

As práticas adotadas no projeto, advindas do *eXtreme Programming* (XP), metodologia que preza a redução de custos na mudança do software, dividindo as atividades de planejamento, analise e design em partes menores, vale destaque para as (Beck, 1999):

- Pequenas *Releases*: antes mesmo do sistema inteiro estar pronto, parte dele é posto em produção, e lançado novas versões a medida da evolução deste;
- *Pair Programming*: todo o código é escrito por uma dupla compartilhando o mesmo equipamento;
- Integração contínua: toda modificação ou evolução é integrada em poucas horas ao código anterior, sem quebra de testes e funcionalidades.
- *Planning Game*: o cliente decide escopo e tempo para cada entrega baseada em estimativas feitas pelos desenvolvedores.

Cada etapa definida no plano de trabalho do projeto é correspondente a uma iteração do XP. A ideia é que nas iterações ocorram a definição de tarefas pelo cliente, assim como o planejamento da equipe a executar, o tempo de cada tarefa e implementação dessas por parte da equipe de projeto (Beck, 1999).

Atrelado as estas prática, temos a aplicação de *sprints*, proveniente do Scrum, que é a definição de períodos *time-box* para realizar o planejamento em nível técnico, a

implementação e o *review*, ou seja, o levantamento do que foi feito dentro desta *sprint* (SUTHERLAND; SCHWABER, 2017).

Quadro 5 – Informações gerais sobre o projeto de co-desenvolvimento da *chatbot* Tais (LACERDA; AGUIAR, 2019)

Quantidade de membros que compuseram o time	21 membros
Quantidade de *releases*	22 *releases*
Quantidade de intenções	104 intenções
Quantidade de respostas	118 respostas (*utters*)
Quantidade de perguntas no FAQ inicial	35 questões

4.2 Metodologia do trabalho

O presente trabalho tem como metodologia o estudo de caso, método de pesquisa que comumente utiliza dados quantitativos coletados a partir de eventos, com o objetivo de explicar, explorar ou descrever fenômenos atuais inseridos em seu próprio contexto (BRANSKI et al., 2010).

Esta pesquisa tem caráter exploratório, ou seja, tem como finalidade desenvolver, esclarecer conceitos e ideias, proporcionando uma visão geral acerca de um fato (GIL, 2008). Como o objetivo é definir diretrizes de construção de agentes conversacionais com base no caso da Tais, o protocolo a ser seguido é a descrição e análise das etapas de desenvolvimento desse *chatbot* em termos dos seguintes tópicos:

- Levantamento de inicial de conteúdo
- Fluxo de diálogo
- Uso de *Business Intelligence*
- Ferramentas de inserção de conteúdo

E para apoiar a realização desta etapa, a preparação e coleta de dados dará nas fontes de informação do projeto, descritos no quadro 6.

Após a coleta e descrição dos dados, o trabalho tem como proposta responder essas duas questões de pesquisa:

1. Quais são as diretrizes recomendadas para desenvolvimento de *chatbot*?
2. Quais elementos podem ser compartilhado entre o desenvolvimento de um *chatbot* e de um outro produto de *software*?

Quadro 6 – Fontes de informação da construção da *chatbots* Tais

Fonte de informação	Descrição
Desenvolvimento do *chatbot*	*Pull requests* e *issues* levantadas no repositório
Relatórios de entrega	Relatórios produzidos para entregas à Secretaria Especial da Cultura presentes no repositório <https://github.com/lappis-unb/EcossistemasSWLivre>
Ferramenta de BI	*Dashboards* e visualizações de dados produzidos pelo projeto
Ferramentas de inserção de conteúdo	Ferramenta desenvolvida pelo projeto, disponível no repositório <https://github.com/lappis-unb/BotFlow>
Questionário e Fluxos	Fornecidos pelo Laboratório de Pesquisa, Produção e Inovação em Software (LAPPIS)

4.3 Fase 1 - Levantamento de Conteúdo

A primeira versão da Tais, lançada em abril de 2018, foi desenvolvida com a tecnologia *HubotNatural* (UNIVERSIDADE DE BRASÍLIA, 2018i; UNIVERSIDADE DE BRASÍLIA, 2018j), seis meses após o início do projeto.

O conteúdo inserido tem como fonte as perguntas frequentes disponibilizadas no portal da Lei de Incentivo a Cultura (BRASIL, 2019c). Com auxílio dos especialistas de negócio, as perguntas recorrentes respondidas pelo setor de ouvidoria, também compuseram o conteúdo de treinamento, totalizando 117 interações separadas em 5 arquivos de treinamento.

Leva-se em consideração que essa primeira construção do *chatbot*, mesmo com o levantamento de conteúdo com base nas perguntas frequentes, foi feita por alunos de Engenharia de Software, utilizando somente os conhecimentos de melhores prática de Engenharia de Software. Não houveram melhorias linguísticas e de interação com profissionais desta área.

Após esta inserção de conteúdo, realizou-se teste para a homologação do agente conversacional (UNIVERSIDADE DE BRASÍLIA, 2018j). Foram convidados voluntários conhecedores do contexto para utilizar a Tais, usando como base objetivos a serem cumpridos. Cada voluntário realizaria 2 dos 17 objetivos, que foram separados em 4 grupos diferentes (UNIVERSIDADE DE BRASÍLIA, 2018j).

O LAPPIS disponibilizou a análise das interações voluntário-chatbot desta homologação. Em todas estas, a Tais (à época nomeada Rouana), realizou interações erradas, classificando a resposta apropriada erroneamente, não identificando o contexto corretamente (UNIVERSIDADE DE BRASÍLIA, 2018j).

Destes erros, 4 deles foram relacionados a conteúdo não inserido, porém, ao invés de classificar a melhor resposta como o *out of scope*, a *chatbot* Tais interpretou como outra intenção, como no exemplo em que o voluntário perguntou *Qual é o limite de arquivo para anexar no SALIC?* e a Tais interpretou como *Qual é o novo fluxo do Salic?* e respondeu de acordo com essa classificação. Houve um caso de desistência do diálogo com a Tais (UNIVERSIDADE DE BRASÍLIA, 2018j).

Neste teste de homologação, um questionário foi passado em relação à experiência da interação. O questionário presente no anexo B.1, mostrou que aproximadamente 36% dos 11 voluntários acharam a interação com a Tais **ruim** ou **muito ruim**, como mostra a Figura 11. Também 6 voluntários classificaram a conversa inteira com a Tais como **insatisfatória** ou **muito insatisfatória**, ilustrado na Figura 12.

Figura 11 – Gráfico de classificação da interação com Tais - versão Rouana (UNIVERSIDADE DE BRASÍLIA, 2018a)

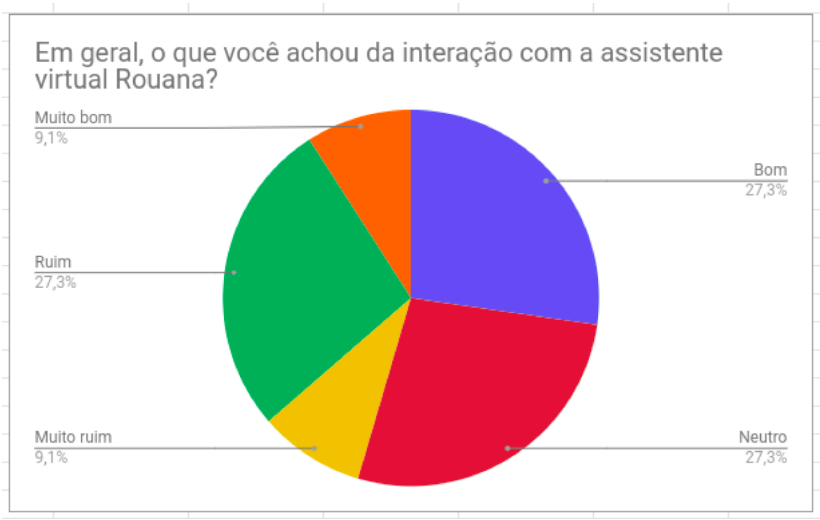


Figura 12 – Gráfico de classificação da conversa com Tais - versão Rouana (UNIVERSIDADE DE BRASÍLIA, 2018a)

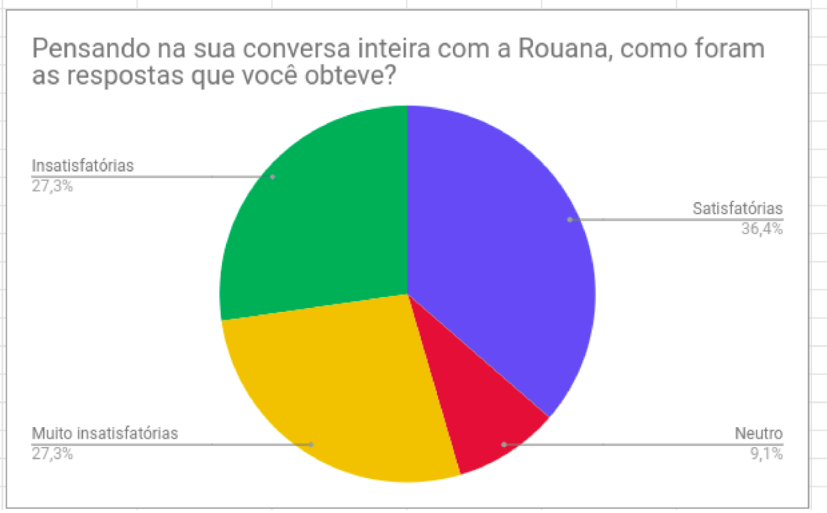


Tabela 1 – Número total de contribuições (*commits*) nos repositórios (Novembro/2017 - Abril/2018) (ROCKETCHAT, 2018; GMBH, 2018b)

	Qt. total de contribuições (*commits*) no código (Nov/2017 - Abr/2018)
Hubot Natual	31
Rasa	721

Com as análises feitas sobre a homologação, uma pesquisa de tecnologias para construção de *chatbots* foi realizada, e houve uma decisão interna de mudança da tecnologia da Tais de *HubotNatural* para *Rasa* (UNIVERSIDADE DE BRASÍLIA, 2018f). Este estudo mostra que o *Rasa* é a melhor opção de software livre para o desenvolvimento, levando em conta que este também possui processamento de linguagem natural atrelado.

As falhas apontadas neste primeiro teste podem ser rastreadas como limitações a tecnologia utilizada. Os algoritmos de classificação do *HubotNatural* são baseados na língua inglesa, o que dificulta a identificação de acentuação. Outra desvantagem desta tecnologia é a não capacidade manter o contexto da conversa e classificar corretamente as interações. Observa-se que as duas opções de classificadores não eram suficiente para o contexto em que as interações possuem palavras iguais, como no exemplo citado anteriormente. Em comparação, a tecnologia *Rasa*, escolhida para substituição, já possuía seu processamento de linguagem natural com suporte para a língua portuguesa, e fornecia dois tipos de *pipeline* que ajudam a identificar palavras similares nos vetores, tanto com o uma base pré-treinada, quanto dentro dos arquivos de treinamento fornecida (GMBH, 2018a; GMBH, 2018c). O *Rasa* em seu classificador fornecia *policies*, conjuntos de algoritmos de classificação, configuráveis, que o aumenta sua precisão (GMBH, 2018d).

Desde o começo do projeto a equipe de desenvolvimento participou ativamente da comunidade do Hubot, com aproximadamente 26 das 31 contribuições em seu código fonte (ROCKETCHAT, 2018) durante o período de novembro de 2017 e abril de 2018. Porém, comparando ao *Rasa*, no mesmo período, a nova tecnologia mostrou ter uma comunidade mais ativa com 721 contribuições, como mostra a Tabela 1.

Antes da implementação do novo protótipo, o laboratório produziu um estudo de melhores práticas de *chatbot* (UNIVERSIDADE DE BRASÍLIA, 2018d). O estudo pontua a definição de personalidade, aplicação do fluxo de *chatbots* como ferramenta de construção da interação com o usuário, e tratamento de conversas como, fazer o *bot* promover o inicio do diálogo, engajar o usuário, o direcionando a perguntar sobre conteúdos que o *chatbot* suporta (UNIVERSIDADE DE BRASÍLIA, 2018g).

4.3.1 Lições aprendidas da Fase 1

As respostas dos voluntários nas figuras 11 e 12 indicam que para um *chatbot* ser bem sucedido, descrita no capítulo 3.2, a interação deve ser clara e concisa. Ou seja, a escolha tecnológica impacta diretamente a performance do *bot*, e logo a experiência do usuário. E para que isso aconteça, utilizar uma tecnologia que consiga identificar as intenções e classificar as melhores respostas com um nível elevado de confiança é uma das ações a se tomar.

Além disso, os números apresentados na Tabela 1 e o estudo de tecnologias para *chatbot* (UNIVERSIDADE DE BRASÍLIA, 2018f) mostrou que adotar tecnologias FLOSS, com uma comunidade ativa, facilita na manutenibilidade por permitir o estudo e melhoria do código por qualquer pessoa. E para a construção de um SPB, a independência de fornecedores e redução de custo para manutenção são essenciais, como uma comunidade ativa que também traz essa vantagem.

O Quadro a seguir reúne as lições aprendidas da Fase 1, facilitando a comparação com as fases seguintes.

Quadro 7 – Lições aprendidas da Fase 1

Time	Um time composto somente por engenheiros de *software* se limita somente em desenvolver o código do *chatbot*, sendo necessário diversificar a equipe para alcançar o objetivo de um agente conversacional com uma boa experiência de usuário.
Ferramentas	Mudança para uma tecnologia com uma comunidade mais ativa e melhor na classificação de intenções
Processos	Desenvolver uma versão inicial do agente conversacional a partir das perguntas cadastradas no sistema de FAQ da Secretaria Especial de Cultura e aplicar questionários com especialistas do contexto para avaliar o desempenho e comportamento do *chatbot*

4.4 Fase 2 - Sistematização do processo de design de interação

Após a finalização da Fase 1 pontos importantes para prosseguimento do desenvolvimento foram levantados. A partir do estudo de melhores práticas realizado na Fase 1 (4.3.1), o time decidiu focar no design conversacional antes de começar a desenvolver o *chatbot* em sua nova tecnologia. Para isto, foi necessário a fragmentação do time em sub-áreas para desenvolvimento e entendimento da nova tecnologia, e a construção dos fluxos conversacionais para realização de testes com especialista do contexto.

O fluxo criado para a Tais, partiu do protótipo da versão 1.0, previamente homologada. A *versão 0* do fluxo parte do cumprimento do *chatbot*, seguido pela diferenciação entre o usuário inexperiente, que gostaria de saber curiosidades e informações básicas da

Lei de Incentivo à Cultura, e o usuário que necessita sanar dúvidas específicas. Dessas dúvidas específicas, ramificou-se em 4 grupos, considerado macros dos questionamentos relacionados a lei, observado na Figura 17 (presente no Anexo A). Um mapa mental desses grupos foi desenvolvido de forma a melhorar a visualização do conteúdo relacionado (Figura 16, presente no Anexo A).

A partir desta versão, foram desenvolvidos 5 outros fluxos. As versões 1, 2 e 3 do fluxo passaram pelo teste **Mágico de Oz**. Durante o teste, a interação foi analisada de acordo com 3 tópicos (**etapas concluídas**, **erros encontrados** e **becos sem saída**). Cada participante tomou um papel, ou de usuário comum, ou proponente, com perguntas adequadas a cada perfil. Dentre os participantes desse teste estão integrantes do próprio projeto, e também pessoas não ligadas ao projeto, mas que tinham familiaridade com a lei, como alguns possíveis proponentes, ou pessoas que já submeteram projetos.

As etapas concluídas significam etapas pelas quais o usuário deve passar para atingir o objetivo proposto pelo teste. A quantidade de erros encontrados, são as decisões tomadas pelos voluntários, e induzidas pelo comportamento do *bot* em relação a intenção, que não o desviam do caminho ótimo para atingir o objetivo. E os becos sem saída, são partes do fluxo implementado, em que o usuário chega, porém não consegue sair para completar o teste.

Para a versão 1 do fluxo, ilustrado na Figura 18 (Anexo A), 5 voluntários responderam entre 4 a 5 perguntas. Os resultados com relação a este fluxo se encontra na Tabela 2. Foram 14 interações, com média de 3,5 etapas concluídas por pergunta, e tanto a média de erros encontrados por interação, quanto a média de becos sem saídas (também por interação) foram menor que 1 por conversa, o que indica a coerência do fluxo.

Tabela 2 – Resultados do mágico de Oz para cada fluxo (UNIVERSIDADE DE BRASÍLIA, 2018a)

Fluxo	**Versão 1**	**Versão 2**	**Versão 3**
Número de Participantes	5	6	4
Quantidade de interações	14	16	11
Média de EtapasConcluídas (qtd. de etapas concluídas/qtd. de voluntários)	3,5	4	4,75
Média de Erros Encontrados (qtd. de erros encon-trados/qtd.de interações)	0,428	0,33	0,18
Média de Becos sem saídas (qtd.de becos sem saídas/qtd. de interações)	0,214	0,33	0,18

Na versão 2, Figura 19 (Anexo A), 1 pergunta foi feita a cada um dos 6 voluntários. Já na versão 3, Figura 20 (Anexo A), 5 tarefas foram dadas aos 4 participantes, onde resultados se encontram na Tabela 2.

Apesar de não se poder comparar a quantidade de etapas concluídas, por conta da mudança do fluxo e posição do conteúdo, média de erros e becos sem saídas diminuiu comparado os resultados do primeiro com os dois próximos fluxos.

É possível inferir a partir da evolução das médias descritas na Tabela 2 que com o aprimoramento dos fluxos a média de **erros encontrados** e **becos sem saídas** reduziu em aproximados, respectivamente, 58% e 16%. Mostrando a assertividade na indicação do objetivo do usuário.

A construção do fluxo 4 (Figura 21, Anexo A) não passaram por um teste. Porém, o fluxo 5 (Figura 22, Anexo A) foi validado por 3 especialistas do contexto. Todos os especialistas tiveram 5 objetivos diferentes com o *chatbot*. Durante os testes dessa versão não tiveram **becos sem saídas** e **erros encontrados**.

4.4.1 Design Conversacional

O design conversacional da Tais começou após a implementação da 5ª versão do fluxo projetado. A primeira etapa do design foi o refinamento feito com profissionais da área de ouvidoria da Secretaria. Este refinamento teve como objetivo levantar pontos de falhas do *chatbot*, sinônimos de intenções, e possíveis erros ortográficos que agregariam na base de treinamento (UNIVERSIDADE DE BRASÍLIA, 2018b). Em primeiro momento, o time de design do projeto assumiu a tarefa dessas melhorias no diálogo.

Após o refinamento, e para certificar que tanto o conteúdo, quanto a usabilidade do *chatbot* atendia o público antes mesmo de ser posta em produção, selecionou-se *beta testers*, ou seja, voluntários indicados pela Secretaria por serem produtores culturais, proponentes e possíveis proponentes, que são conhecedores da lei e do processo de submissão de propostas.

Na primeira semana de interação dos *beta testers* com o *chatbot*, 8 pessoas responderam um questionário aplicado (anexo B.2). No total, 50% dos voluntários que preencheram o questionário acharam o processo de tirar dúvidas com a Tais confortável, mais da metade achou a interação boa ou neutral, porém 75% considerou as resposta que obtiveram insatisfatória, como mostras os gráficos das figuras 13, 14 e 15 (UNIVERSIDADE DE BRASÍLIA, 2018k).

Figura 13 – Resultados do questionário do *Beta Test* (UNIVERSIDADE DE BRASÍLIA, 2018k)

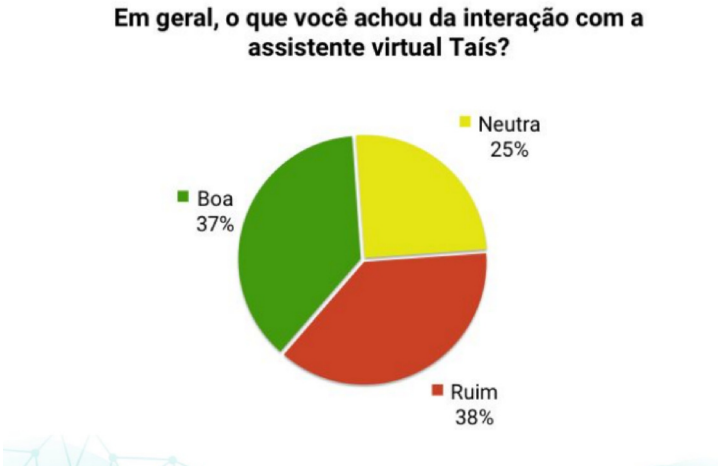


Figura 14 – Resultados do questionário do *Beta Test* (UNIVERSIDADE DE BRASÍLIA, 2018k)

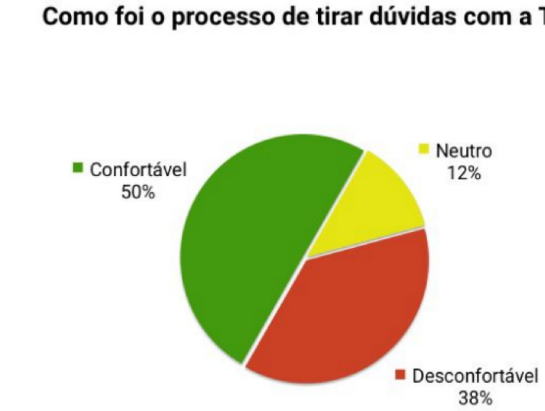


Figura 15 – Resultados do questionário do *Beta Test* (UNIVERSIDADE DE BRASÍLIA, 2018k)

Da atividade de planejamento dos *Beta Testers*, surgiram *issues*, ou seja, tarefas de desenvolvimento, atreladas (UNIVERSIDADE DE BRASÍLIA, 2018h) para a resolução de problemas relacionados ao conteúdo. Vale destaque para *issue #123 - Remover menu de contexto*. Esta *issue* está relacionada à remoção de parte do fluxo que foi implementada para orientar o usuário através de sugestões. Por conta da arquitetura da ferramenta *Rasa*, é possível navegar entre intenções e respostas, guardando o contexto da conversa. Logo, podemos criar o fluxo, ou seja, classificar a próxima respostas a partir da intenção (*intent*) e resposta (*utterance*) anterior, ou mesmo a partir de perguntas não relacionadas e manter um diálogo coeso.

Exemplificando, na versão 5 do fluxo, Figura 22 (Anexo A), ao invés de responder a pergunta "Você quer conversar sobre a criação e andamento de projeto?", o usuário pode simplesmente fazer a pergunta desejada e o *chatbot* responderia com a mesma precisão. Contudo, estes trechos de conversas que são desconexos necessitam fluidez de transição. Para isso, o LAPPIS realizou um estudo de estratégias de conversação, destacando a necessidade de engajar a conversa usando frases como *"Em que mais posso te ajudar?"*, ou reagindo a entradas não esperadas como palavrões *"Hummm... Não gostei muito dessa expressão que você usou. Que tal falar de outra forma?"*, e direcionando o usuário a tirar dúvidas da interação com *hashtags* (#MEAJUDA) (UNIVERSIDADE DE BRASÍLIA, 2018e).

Também, como evolução no conteúdo da agente conversacional, foram mapeadas 101 *issues* relacionadas a conteúdo do *chatbot* no repositório da Tais (UNIVERSIDADE DE BRASÍLIA, 2019d).

Percebendo que, mesmo o time de design estando na frente das mudanças no diálogo, e na frente da experiência de usuário do *chatbot*, um papel específico na área linguística era necessário para realizar as mudanças, e atividades mapeadas. Essas atividades, então, foram resolvidas por uma profissional da área de linguística que integrou o time de design conversacional do LAPPIS. Além de correções ortográficas, esta profissional também se atentava como o agente conversacional se expressava com o usuário, definindo sua personalidade, a melhor forma de responder, a fim de trazer nitidez e desenvoltura ao *chatbot*. Em seu cotidiano a designer conversacional mesmo não sendo da área de desenvolvimento de *software*, fazia contribuições diretamente no código fonte relacionadas à base de conhecimento da Tais. Essa habilidade, de entender e conseguir modificar diretamente o código, poderia atrapalhar nas tarefas que a designer realizaria, porém, o projeto conseguiu achar uma profissional que entendia além de linguística, mas também de práticas de programação.

Repetição de palavras e vícios de linguagem foram problemas observados que causavam má classificação das intenções, aonde a designer conversacional atuou para mitigar o problema e aprimorar a interação do *bot* com o usuário (UNIVERSIDADE DE BRASÍ-

LIA, 2018e). Além disso, o time de desenvolvimento calibrou o treinamento do aprendizado de máquina por trás da tecnologia do *Rasa*, definindo qual seria o nível de confiança aceitável (ou seja, *threshold*) da classificação das intenções.

4.4.2 Lições aprendidas da Fase 2

Todos os fluxos apresentados (presentes nas figuras 18, 19, 20, 21, 22) foram fundamentais para economizar tempo de desenvolvimento por ser rapidamente validado e aprimorado. Os testes **Mágico de Oz** foram guias para a melhoria dos fluxos, pois estes foram realizado por pessoas conhecedoras do contexto da Lei de Incentivo a Cultura.

Porém, somente o refinamento com o fluxo não foi suficiente para desenvolver uma melhor experiência de uso de um *chatbot* para o usuário, sendo necessário adaptações do fluxo para a tecnologia *Rasa*. A conservação de contexto presente nesta nova ferramenta permite que o usuário transite entre cenários não relacionados, deixando a conversa mais fluida e natural, sem prejudicar a classificação por parte do agente conversacional.

Para além da tecnologia, a melhoria do *bot* também se deu ao envolvimento profissionais de outras áreas e especialistas no contexto. Os resultados advindos dos *beta testers* deram auxilio ao enriquecimento contextual do *chatbot* comandado pela designer conversacional, que além disso modificou a personalidade, tornou a Tais mais objetiva e assertiva, transparecendo em suas interações.

Todos esses ajustes que primeiramente foram feitos por engenheiros de *software* passaram a ser responsabilidade da profissional de linguística do time do LAPPIS. Esta teve que se relacionar com um ambiente de desenvolvimento de *software*, contribuindo diretamente no código fonte da aplicação onde se localizava a base de conhecimento do *chatbot*.

O Quadro 8 permite fazer comparações mais assertivas entre as Fases de desenvolvimento da Tais.

4.5 Fase 3 - Business Intelligence aplicado a TAIS

A Tais foi posta em ambiente de produção após o aprofundamento do conhecimento em relação às tecnologias utilizadas e a consolidação do conteúdo básico através das avaliações dos *beta testers*. Isto ampliou a quantidade de usuários que utilizaram a *chatbot*. Deste modo, o desafio seria monitorar a Tais para acompanhar a sua evolução em relação às expectativas da Secretaria Especial de Cultura sobre o projeto. E também era necessário observar o comportamento da *chatbot* para tomadas de decisões técnicas.

Para monitorar o comportamento do *chatbot* o time obtinha as informações diretamente na ferramenta de mensageria, fazendo a análise mensagem por mensagem e

Quadro 8 – Lições aprendidas da Fase 2

Time	Reorganizar o time em sub-áreas (desenvolvimento e design conversacional) para aprofundar em estudos específicos sobre as tecnologias utilizadas e usabilidade. Integrado ao time uma especialista em linguística para refinar a experiência do usuário.
Ferramentas	Criação dos fluxos conversacionais anteriormente à implementação. Estudo aprofundado sobre a tecnologia *Rasa* e a calibragem da ferramenta de *machine learning* melhora a classificação das intenções através dos ajustes realizados nas *policies*, níveis de confiança aceitáveis e na melhor exemplificação de cada intenção.
Processos	Validação dos fluxos com o teste Mágico de Oz antes da implementação para uma atuação mais rápida da designer conversacional. Participação ativa dos *beta testers* para identificação de erros não previstos e melhorias em relação ao conteúdo.

transcrevendo-as manualmente em planilhas. Ao escalar o projeto, este tipo de abordagem se torna insustentável. Logo, necessitou-se automatizar o tratamento e monitoramento dos dados. Decidindo-se pela Elastic Stack, composta pelo ElasticSearch, motor de busca, e o Kibana, para criação de visualização de dados e métricas reunidos em *dashboards*.

Alinhando a necessidade dos parceiros, levantou-se os 4 principais focos para as informações dos dashboards presentes no Quadro 9 (UNIVERSIDADE DE BRASÍLIA, 2019a).

Quadro 9 – Informações para cada tipo de *dashboards* existentes no Kibana da Tais (UNIVERSIDADE DE BRASÍLIA, 2019a)

Perfil do usuário	Quantidade de conversas por dia, quantidade de perguntas por usuário, nuvem de palavras, filtro de intervalo de tempo desejado, e ranking de perguntas mais realizadas
Tendências (trending)	Aumento na frequência das questões de uma janela de tempo para outra.
Questões não respondidas	Principais informações necessárias para compreender as questões que não foram aprendidas pela Tais.
Desenvolvimento	Principais informações para a equipe técnica. Lista das perguntas feitas pelo usuário, probabilidade das intenções e *Fallbacks*.

Mas, antes mesmo da definição, foi realizado um estudo prévio para definir a relevância das métricas mencionadas no Quadro 9 quando aplicada-as em um contexto de

desenvolvimento e evolução de *chatbot*.

4.5.1 Métricas

O estudo produzido divide as métricas em de **negócio** e **desenvolvimento** (UNIVERSIDADE DE BRASÍLIA, 2019c), definidas, respectivamente em, medir se a solução está efetivamente atendendo aos objetivos propostos do projeto, e auxiliar o time técnico na identificação, priorização e execução de tarefas que impactam o comportamento do *chatbot*.

As métricas definidas como importantes para o contexto da Tais se encontram nos quadros 10, 11 (UNIVERSIDADE DE BRASÍLIA, 2019c).

Quadro 10 – Métricas de Negócios para a *chatbot* Tais (UNIVERSIDADE DE BRASÍLIA, 2019c)

Quantidade de usuários totais	Quantidade de usuário que já interagiram com a Tais.
Interações por usuário [IU]	Média de perguntas feitas por usuário.
Horas com mais atividades	Identificar em qual horário os usuários mais interagem com o *bot*.
Perguntas mais frequentes	Analisar as perguntas que são feitas com mais frequências.
Taxa de satisfação	Medida que diz respeito à taxa de satisfação em relação ao serviço prestado pelo *bot*.
Self-service rate	Quantos usuários conseguem atingir o seu objetivo com a conversa, sem a interação externa de um humano.
Retention Rate	Quantos usuários retornam a interagir com o *chatbot*. Esta métrica pode se relacionar, também, com o intervalo de tempo entre cada sessão do usuário.

4.5.2 Dashboards

Foram criados dois *dashboards* para dividir as métricas de negócio (Quadro 10) e de desenvolvimento (Quadro 11). O primeiro é relacionado ao perfil do usuário, trazendo dados sobre interação usuário vs. *bot*. O segundo agrega as visualizações que auxiliam no desenvolvimento e manutenção do *chatbot*.

Na Figura 23 podemos observar 5 *visualizações* do *dashboard* perfil de usuários. Observa-se um gráfico de quantidade de usuários por dia, relacionado ao mês, também a descrição de média de usuários por semana e mensagens por semana que a *chatbot* atende.

Quadro 11 – Métricas de Desenvolvimento para a *chatbot* Tais (UNIVERSIDADE DE BRASÍLIA, 2019c)

Taxa de confusão	Calcular a quantidade de *fallbacks* em relação à quantidade de perguntas realizadas pelos usuários.
Frases/palavras mais frequentes	Analisar as frases/palavras que são mais realizadas.
Perguntas mais frequentes	Analisar as perguntas que são feitas com mais frequências. Neste caso, pode-se definir como a pergunta mais realizada em todo o tempo, ou então a pergunta que foi tendência em determinado intervalo de tempo.
Etapas de conversação	Calcular a quantidade média de etapas realizadas por sessão. Uma etapa é definida por uma intenção do usuário e a resposta do *bot*. As conversas que excedem significativamente ou ficam aquém da média da etapa de conversação geralmente indicam uma experiência ruim para o usuário.
Fallback* por *intent	Identificar quais são as intenções de usuários que mais geram *fallbacks*.
Fluxo de sessão	Um fluxograma que mostra o "caminho"percorrido pelos usuários em cada sessão de conversa e a porcentagem de cada "caminho". Relacionando também com a métrica anterior de *Fallback* por *intent*, a qual identifica em qual intenção o *bot*entrou no *fallback*.

A Tais, analisados os 7 primeiros meses de funcionamento, tinha 6062 usuários, e estes fizeram em média de 5,472 perguntas (UNIVERSIDADE DE BRASÍLIA, 2018a).

Entender a média de perguntas por usuário é importante para o planejamento de novos fluxos e aprimoramento dos existente para modelar as *stories* fazendo com que essas contenham uma quantidade de intenções e respostas semelhantes à essa média. Esta ficou, aproximadamente, 15% maior do que a média identificada nos testes do Mágico de Oz a partir da versão 3 (Tabela 2), mostrando um engajamento maior dos usuários.

No mesmo *dashboard* é monitorado quantas pessoas utilizaram a estratégia da intenção #MEAJUDA e quantas vezes esta foi disparada. No total, 1320 usuários utilizaram essa funcionalidade, dando um total de 4054 mensagens com o texto #MEAJUDA, como mostra a Figura 24 (Anexo C). Observa-se também a quantidade de pessoas que caíram na situação de *fallback*, e ainda a quantidade de perguntas não entendidas pelo *chatbot*. Foram 2208 usuários em situação de *fallback* e 6117 perguntas não compreendidas.

A estratégia da intenção do #MEAJUDA foi desenvolvida para mitigar o impacto dos Becos sem saída na interação do usuário com a Tais. Sendo então, utilizada por 21.77% dos usuários. É possível identificar que a média de Erros Encontrados é de 0.185, estando bem próxima à média dos testes do Mágico de Oz realizados nas versões finais dos fluxos (Tabela 2). Mostrando que o planejamento e execução do fluxo, com o auxilio da tecnologia *Rasa*, foram bem alinhados.

Além disso, existe uma visualização que mostra as ocorrências incomuns de intenções, ou seja, em um intervalo de tempo, quais foram as intenções que sofreram uma mudança significativa na popularidade medida entre um conjunto de primeiro e segundo plano. Isso pode ser visto também na Figura 24. Dando base para inserção de conteúdos que parecem ser demandas urgentes dos usuários.

Na parte dedicada ao design conversacional, vê-se uma nuvem de palavras (Figura 25), em que cada palavra é uma intenção distinta, e o tamanho das fontes e sua distância do centro da visualização, diferencia a quantidade de ocorrência destas. As mais ao centro e com maior fonte tiveram um maior número de ocorrências.

Clicando em uma das intenções, o *dashboard* seguinte (Figura 26), mostra as frases dos usuários que foram classificadas com esta intenção. Isso auxilia no incremento da base de conteúdo, pois podem fornecer tanto sinônimos de uma intenção, quanto mostrar erros de classificação e frases que causam esse erro.

Ao final, encontra-se também as frases que caíram no *fallback*, dando a possibilidade da designer conversacional enxergar se, ou o *chatbot* tem erros na classificação de sua rede neural, ou existem intenções com a base pouco trabalhada.

Baseado na nuvem de palavras, e nas visualizações de perguntas de usuário, o processo de incremento de uma nova intenção, ou evolução de uma existente, é feita tratando as próprias entradas do usuário com as boas práticas de criação de intenções, para serem testas pelos *beta testers*, e para, por fim, irem para produção.

Com os *dashboards* em produção é possível identificar que a média de contribuições por mês referentes à melhoria e criação de conteúdo dobrou, como é possível ver na Tabela 3.

Tabela 3 – Quantidades de *commits* da designer conversacional no repositório da Tais (UNIVERSIDADE DE BRASÍLIA, 2019f; UNIVERSIDADE DE BRASÍLIA, 2019e)

	Período anterior às métricas (03/08/2018 - 03/04/2019)	**Período após às métricas (03/04/2019 - 03/05/2019)**
Quantidades de *commits* da designer conversacional	234	50
Média de *commits* por mês	29,25	50

4.5.3 Lições aprendidas da Fase 3

É de suma importância fazer uma análise constante do *chatbot*, principalmente quando este está em um ambiente de produção. Acompanhar em tempo real seu comportamento com o usuário, permite evoluções mais rápidas e objetivas.

Todas as métricas abordadas no *dashboard* do perfil de usuários, servem para comprovar a eficácia das ações planejadas e executadas. Além de mostrar, numericamente, o impacto da Tais no âmbito da Lei de Incentivo a Cultura Brasileira.

Os resultados presentes nos *dashboards* foram compatíveis com o que vimos no teste Mágico de Oz, o que nos dá base para afirmar que o esforço antes mesmo do desenvolvimento do *chatbot*, forma um produto conciso e com menos experiencias ruins por parte do usuário.

Compreender as necessidades do usuário, através de nuvens de palavras, e visualizar as intenções mais acessadas, permite identificar tendências de comportamento.

Além disso, as visualizações facilitam a evoluir as bases de conhecimentos continuamente. Isto torna possível atualizar o ambiente de produção com uma frequência maior. Adicionar sinônimos e expressões comuns ao usuário nesta base de conhecimento, melhora na classificação de intenções. Direcionando e dando autonomia ao trabalho da designer conversacional.

A inserção de conteúdo realizada manualmente por pessoas não desenvolvedoras geram erros humanos que impactam na sintaxe do código, pela base de conhecimento ser atrelada ao código fonte do *chatbot*, o que atrapalha a classificação das intenções, prejudicando também a experiência do usuário e demandando tempo da equipe de desenvolvimento de *software* para identificar o erro.

Todas informações sobre as lições aprendidas da Fase 3 foram condensadas no Quadro 4.5.3 para auxiliar na comparação com as demais fases.

Quadro 12 – Lições aprendidas da Fase 3

Time	A independência entre os sub-times para trabalhar em suas respectivas áreas acelera o desenvolvimento do projeto, permitindo entregas mais robustas e rápidas.
Ferramentas	Uso de ferramentas de visualização de dados, permitindo monitoramento automatizado e melhoria de classificação das intenções já existentes, assim como criação de novas.
Processos	Conciliar o desenvolvimento com as expectativas dos parceiros e as necessidades dos usuários. Monitoramento contínuo do projeto permite melhorias mais objetivas e rápidas. Embasar as decisões de projeto e produto através de estudos de casos e trabalhos acadêmicos.

4.6 Fase 4 - Ferramenta de Inserção de Conteúdo

Um time *full-stack* de *chatbot*, ou seja, que possui capacidade e conhecimento de áreas fundamentais para a construção de um *chatbot*, normalmente é composto por uma equipe de organização, ou seja, especialistas do contexto, negócio e de tecnologia, uma equipe de *chatbot* e de *Beta Testers* (LACERDA; AGUIAR, 2019). Esta equipe de *chatbot*, além de composta por desenvolvedores, *data scientists*, conta com especialistas de usabilidade, voltados para a área linguística e design conversacional.

Como citado na seção 2.3.2, a ferramenta *Rasa*, trabalha com arquivos do tipo *markdown* e *yml* para inserir seu conteúdo de treinamento. Como estes arquivos estão atrelados ao código fonte do *chatbots*, o designer conversacional deve saber manusear ferramentas de versões usadas pelos desenvolvedores. A alteração dos arquivos de treinamento feita por não desenvolvedores pode causar erros, mesmo com a familiaridade com o *Rasa*.

Em relação às contribuições nas ferramentas de controle de versão, a exemplo, no projeto Tais, a design conversacional possui 316 contribuições (*commits*) no repositório no *GitHub* (UNIVERSIDADE DE BRASÍLIA, 2018c), se tornando a pessoa com mais contribuições dentro do repositório. Ao passo que, o segundo contribuidor desta lista possui 153 *commits*, ou seja, menos da metade das contribuições. Mesmo a métrica de quantidade de *commits* não sendo relevantes, a diferença entre contribuições de conteúdo é notória e considerável.

Além disso, a modificação ou incremento de tópicos nesses arquivos tem a complicação de se enxergar a fluidez e coerência entre respostas e intenções, pois estes se encontram em arquivos diferentes, e dependendo do volume de conteúdo, essa visualização é custosa. Por isso, a tendência é realizar o treinamento e fazer uma conversa prévia com o *chatbot*.

Porém, outro problema é o tempo gasto para treinamento de um *bot*. Este tempo varia em relação ao tamanho da base de treinamento e as políticas (*policies*) utilizadas, quanto maior a base de treinamento, maior o tempo necessário. Logo, uma ferramenta de inserção de conteúdo, que possibilite, acrescentar intenções e respostas e que também seja capaz de testar as *stories* através de uma prévia, reduziria o tempo de desenvolvimento destas.

Ademais, esta fase deu início a última etapa do projeto. Logo, com a Tais estando em um ambiente estável de produção e abordando todo o conteúdo proposto pela Secretaria Especial de Cultura, começou-se a transferência de conhecimento entre as partes. Era sabido que o time da Secretaria responsável pela manutenção e evolução da base de conhecimento do *chatbot* teria um perfil menos técnico em relação à desenvolvimento de *software*.

Tendo em vista a necessidade do desacoplamento da inserção de novos conteúdos do código fonte, para dar mais independência ao time responsável pelo design conversacional, evitar erros de sintaxe de código por pessoas com pouca expertise em desenvolvimento de *software* e facilitar a transferência de conhecimento para a Secretaria Especial de Cultura, foi desenvolvido o BotFlow.

4.6.1 Projeto BotFlow

O projeto BotFlow surgiu da necessidade de prover uma plataforma de inserção de conteúdo mais iterativa e de fácil uso tanto para a designer conversacional, quanto para os funcionários da Secretaria que assumiriam a função de desenvolver o diálogo da Tais após o término da parceria.

O BotFlow é composto pelo *front-end* feito em *React* e a API para a recuperação de dados em *Django Rest*. O projeto tinha como requisitos principais o armazenamento das *intents*, *utterances* e *stories* do *bot*, assim como a funcionalidade de incremento destes, a prévia de conversas, ou seja, visualização da montagem da *stories*.

Mostrado na Figura 27 (Anexo D), a disposição do menu segue a sequência do desenvolvimento do diálogo de *chatbot*. Primeiro, tem-se as intenções, onde é possível recuperar todas as intenções disponíveis na base, assim como criar uma nova e seus vários exemplos.

Na parte de respostas, presente na Figura 28 (Anexo D), além de se poder criar uma resposta, esta pode ser composta tanto de mensagens vindas em sequência, ou seja, em balões de fala diferentes, quanto ter sinônimos para a mesma resposta.

Em relação às *stories*, a ferramenta possibilita a montagem destas, por consequência, do fluxo, selecionando as respostas (*utterances*) e intenções (*intents*), fornecendo uma prévia de diálogo para que a designer conversacional consiga ter uma noção de como seria

a conversa com o *chatbot*, como mostrado na Figura 29.

4.6.2 Pipeline CI/CD

Desde o início do projeto da Tais utilizávamos *pipelines* de CI/CD com o foco na redução do trabalho braçal, agilizar e fazer entregas mais robustas. Além disso, com a evolução do projeto e a divisão da equipe em diversas áreas o foco do CI/CD foi também para incrementar o desacoplamento, a independência da inserção de conteúdo do código fonte e liberar a equipe de desenvolvimento de *software* para focar em criação, foram desenvolvidos os *pipelines* de integração contínua (CI) e entrega contínua (CD). Os quais realizam, respectivamente, a mesclagem entre duas versões distintas do código após a validação do código realizada através dos testes automatizados, e a implantação de uma versão final em ambientes de homologação e/ou de produção quando identificado adições no código referente a cada ambiente. Estas ações automatizadas permitem a identificação de erros precocemente e a disponibilização de versões do projeto de maneira mais rápida e com uma maior frequência (Shahin; Ali Babar; Zhu, 2017).

O GitLab CI/CD foi utilizado em ambos *pipelines*, porém com papéis e configurações diferentes. Para a realização de um ciclo de integração contínua quando modificações de código eram aceitas para as *branchs* de homologação e/ou de produção, era necessário realizar ações sequenciais:

1. Gerar a *build* dos pacotes necessários para fazer o *Rasa* iniciar (*container* separado).
2. Gerar as imagens dos *containers* do *coach* (responsável pelo treinamento do modelo) e do *bot* (responsável pela configuração e integração com os outros serviços), ou seja, a arquitetura do projeto.
3. Testar a sintaxe do código escrito em Python.
4. Validar se houve alteração na base de conhecimento e se a sintaxe do conteúdo está correta.
5. Realizar os testes das *stories*.

Os testes de *stories* eram executados linearmente para todas *stories* desejadas. Assim, eram definidos caminhos felizes para cada história, ou seja, dadas intenções espera-se respostas previamente definidas. Logo, são identificados erros caso o resultado não obedeça a pré-definição do teste.

Após a realização da integração contínua, verificado e validado as novas alterações no código, seguia-se para a entrega contínua, que seguia sequencialmente os seguintes passos:

1. Criar uma nova *tag* e atualizar as novas imagens dos *containers coach* e do *bot*, criadas na integração contínua.

2. Atualizar o ambiente de produção e/ou homologação caso haja alguma modificação nas imagens do Docker (identificada através da comparação entre as *tags*).

4.6.3 Lições aprendidas da Fase 4

Com o amadurecimento da equipe nas tecnologias, no processo e na metodologia adotada para o projeto foi possível concluir em tempo hábil a implementação de todos os componentes que formam um *chatbot*.

O desacoplamento da inserção de conteúdo do código e os *pipelines* de integração e de entrega contínua, dão total autonomia para o time de design conversacional, fazendo com que este possa ser mais diverso, não necessitando de conhecimento técnico voltado para o mundo da computação, como por exemplo saber sobre Git. Facilitando também a transição de projeto entre o LAPPIS e a Secretaria Especial de Cultura. Além disso, permite que o time de desenvolvimento de *software* foque em funcionalidades mais técnicas.

Antes mesmo do termo MLOps (*Machine Learning Operations*) ser cunhado (MODELOP, 2020), o processo já estava inerte em nossa metodologia de desenvolvimento. Podendo, este, ser visto de maneira completa nesta quarta fase através dos passos subsequentes: Disponibilizar uma versão do produto em seus respectivos ambientes, fazer o monitoramento do uso do produto, planejar melhorias, implementar as melhorias, treinar o modelo (no caso, a base de conhecimento do *chatbot*), fazer a integração contínua passando por todas as etapas de testes necessárias e, por fim, publicar uma nova versão. Todos esses passos formam um ciclo do MLOps (MODELOP, 2020), que já aplicávamos integralmente em nossa quarta fase.

Aplicando este processo e com as equipes focadas em suas respectivas áreas, as novas versões eram geradas com mais agilidade e com um nível maior de confiabilidade.

O amadurecimento da equipe, principalmente na questão de definição de papéis e atribuições necessárias para o desenvolvimento, manutenção e evolução do projeto, permitiu criar uma comunidade em volta dos produtos e das contribuições geradas pelo LAPPIS. Tal comunidade surgiu a partir de palestras, *workshops* e compartilhamento de conhecimento de forma livre em variados canais de comunicação.

O Quadro 13 reúne todas as lições aprendidas e absorvidas da fase 4 de desenvolvimento do projeto da Tais e de seus derivados.

Quadro 13 – Lições aprendidas da Fase 4

Time	O time chegou em uma composição ideal para a manutenção e evolução do projeto. Este amadureceu em processos, conhecimento e metodologia. Sendo possível visualizar isso através dos produtos desenvolvidos, na comunidade criada e na transferência de conhecimento para a Secretaria Especial de Cultura.
Ferramentas	A criação de uma plataforma de inserção de conteúdo tornou os times ainda mais independentes entre si, além de mitigar riscos referentes à sintaxe de código. Riscos tais, que foram ainda controlados através da implantação do CI/CD automatizados.
Processos	A adoção do MLOps como processo que guiava a manutenção e evolução da Tais permitiu a equipe ser mais ágil e assertiva em relação às entregas.

5 Conclusão

5.1 Lições Aprendidas

O desenvolvimento da Tais teve foco direto em alguns princípios para governança digital. O foco nas necessidades da sociedade, simplicidade, priorização de serviços públicos disponibilizados em meio digital, e inovação (BRASIL, 2016). A Lei de Incentivo a Cultura necessita de preenchimento de vários critérios, e muito destes tem linguajar específico, o que é uma barreira para a submissão e aprovação de projetos culturais. Logo, trazer uma linguagem cotidiana ajuda o cidadão e desafoga a ouvidoria da Secretaria Especial da Cultura, concordando com os benefícios levantados pelo Misra (2006), como fornecimento de serviço disponível 24 horas por dia para os cidadãos, melhor gerenciamento de processos governamentais, e melhor disseminação de regras, regulamentos e atividades do governo.

A popularização de *chatbots* (FACEBOOK, 2018) facilita o engajamento dos usuários e a disponibilização de serviços digitais, que podem estar presentes em diversas plataformas e dispositivos. Como um dos princípios da governança digital diz que o cidadão é o principal instrumento para o desenvolvimento dos SPBs (BRASIL, 2016), neste projeto o usuário fez parte da construção do *chatbot* desde o início. A primeira versão da Tais utilizando o Hubot mostrou a necessidade de usar uma tecnologia mais robusta para a classificação das intenções. Além disso leva-se em consideração o comportamento e as atividades das comunidade dessas tecnologias. Seguindo as diretrizes de um SPB, a utilização de FLOSS é importante para a otimização de recursos e investimentos em tecnologia da informação (BRASIL, 2003). Assim, a decisão da mudança do Hubot Natural para o Rasa, tem como ponto a se observar a diferença de atividade entre as suas comunidades, fora a melhor performance de classificação de intenção do Rasa.

Para enxergar a ineficiência da tecnologia, foi necessário testes iniciais com entendedores do contexto. Logo, desde o inicio do projeto, as avaliações de possíveis usuários, auxiliou na evolução do *chatbot*, assim na percepção da Secretaria, pois, mesmo com os caminhos felizes implementados, ainda existia pontos que prejudicavam a experiência do usuário, e que poderiam acarretar o fracasso do projeto.

Com a dificuldade de adaptar o contexto da Lei de Incentivo à Cultura para diálogos, por ter termos e linguagens não usuais, uma maneira de integrar o possível usuário ao desenvolvimento e obter suas percepções de maneira mais rápida, é utilizar os fluxos de conversas associadas ao teste Mágico de Oz. Com estas técnicas, foi possível construir 5 versões diferentes de interações entre o usuário e a Tais, minimizando os possíveis erros

e descontentamentos dos usuários. Porém, a construção desses fluxos foi realizada por pessoas especialistas da área de tecnologia da informação, que não compreendiam as nuances por trás da área linguística. Assim, a entrada de um especialista de linguística no projeto permitiu a melhoria da interação entre o *bot* Tais e o usuário, aplicando melhores práticas levantadas por Moore et al. (2017), como a aplicação de uma persona ao agente conversacional, fragmentar as informações a fim de não sobrecarregar o usuário, e fazer o *chatbot* ser receptivo, deixando claro o que pode ou não ser respondido.

Após as etapas de testes, com os *beta testers* utilizando o Mágico de Oz, e a reestruturação feita pela designer conversacional, a Tais estava bem estruturado a ponto de ser posto em produção. Contudo, para mantê-la atualizada, monitorar o comportamento do usuário, através do *dashboard* de métricas de desenvolvimento, é importante para obter dados que auxiliam sua evolução em tempo real. Observar intenções mais acessadas dá base para entender quais são as dúvidas presentes, monitorar quantas vezes o *chatbot* se confundiu, ajuda a melhorar a base de conhecimento, e também essas métricas fornecem sinônimos para que a *designer* conversacional enriqueça o conteúdo com a própria linguagem do usuário.

Cabe aqui analisar que a evolução deste produto se deu pela composição de um time diverso como profissionais da área de linguística e *design*, e não somente de desenvolvedores de *software*. Para sua fluidez na conversa, foi necessário a atuação de uma *designer* conversacional, dissecando possíveis formas de fornecer a informação ao usuário. Do lado do desenvolvimento de *software*, o conhecedor de *machine learning* deu base para mudança de tecnologia, e alterações em *policies* para classificação de intenções, assim como o *DevOps* otimizou a *pipeline* de entrega continua, para fornecer versões novas de maneira mais rápida e automatizada. Agora, para o monitoramento, o papel de cientista de dados ajuda a munir o time de desenvolvimento e o design conversacional com métricas de negócio e desempenho do produto.

Para que os profissionais de inserção de conteúdo não dependam dos desenvolvedores, utilizar uma ferramenta de inserção de base de conhecimento se faz necessária. A tecnologia Rasa possui a sua própria ferramenta, porém, vai de desencontro às diretrizes de construção de um SPB, pois esta era uma ferramenta não livre e paga. Logo, além de um *chatbot* o projeto abarcou o desenvolvimento de uma ferramenta *software* livre para a inserção de conteúdo, o BotFlow.

As decisões de ferramentas e abordagens utilizadas, sempre foram baseadas em estudos do que a academia e o mercado tinha de mais avançado. Estes estudos não eram bloqueio para o desenvolvimento continuo, pois a equipe se dividia de modo a desenvolverem paralelamente o estudo de ações futuras que seriam realizadas no projeto. Isso pode ser percebido no inicio do projeto, quando mesmo utilizando a tecnologia *Hubot*, já se executava estudos para saber qual seria a melhor tecnologia para o contexto, e também o

estudo de melhores práticas, de métricas para *chatbot*, e de design feito antes mesmo da mudança de tecnologia.

Todo o conhecimento gerado foi propagado pelo uso de FLOSS. Uma das bases do *software* livre, é a disponibilização do código fonte para que todos possam ver e sugerir mudanças, e para isso, é necessário que todos possam estudar e entender o código. O LAPPIS disponibilizou os materiais criados durante o projeto, documentos e códigos, e apresentou *workshops*, para auxiliar a independência do governo em manter seu SPB. Além disso, para a evolução desse produto, o FLOSS foi importante, pois o time conseguiu apoio nas comunidade das tecnologias utilizadas, em seus fóruns e canais de dúvidas, além de, também contribuir com essas comunidades, melhorando não somente o *chatbot* desenvolvido, como outros.

Dado estes marcos da construção do *chatbot* Tais, observamos estas diretrizes, presentes no Quadro 14 para o desenvolvimento de um agente conversacional aonde uma boa interação com o usuário é o foco principal:

Quadro 14 – Recomendações para construção de *chatbots*

Criação de fluxos para teste	Necessário para montar a base do *chatbot* na perspectiva do usuário antes mesmo de implementar o código.
Especialista sobre o contexto para serem *beta testers*	Pessoas conhecedoras do contexto podem auxiliar em testes prévios ao lançamento do produto, para identificar possíveis erros ou experiencias ruins com o *chatbot*.
Uso do Mágico de Oz	Possibilita a avaliação da interação *chatbot* e usuário de maneira rápida, e também determina becos sem saída e erros durante conversa.
Profissionais de Design Conversacional	Especialista em linguística e *design*, acrescentam não só no tratamento de intenções, mas também enriquecem o diálogo com aplicação de personalidade e traços de linguagem na fala do *chatbot*.
Uso de métricas	Permite a evolução em tempo real da base de conhecimento e mostra o desempenho do *chatbot*.
Uso da ferramenta de inserção de conteúdo	Separar o espaço de trabalho de *design* e de desenvolvimento de *software*, previne erros na sintaxe da linguagem do *chatbot*, e promove um meio mais amigável para inserção de conteúdo por pessoas fora da área da computação.

Desta maneira, acredita-se que as recomendações presentes no Quadro 14 possam

ser usadas como base para o desenvolvimento de um agente conversacional independente de seu contexto.

Ademais, durante o desenvolvimento deste produto, enxergamos também recursos que podem ser aplicados para além da construção de um *chatbot*. Estes recursos encontram-se no Quadro 15.

Quadro 15 – Recomendações para desenvolvimento de um produto

Formação de um time diverso	Integrar áreas de *designer*, *devops*, *machine learning*, ciência de dados e desenvolvimento de *software* ajuda na construção de um produto robusto.
Desenvolvimento orientado à estudos	Decisões que serão tomadas futuramente podem ser estudadas paralelamente ao desenvolvimento do produto. Assim, quando for preciso o estudo norteia a decisão.
Desenvolvimento orientado à testes	Conciliar o desenvolvimento do código com a criação de testes previne erros futuros.
Automação de tarefas	Uso de ferramentas para automatizar a integração e a entrega contínua do código impacta a chegada mais rápida de novas funcionalidades ao usuário final.
Interação com a comunidade FLOSS	A adesão, contribuição e criação de comunidades FLOSS auxilia na disseminação de conhecimento, evolução do projeto e redução de custos.

5.2 Conclusão

O projeto Tais foi bem sucedido, e no tempo que esteve no ar, forneceu mais de 6 mil respostas aos usuários, o que mostra o engajamento da população com as tecnologias fornecidas pelo governo.

Precursor de comunidades de *chatbot*, o LAPPIS forneceu informações ricas e relevantes, que permitiu a construção desse trabalho. Apesar disso, os estudos feitos durante o projeto, como por exemplo o teste Mágico de Oz, careceram de rigidez em sua estrutura. Utilizando o mesmo exemplo, o teste Mágico de Oz poderia ter mantido uma quantidade igual de participantes e de perguntas para todas rodadas aplicadas.

No geral, o projeto se mostra importante, pois reúne e consolida estudos e práticas diversas para o desenvolvimento de um *chatbot*, que neste caso, conseguiu atender usuários em um contexto governamental. Ele demonstra também a maturidade da equipe de projeto do laboratório universitário.

O presente trabalho faz o esforço de condensar 2 anos de dedicação dos parceiros, para indicar as estruturas bases por trás da construção de um produto de *chatbot* e de software. Esse apanhado é valoroso para os próximos projetos de software, possibilitando o uso das diretrizes levantas como guias, além de poderem visualizar os erros e o racional da equipe os solucionou e que trouxe o sucesso do projeto.

5.2.1 Trabalhos Futuros

Como primeiros trabalhos futuros sugeridos, vem a aplicação dessas diretrizes na construção de um *chatbot* e produtos de software em diferentes contextos. Assim tem-se mais insumos para definir acertos, e possíveis falhas dessa diretrizes.

Partindo para a área de *chatbots*, pode-se analisar a construção em diferentes contextos, observando a atuação da tecnologia utilizada e sua contribuição para o desempenho das respostas.

Outro questionamento que surge, é a definição e coleta de métricas importantes para o *chatbot*, definindo um guia para outros que venham a existir. Para testes de Mágico de Oz, existe a possibilidade de consolidar o que se deve observar e quais possíveis perguntas fazer aos participantes em um contexto de *chatbot*.

As oportunidades para estudos na área de *chatbots* são inúmeras, pois é um contextos cada vez mais requisitado, e é importante aplicar melhores práticas de um produto de software neste contexto também.

Referências

AGUIAR, C.; MENDES, F.; NERI, H. *Plano de Trabalho - Ministério da Cultura*. 2017. <https://github.com/lappis-unb/EcossistemasSWLivre/blob/master/plano-de-trabalho/plano-minc.pdf>. Acessado em: 07 de novembro de 2019. Citado na página 53.

Apple Inc. *Siri – Apple (BR)*. 2019. <https://www.apple.com/br/siri/>. Acessado em: 03 de novembro de 2019. Citado na página 45.

AVERBUG, M. C. Variedade linguística nas escolas brasileiras: pronomes nominativo, acusativo e dativo. *XXII Encontro Nacional da Associação Portuguesa de Linguística, Lisboa, Anais. Lisboa: APL*, p. 95–110, 2007. Citado 2 vezes nas páginas 41 e 47.

AVULA, S. et al. Searchbots: User engagement with chatbots during collaborative search. In: *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*. New York, NY, USA: ACM, 2018. (CHIIR '18), p. 52–61. ISBN 978-1-4503-4925-3. Disponível em: <http://doi.acm.org/10.1145/3176349.3176380>. Citado 2 vezes nas páginas 46 e 47.

Beck, K. Embracing change with extreme programming. *Computer*, v. 32, n. 10, p. 70–77, Oct 1999. Citado na página 54.

BOCKLISCH, T. et al. Rasa: Open source language understanding and dialogue management. *ArXiv*, abs/1712.05181, 2017. Citado 3 vezes nas páginas 13, 38 e 39.

BOCKLISCH, T. et al. Rasa: Open source language understanding and dialogue management. *CoRR*, abs/1712.05181, 2017. Disponível em: <http://arxiv.org/abs/1712.05181>. Citado 2 vezes nas páginas 35 e 36.

BOCKLISCH, T. et al. Rasa: Open source language understanding and dialogue management. *CoRR*, abs/1712.05181, 2017. Disponível em: <http://arxiv.org/abs/1712.05181>. Citado na página 38.

BOTFLOW. *Botflow - No code chatbot builder*. 2021. <https://botflowapp.com/>. (Acessado em: 24 de fevereiro de 2021). Citado na página 36.

BRANSKI, R. et al. Metodologia de estudo de casos aplicada à logística. In: . [S.l.: s.n.], 2010. Citado na página 55.

BRASIL, B. do. *Operações bancárias via WhatsApp - Você | Banco do Brasil*. 2019. <https://www.bb.com.br/pbb/pagina-inicial/atendimento/bb-nas-midias-sociais/operacoes-bancarias-via-WhatsApp#/>. Acessado em: 26 de setembro de 2019. Citado na página 27.

BRASIL. Comitê Técnico de Implementação do Software Livre no Governo Federal. *Diretrizes da Implementação do Software Livre no Governo Federal*: Resultado do planejamento estratégico do comitê técnico de implementação do software livre no governo federal, aprovado no dia 02.10.2003. Brasília, 2003. Acessado em: 04 de outubro de 2020. Citado 2 vezes nas páginas 50 e 75.

BRASIL. Governo Federal. *Sobre o Portal - Benefícios do Software Público*. Brasília, 2019. <https://www.gov.br/governodigital/pt-br/software-publico/sobre/sobre-o-portal>. Acessado em: 04 de outubro de 2020. Citado na página 50.

BRASIL. Ministério da Cidadania. Secretaria Especial da Cultura. *Como Funciona – Lei de Incentivo à Cultura*. Brasília, 2019. <http://leideincentivoacultura.cultura.gov.br/como-funciona/>. Acessado em: 10 de novembro de 2019. Citado 3 vezes nas páginas 13, 53 e 54.

BRASIL. Ministério da Cidadania. Secretaria Especial da Cultura. *O que é a Lei de Incentivo – Lei de Incentivo à Cultura*. Brasília, 2019. Acessado em: 10 de novembro de 2019. Citado 3 vezes nas páginas 24, 53 e 56.

BRASIL. Ministério da Ciência, Tecnologia e Inovações. *Estratégia Nacional de Ciência, Tecnologia e Inovação*. 2018. <http://www.finep.gov.br/images/a-finep/Politica/16_03_2018_Estrategia_Nacional_de_Ciencia_Tecnologia_e_Inovacao_2016_2022.pdf>. Acessado em: 24 de setembro de 2019. Citado na página 24.

BRASIL. Ministério da Ciência, Tecnologia, Inovações e Comunicações. *revisaodaestrategiadegovernancadigital20162019.pdf*. 2016. <https://www.gov.br/governodigital/pt-br/estrategia-de-governanca-digital/revisaodaestrategiadegovernancadigital20162019.pdf>. (Accessed on 11/02/2020). Citado 3 vezes nas páginas 42, 43 e 75.

BRASIL. Ministério da Ciência, Tecnologia, Inovações e Comunicações. *eDigital.pdf*. Brasília, 2018. <https://www.gov.br/governodigital/pt-br/estrategia-de-governanca-digital/eDigital.pdf>. (Accessed on 11/02/2020). Citado na página 42.

BRASIL. Ministério da Economia. Instituto de Pesquisa Econômica Aplicada. *INOVAÇÃO NO SETOR PÚBLICO. teoria, tendências e casos no Brasil*. 2017. <http://inova.gov.br/wp-content/uploads/2017/10/171002_inovacao_no_setor_publico.pdf>. Acessado em: 23 de setembro de 2019. Citado na página 23.

BRASIL. Ministério do Planejamento, Desenvolvimento e Gestão. Secretaria de Tecnologia da Informação. *Portaria nº 46, de 28 de setembro de 2016. Dispõe sobre a disponibilização de Software Público Brasileiro e dá outras providências*. Brasília, 2016. Acessado em: 04 de outubro de 2020. Citado na página 50.

BRASIL. Presidência da República. Casa Civil. Subchefia para Assuntos Jurídicos. *Decreto de 29 de Outubro de 2003. Institui Comitês Técnicos do Comitê Executivo do Governo Eletrônico e dá outras providências*. Brasília, 2019. Acessado em: 04 de outubro de 2020. Citado na página 50.

BRASIL. Presidência da República. Secretaria-Geral. Subchefia para Assuntos Jurídicos. *Decreto nº 8.638, de 15 de Janeiro de 2016. Revogado pelo Decreto nº 10.332, de 2020. Institui a Política de Governança Digital no âmbito dos órgãos e das entidades da administração pública federal direta, autárquica e fundacional*. Brasília, 2016. Acessado em: 08 de março de 2021. Citado na página 23.

BUXTON, B. *Sketching User Experiences: Getting the Design Right and the Right Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. 239-244 p. ISBN 0123740371, 9780123740373. Citado na página 46.

CAVALCANTE, P.; CAMõES, M. Do the brazilian innovations in public management constitute a new model? *RAI Revista de Administração e Inovação*, v. 14, n. 1, p. 90 – 96, 2017. ISSN 1809-2039. Disponível em: <http://www.sciencedirect.com/science/article/pii/S180920391631124X>. Citado na página 23.

CHAUDHURI, S.; DAYAL, U.; NARASAYYA, V. An overview of business intelligence technology. *Commun. ACM*, ACM, New York, NY, USA, v. 54, n. 8, p. 88–98, ago. 2011. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1978542.1978562>. Citado 2 vezes nas páginas 48 e 49.

COLBY, K. M.; WEBER, S.; HILF, F. D. *Artificial Paranoia*. New York, NY, USA: Elsevier Science Inc., 1971. v. 2. 1-25 p. Citado 2 vezes nas páginas 27 e 43.

ELASTIC. *Kibana — your window into the Elastic Stack | Kibana Guide [master] | Elastic*. 2019. <https://www.elastic.co/guide/en/kibana/master/introduction.html>. Acessado em: 18 de outubro de 2020. Citado na página 49.

ELASTIC. *Query DSL | Elasticsearch Reference [7.9] | Elastic*. 2020. <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>. Acessado em: 18 de outubro de 2020. Citado na página 49.

ELASTICSEARCH B.V. *What is Elasticsearch | Elastic*. 2019. <https://www.elastic.co/pt/what-is/elasticsearch>. Acessado em: 05 de novembro de 2019. Citado na página 49.

FACEBOOK. *F8 2018: Novas Ferramentas para empresas e pessoas aprofundarem conexões no Messenger | Facebook Newsroom Brazil*. 2018. <https://about.fb.com/br/news/2018/05/f8-2018-novas-ferramentas-para-empresas-e-pessoas-aprofundarem-conexoes-no-messenger/>. Acessado em: 20 de novembro de 2019. Citado 2 vezes nas páginas 24 e 75.

FALK, J. et al. Pica: Proactive intelligent conversational agent for interactive narratives. In: *Proceedings of the 18th International Conference on Intelligent Virtual Agents*. New York, NY, USA: ACM, 2018. (IVA '18), p. 141–146. ISBN 978-1-4503-6013-5. Disponível em: <http://doi.acm.org/10.1145/3267851.3267892>. Citado na página 28.

FERGUSON, M. Estratégias de governo eletrônico: o cenário internacional em desenvolvimento. In: EISENBERG, J.; CEPIK, M. *Internet e política: teoria e prática da democracia eletrônica*. Belo Horizonte: Editora UFMG, 2002. p. 103 – 140. Citado na página 23.

FOLLOW, S. C. A. *How to Build Great Bots for Big Companies - Chatbots Magazine*. 2017. <https://chatbotsmagazine.com/building-great-bots-an-enterprise-chatbot-methodology-c89aa188da2f>. Acessado em: 03 de novembro de 2019. Citado 3 vezes nas páginas 13, 44 e 45.

FOUNDATION, D. S. *The Web framework for perfectionists with deadlines | Django*. 2021. <https://www.djangoproject.com/>. (Acessado em: 24 de fevereiro de 2021). Citado na página 35.

FRAMPTON, M.; LEMON, O. Recent research advances in reinforcement learning in spoken dialogue systems. *The Knowledge Engineering Review*, Cambridge University Press, v. 24, n. 4, p. 375–408, 2009. Citado na página 34.

FRASER, N. M.; GILBERT, G. Simulating speech systems. *Computer Speech & Language*, v. 5, n. 1, p. 81 – 99, 1991. ISSN 0885-2308. Disponível em: <http://www.sciencedirect.com/science/article/pii/088523089190019M>. Citado 2 vezes nas páginas 17 e 46.

GIL, A. *Métodos e técnicas de pesquisa social*. [S.l.]: Atlas, 2008. ISBN 9788522451425. Citado na página 55.

GMBH, R. T. *Choosing a Pipeline*. 2018. <https://legacy-docs.rasa.com/docs/nlu/0.13.0/choosing_pipeline/>. Acessado em: 04 de outubro de 2020. Citado na página 58.

GMBH, R. T. *Contributors to RasaHQ/rasa*. 2018. <https://github.com/RasaHQ/rasa/graphs/contributors?from=2017-10-07&to=2018-03-27&type=c>. (Acessado em: 05 de novembro de 2020). Citado 2 vezes nas páginas 15 e 58.

GMBH, R. T. *Language Support*. 2018. <https://legacy-docs.rasa.com/docs/nlu/0.13.0/languages/>. Acessado em: 04 de outubro de 2020. Citado na página 58.

GMBH, R. T. *Training and Policies*. 2018. <https://legacy-docs.rasa.com/docs/core/0.10.4/policies/>. Acessado em: 04 de outubro de 2020. Citado na página 58.

GMBH, R. T. *Rasa NLU in Depth: Part 1 – Intent Classification*. 2019. <https://blog.rasa.com/rasa-nlu-in-depth-part-1-intent-classification/>. Acessado em: 10 de outubro de 2019. Citado 2 vezes nas páginas 29 e 30.

GNU PROJECT. Free Software Foundation. *Filosofia do Projeto GNU*. 2009. <https://www.gnu.org/philosophy/floss-and-foss.pt-br.html>. Acessado em: 04 de outubro de 2020. Citado na página 50.

GNU PROJECT. Free Software Foundation. *O que é o software livre?* 2012. <https://www.gnu.org/philosophy/free-sw.html>. Acessado em: 04 de outubro de 2020. Citado na página 50.

GOOGLE. *Build chatbots with Dialogflow | Google Developers*. 2021. <https://developers.google.com/learn/pathways/chatbots-dialogflow>. (Acessado em: 24 de fevereiro de 2021). Citado na página 36.

GUIMARãES, T. d. A.; MEDEIROS, P. H. R. A relação entre governo eletrônico e governança eletrônica no governo federal brasileiro. *Cadernos EBAPE.BR*, scielo, v. 3, p. 01 – 18, 12 2005. ISSN 1679-3951. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1679-39512005000400004&nrm=iso>. Citado na página 23.

HARMS, J.-G. et al. Approaches for dialog management in conversational agents. *IEEE Internet Computing*, PP, p. 1–1, 11 2018. Citado 8 vezes nas páginas 13, 28, 29, 30, 31, 32, 33 e 34.

HUANG, J.; ZHOU, M.; YANG, D. Extracting chatbot knowledge from online discussion forums. In: *IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence*. [S.l.: s.n.], 2007. p. 423–428. Citado 3 vezes nas páginas 27, 44 e 47.

HUBOITO. *Huboito/Hubot: A customizable life embetterment robot*. 2011. <https://hubot.github.com/>. Acessado em: 28 de setembro de 2020. Citado na página 36.

INC., F. *React Native · A framework for building native apps using React*. 2021. <https://reactnative.dev/>. (Acessado em: 24 de fevereiro de 2021). Citado na página 35.

INC., F. *React – Uma biblioteca JavaScript para criar interfaces de usuário*. 2021. <https://pt-br.reactjs.org/>. (Acessado em: 24 de fevereiro de 2021). Citado na página 35.

INICIATIVE, O. S. *History of the OSI | Open Source Initiative*. 2018. <https://opensource.org/history>. Acessado em: 04 de outubro de 2020. Citado na página 51.

INNGAGE. *O que é uma boa taxa de retenção? • Inngage*. 2021. <https://inngage.com.br/2018/04/02/o-que-e-uma-boa-taxa-de-retencao/>. (Acessado em: 24 de fevereiro de 2021). Citado 2 vezes nas páginas 13 e 48.

INTELLIGENCE, B. I. *80% of businesses want chatbots by 2020 - Business Insider*. 2016. <https://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12>. Acessado em: 20 de novembro de 2019. Citado na página 24.

LACERDA, A. R. T. de; AGUIAR, C. S. R. Floss faq chatbot project reuse: How to allow nonexperts to develop a chatbot. In: *Proceedings of the 15th International Symposium on Open Collaboration*. New York, NY, USA: ACM, 2019. (OpenSym '19), p. 3:1–3:8. ISBN 978-1-4503-6319-8. Disponível em: <http://doi.acm.org/10.1145/3306446.3340823>. Citado 4 vezes nas páginas 17, 41, 55 e 70.

LISON, P. A hybrid approach to dialogue management based on probabilistic rules. *Comput. Speech Lang.*, Academic Press Ltd., London, UK, UK, v. 34, n. 1, p. 232–255, nov. 2015. ISSN 0885-2308. Disponível em: <http://dx.doi.org/10.1016/j.csl.2015.01.001>. Citado na página 35.

MCTEAR, M.; CALLEJAS, Z.; GRIOL, D. Dialog management. In: _____. *The Conversational Interface: Talking to Smart Devices*. Cham: Springer International Publishing, 2016. p. 209–233. ISBN 978-3-319-32967-3. Disponível em: <https://doi.org/10.1007/978-3-319-32967-3_10>. Citado 4 vezes nas páginas 32, 33, 34 e 35.

MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. Citado na página 30.

Misra, D. C. Defining e-government: A citizen-centric criteria-based approach. 2006. Citado 2 vezes nas páginas 23 e 75.

MODELOP. *ModelOps RFP Template | ModelOps and MLOps Resource Hub*. 2020. <https://modelop.io/modelops-rfp/>. (Acessado em: 26 de novembro de 2020). Citado na página 73.

MOORE, R. J. et al. Conversational ux design. In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2017. (CHI EA '17), p. 492–497. ISBN 9781450346566. Disponível em: <https://doi.org/10.1145/3027063.3027077>. Citado 4 vezes nas páginas 17, 43, 44 e 76.

NASS, C.; STEUER, J.; TAUBER, E. R. Computers are social actors. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 1994. (CHI '94), p. 72–78. ISBN 0-89791-650-6. Disponível em: <http://doi.acm.org/10.1145/191666.191703>. Citado 2 vezes nas páginas 43 e 44.

NATURALNODE. *NaturalNode/natural: general natural language facilities for node*. 2011. <https://github.com/NaturalNode/natural>. Acessado em: 16 de novembro de 2019. Citado na página 36.

Noh, H. et al. An example-based approach to ranking multiple dialog states for flexible dialog management. *IEEE Journal of Selected Topics in Signal Processing*, v. 6, n. 8, p. 943–958, Dec 2012. Citado na página 34.

PRZEGALINSKA, A. et al. In bot we trust: A new methodology of chatbot performance measures. 2019. Disponível em: <https://www.journals.elsevier.com/business-horizons>. Citado na página 27.

RAMESH, K. et al. A survey of design techniques for conversational agents. In: KAUSHIK, S. et al. (Ed.). *Information, Communication and Computing Technology*. Singapore: Springer Singapore, 2017. p. 336–350. ISBN 978-981-10-6544-6. Citado na página 28.

RASA. *Actions*. 2019. <https://rasa.com/docs/rasa/core/actions/#utterance-actions>. Acessado em: 13 de dezembro de 2019. Citado na página 39.

RASA. *Policies*. 2019. <https://rasa.com/docs/rasa/core/policies>. Acessado em: 13 de dezembro de 2019. Citado 3 vezes nas páginas 17, 38 e 39.

Rasa Technologies GmbH. *Fallback Policy*. 2017. <https://rasa.com/docs/rasa/core/policies/#fallback-policy>. Acessado em: 03 de novembro de 2019. Citado na página 45.

RED HAT, INC. *O que é o open source?* 2020. <https://www.redhat.com/pt-br/topics/open-source/what-is-open-source>. Acessado em: 04 de outubro de 2020. Citado na página 50.

ROCKETCHAT. *RocketChat/hubot-natural: Natural Language Processing Chatbot for RocketChat*. 2017. <https://github.com/RocketChat/hubot-natural>. Acessado em: 16 de novembro de 2019. Citado 4 vezes nas páginas 13, 36, 37 e 38.

ROCKETCHAT. *Contributors to RocketChat/hubot-natural*. 2018. <https://github.com/RocketChat/hubot-natural/graphs/contributors?from=2017-10-15&to=2018-04-07&type=c>. (Acessado em: 05 de novembro de 2020). Citado 2 vezes nas páginas 15 e 58.

Scholl, H. J. E-government: a special case of ict-enabled business process change. In: *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*. [S.l.: s.n.], 2003. p. 12 pp.–. Citado na página 23.

Shahin, M.; Ali Babar, M.; Zhu, L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, v. 5, p. 3909–3943, 2017. Citado na página 72.

SOLUTIONS, C. G. *chatbot2*. 2018. <https://www.cgsinc.com/sites/default/files/media/resources/pdf/CGS_Consumer%2BCustServ%2Binfographic%2B2018.pdf>. Acessado em: 20 de novembro de 2019. Citado na página 24.

STALLMAN, R. *FLOSS e FOSS - GNU Project - Free Software Foundation*. 2012. <https://www.gnu.org/philosophy/open-source-misses-the-point.html>. Acessado em: 04 de outubro de 2020. Citado na página 51.

STALLMAN, R. *Software Livre é Ainda Mais Importante Agora*. 2015. <https://www.gnu.org/philosophy/floss-and-foss.pt-br.html>. Acessado em: 04 de outubro de 2020. Citado na página 50.

STALLMAN, R. *FLOSS e FOSS*. 2016. <https://www.gnu.org/philosophy/floss-and-foss.pt-br.html>. Acessado em: 04 de outubro de 2020. Citado na página 51.

SUTHERLAND, J.; SCHWABER, K. *Scrum Guide | Scrum Guides*. 2017. <https://www.scrumguides.org/scrum-guide.html>. Acessado em: 12 de novembro de 2019. Citado na página 55.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. 2018. Citado 21 vezes nas páginas 13, 14, 15, 57, 60, 67, 93, 95, 96, 97, 98, 99, 100, 106, 110, 112, 113, 114, 115, 117 e 118.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Analisar o Histórico de Conversa · Issue #60*. Brasília, 2018. <https://github.com/lappis-unb/tais/issues/60#issuecomment-420854938>. Acessado em: 20 de novembro de 2019. Citado na página 61.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Contributors to lappis-unb/tais*. Brasília, 2018. <https://github.com/lappis-unb/tais/graphs/contributors>. Acessado em: 25 de novembro de 2019. Citado na página 70.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Estudar chatbot UX · Issue #20*. Brasília, 2018. <https://github.com/lappis-unb/tais/issues/20>. Acessado em: 17 de novembro de 2019. Citado na página 58.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Estudo de Interação*. Brasil, 2018. <https://github.com/lappis-unb/tais/wiki/Estudo-de-Intera%C3%A7%C3%A3o>. Acessado em: 20 de novembro de 2019. Citado 2 vezes nas páginas 63 e 64.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Estudo sobre ferramentas de bots*. Brasília, 2018. <https://github.com/lappis-unb/tais/wiki/Estudo-sobre-ferramentas-de-bots>. Acessado em: 17 de novembro de 2019. Citado 3 vezes nas páginas 36, 58 e 59.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Estudo sobre melhores práticas de bots*. Brasília, 2018. <https://github.com/lappis-unb/tais/wiki/Estudo-sobre-melhores-pr%C3%A1ticas-de-bots>. Acessado em: 17 de novembro de 2019. Citado na página 58.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Monitorar as instâncias oficiais da Taís · Issue #129*. Brasília, 2018. <https://github.com/lappis-unb/tais/issues/129>. Acessado em: 20 de novembro de 2019. Citado na página 63.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Release 1.0.0: Update bot answers*. Brasília, 2018. <https://github.com/lappis-unb/tais/releases/tag/1.0.0>. Acessado em: 16 de novembro de 2019. Citado na página 56.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Rouana Homologação v1.0*. Brasília, 2018. <https://github.com/lappis-unb/tais/wiki/Rouana-Homologa%C3%A7%C3%A3o-v1.0>. Acessado em: 16 de novembro de 2019. Citado 2 vezes nas páginas 56 e 57.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Tais Homologação v2.0*. Brasília, 2018. <https://github.com/lappis-unb/tais/wiki/Tais-Homologa%C3%A7%C3%A3o-v2.0>. Acessado em: 20 de novembro de 2019. Citado 4 vezes nas páginas 13, 42, 61 e 62.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Elastic - dashboards de monitoramento de chatbots · Issue #33*. Brasília, 2019. <https://github.com/lappis-unb/tais/issues/337>. Acessado em: 28 de novembro de 2019. Citado 2 vezes nas páginas 17 e 65.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Esquema simplificado de um arquitetura de chatbot*. Brasília, 2019. Apresentação Chatbot Indie - Isso existe? - Campus Party BSB 2019. Acessado em: 25 de novembro de 2020. Citado 2 vezes nas páginas 13 e 41.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Estudo sobre metricas para bots · lappis-unb/tais Wiki*. Brasília, 2019. <https://github.com/lappis-unb/tais/wiki/Estudo-sobre-metricas-para-bots>. Acessado em: 28 de novembro de 2019. Citado 3 vezes nas páginas 17, 66 e 67.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Issues de Conteúdo · Repositório lappis-unb/tais*. Brasília, 2019. <https://github.com/lappis-unb/tais/issues?q=is%3Aissue+label%3Aconteudo+is%3Aclosed>. Acessado em: 09 de novembro de 2020. Citado na página 63.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Tais - Contributors Apr 3, 2019 – May 3, 2019*. Brasília, 2019. <https://github.com/lappis-unb/tais/graphs/contributors?from=2019-04-03&to=2019-05-03&type=c>. Acessado em: 25 de novembro de 2020. Citado 2 vezes nas páginas 15 e 69.

UNIVERSIDADE DE BRASÍLIA. Laboratório Avançado de Pesquisa, Produção e Inovação em Software - LAPPIS. *Tais - Contributors Aug 3, 2018 – Apr 3, 2019*. Brasília, 2019. <https://github.com/lappis-unb/tais/graphs/contributors?from=2018-08-03&to=2019-04-03&type=c>. Acessado em: 25 de novembro de 2020. Citado 2 vezes nas páginas 15 e 69.

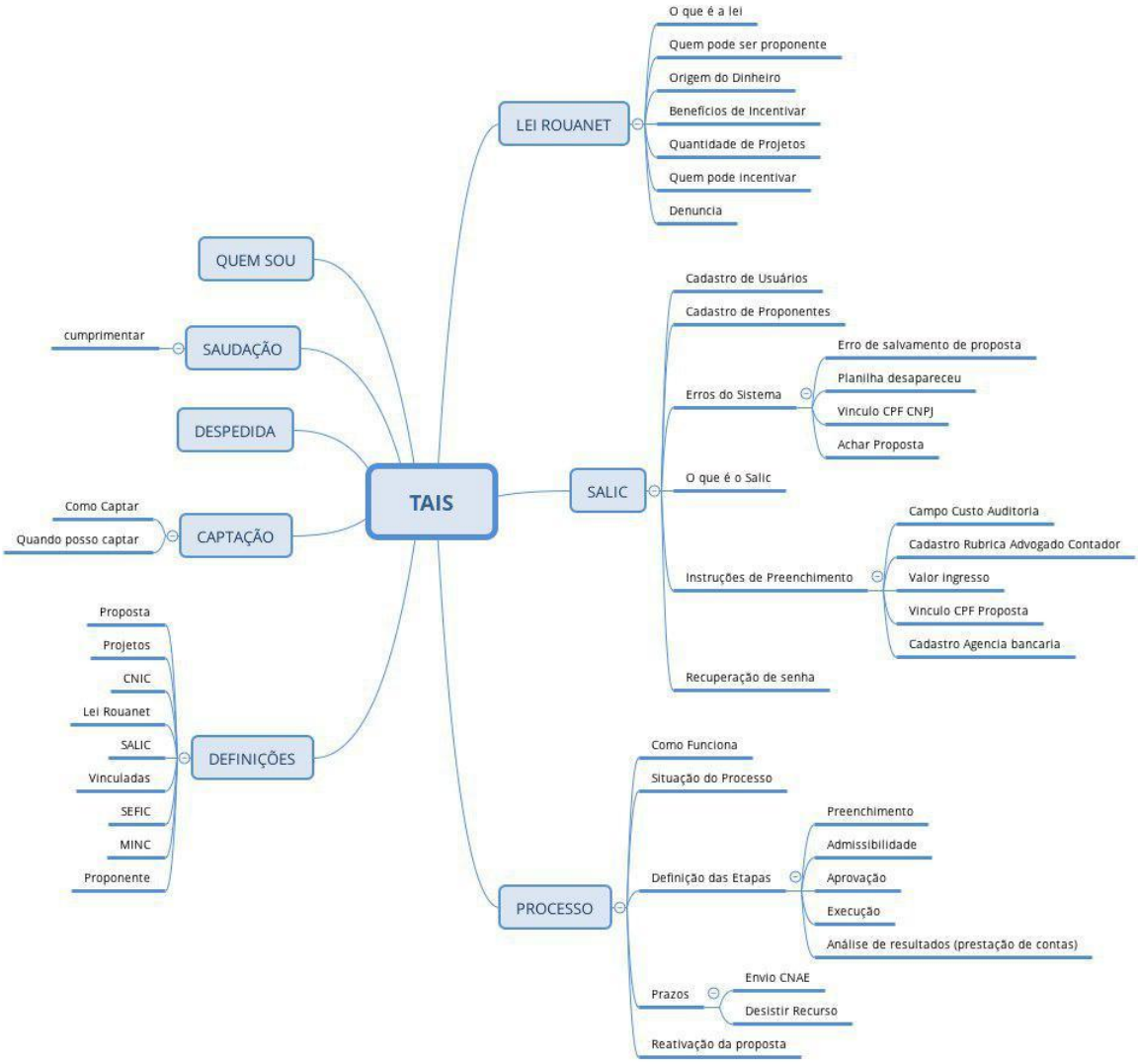
WEIZENBAUM, J. Eliza a computer program for the study of natural language communication between man and machine. In: *Communications of the ACM*. [S.l.: s.n.], 1966. v. 9, n. 1, p. 36–45. Citado na página 27.

Anexos

ANEXO A – Fluxos Chatbot Tais

A.1 Mapa mental - Conteúdo da Tais

Figura 16 – Mapa mental - Conteúdo da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)



A.2 Fluxo - Versão 0

A.3 Fluxo - Versão 1

A.4 Fluxo - Versão 2

A.5 Fluxo - Versão 3

A.6 Fluxo - Versão 4

A.7 Fluxo - Versão 5

Figura 17 – Versão 0 - Fluxo da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

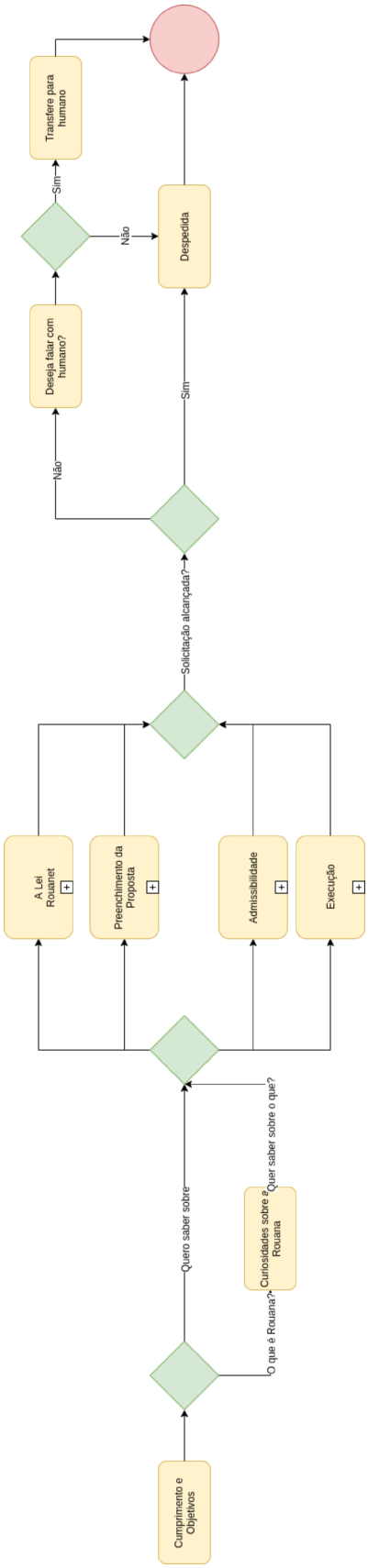


Figura 18 – Versão 1 - Fluxo da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

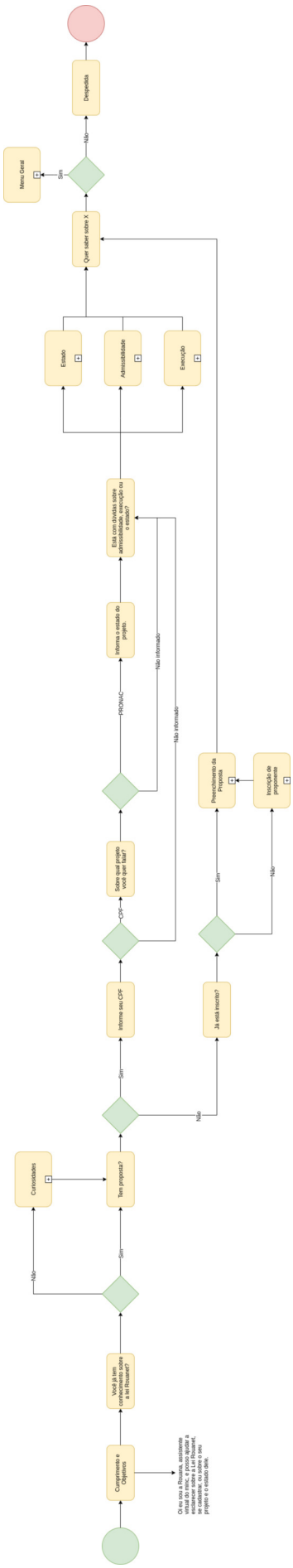


Figura 19 – Versão 2 - Fluxo da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

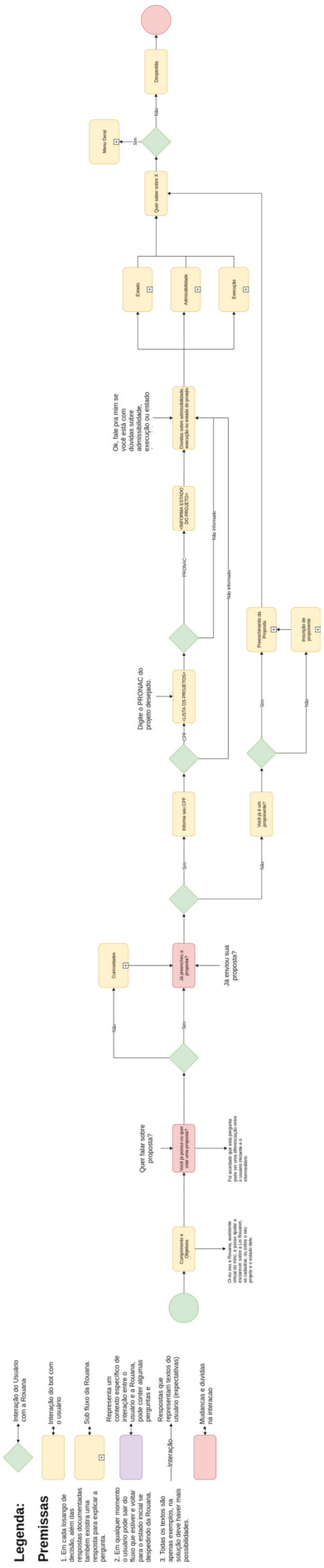


Figura 20 – Versão 3 - Fluxo da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

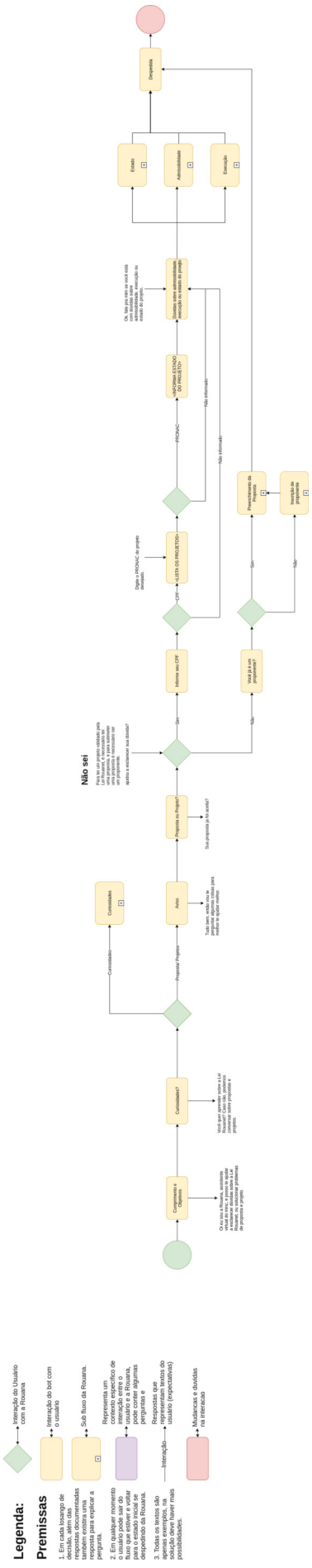


Figura 21 – Versão 4 - Fluxo da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

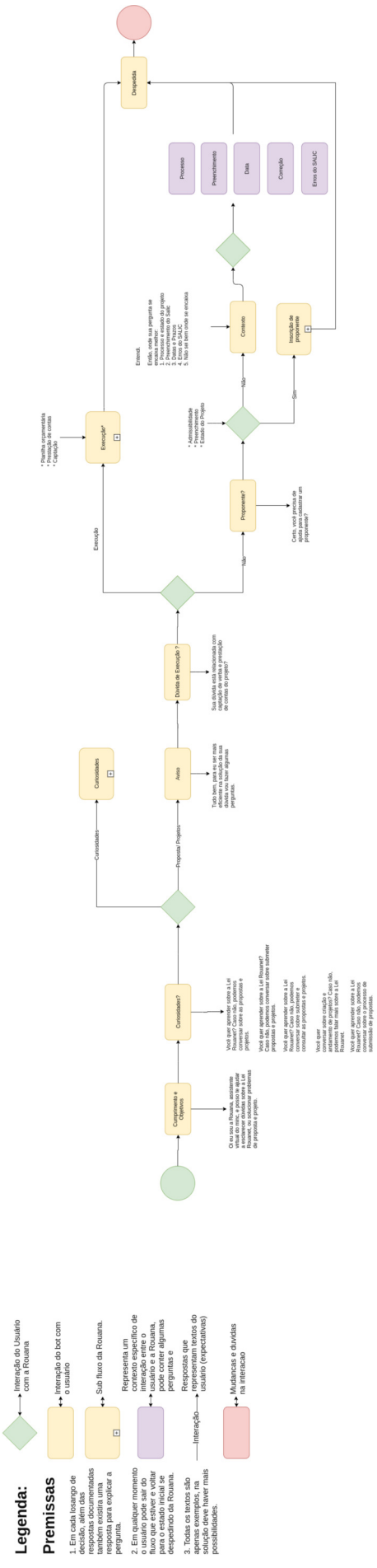
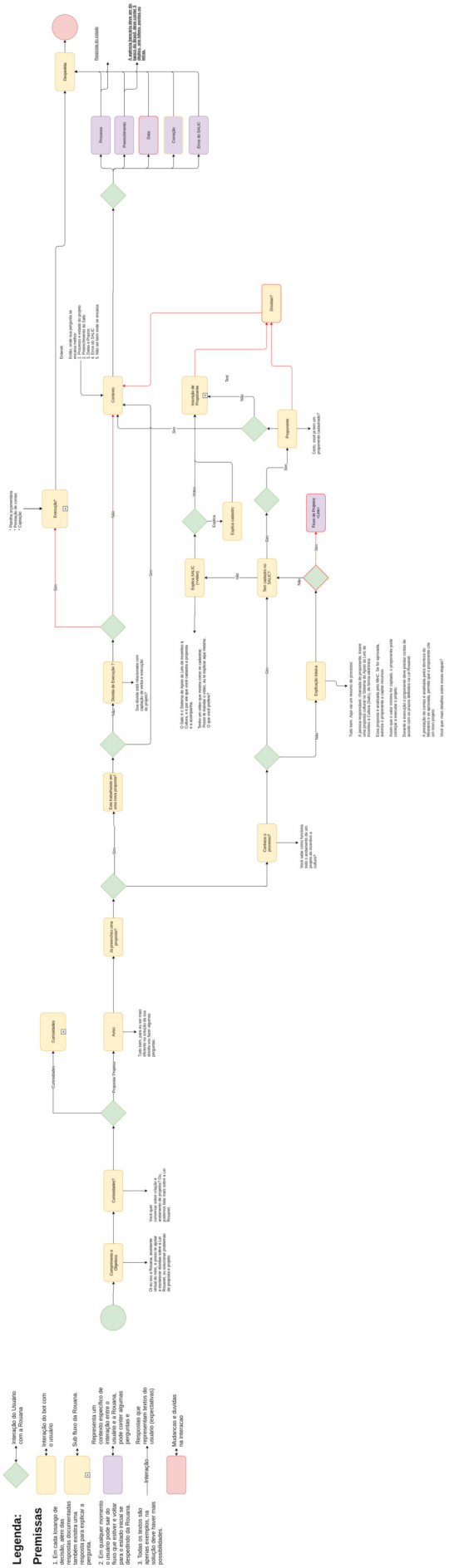


Figura 22 – Versão 5 - Fluxo da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)



ANEXO B – Questionários Aplicados

B.1 Estudo sobre o uso do Chatbot Rouana (UnB/LAPPIS

Estudo sobre o uso da assistente virtual Rouana (UnB/Lappis)

Se você permitir, usaremos suas respostas anonimizadas em uma pesquisa sobre o uso da assistente virtual Rouana desenvolvido com a intenção de ajudar seus usuários a entender o funcionamento da Lei Rouanet.

*Obrigatório

1. **Nome** *

2. **Email** *

3. **Sobre as minhas respostas anônimas a este questionário:**
   Marque todas que se aplicam.

   ☐ Permito o uso dos dados anonimizados que forneço neste questionário sobre o projeto MinC e UnB/Lappis

Percepção geral sobre Rouana

Como foi a sua experiência geral com a Rouana

4. **Em geral, o que você achou da interação com a assistente virtual Rouana?** *
   Marcar apenas uma oval.

   ◯ Muito bom
   ◯ Bom
   ◯ Neutro
   ◯ Ruim
   ◯ Muito ruim

5. **Por quê?** *

6. **Como foi o processo de tirar dúvidas com a Rouana?** *
Marcar apenas uma oval.

- ◯ Muito confortável
- ◯ Confortável
- ◯ Neutro
- ◯ Desconfortável
- ◯ Muito desconfortável

7. **Por quê?** *

_____

_____

_____

_____

_____

8. **A Rouana foi projetada para interagir com cidadãs e cidadãos no auxílio do processo de incentivo à cultura. Como você se sente em relação a isso?** *
Marcar apenas uma oval.

- ◯ Muito motivada(o)
- ◯ Moderadamente motivada(o)
- ◯ Pouco motivada(o)
- ◯ Nem um pouco motivada(o)

Sobre as respostas dadas pela Rouana

Análise da performance da Rouana (taxa de acerto das respostas)

9. **Pensando na sua conversa inteira com a Rouana, como foram as respostas que você obteve?** *
Marcar apenas uma oval.

- ◯ Muito satisfatórias
- ◯ Satisfatórias
- ◯ Neutro
- ◯ Insatisfatórias
- ◯ Muito insatisfatórias

10. **Qual foi a taxa de acerto das respostas dadas pela Rouana?** *
Marcar apenas uma oval.

	0	1	2	3	4	5	
(errou tudo)	◯	◯	◯	◯	◯	◯	(acertou tudo)

11. **Em linhas gerais, como você descreve as respostas dadas pela Rouana?** *

Sobre o conteúdo passado pela Rouana

Análise sobre o fluxo das interações com a Rouana

12. **De forma geral, o que você achou dos textos nas respostas da Rouana?** *

Marcar apenas uma oval.

- ◯ Muito claros
- ◯ Claros
- ◯ Neutro
- ◯ Confusos
- ◯ Muito confusos

13. **Por quê?** *

14. **Caso a Rouana não tiver respondido corretamente, compartilhe essa experiência.**

15. **Como você avalia o tamanho do texto das respostas da Rouana?** *

Marcar apenas uma oval.

- ◯ Muito satisfatório
- ◯ Satisfatório
- ◯ Neutro
- ◯ Insatisfatório
- ◯ Muito insatisfatório

16. **Por quê?** *

17. **Quantas perguntas você precisou fazer até receber uma resposta certa (na maioria das questões)?** *

Marcar apenas uma oval.

- Obtive a resposta certa com apenas uma pergunta
- Foram necessárias fazer várias perguntas até obter a resposta correta
- Mesmo após várias perguntas, não obtive a resposta correta

18. **Gostaríamos de saber mais detalhes sobre a sua experiência com a Rouana. Compartilhe seu relato.**

Powered by
Google Forms

Fonte: (UNIVERSIDADE DE BRASÍLIA, 2018a)

B.2 Questionário - Taís Beta

Feedback sobre a interação com a assistente virtual Taís

*Obrigatório

1. **Nome** *

2. **Email** *

3. **Sobre as minhas respostas anônimas a este questionário:**
   Marque todas que se aplicam.

   ☐ Permito o uso dos dados anonimizados que forneço neste questionário sobre o projeto MinC e UnB/Lappis

Percepção geral

4. **Em geral, o que você achou da interação com a assistente virtual Taís?** *
   Marcar apenas uma oval.

   ◯ Muito boa
   ◯ Boa
   ◯ Neutro
   ◯ Ruim
   ◯ Muito ruim

5. **Por quê?** *

6. **Como foi o processo de tirar dúvidas com a Taís?** *
   Marcar apenas uma oval.

   ◯ Muito confortável
   ◯ Confortável
   ◯ Neutro
   ◯ Desconfortável
   ◯ Muito desconfortável

7. **Por quê?** *

8. **Avalie a habilidade da Taís atender iniciantes no mecanismo de incentivo fiscal e no Salic:**
Marcar apenas uma oval.

	1	2	3	4	5	
Incapaz de atender pessoas iniciantes	◯	◯	◯	◯	◯	Capaz de atender pessoas iniciantes

9. **Justifique sua avaliação** *

Sobre o conteúdo

10. **Pensando na sua conversa inteira com a Taís, como foram as respostas que você obteve?** *
Marcar apenas uma oval.

- ◯ Muito satisfatórias
- ◯ Satisfatórias
- ◯ Neutro
- ◯ Insatisfatórias
- ◯ Muito insatisfatórias

11. **Qual foi a taxa de acerto das respostas dadas pela Taís?** *
Marcar apenas uma oval.

	1	2	3	4	5	
(errou tudo)	◯	◯	◯	◯	◯	(acertou tudo)

12. **Quais tópicos a Taís falhou em responder para você? (Caso isso tenha acontecido)**

13. **O que você achou dos textos nas respostas da Taís?** *
Marcar apenas uma oval.

- ◯ Muito claros
- ◯ Claros
- ◯ Neutro
- ◯ Confusos
- ◯ Muito confusos

14. **Por quê?** *

15. **Dúvidas? Sugestões? Pontos para reflexão? Compartilhe com a gente:**

Powered by
Google Forms

Fonte: (UNIVERSIDADE DE BRASÍLIA, 2018a)

ANEXO C – *Dashboard* do Kibana - Tais

Figura 23 – *Dashboard* Perfil de usuário - Kibana da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

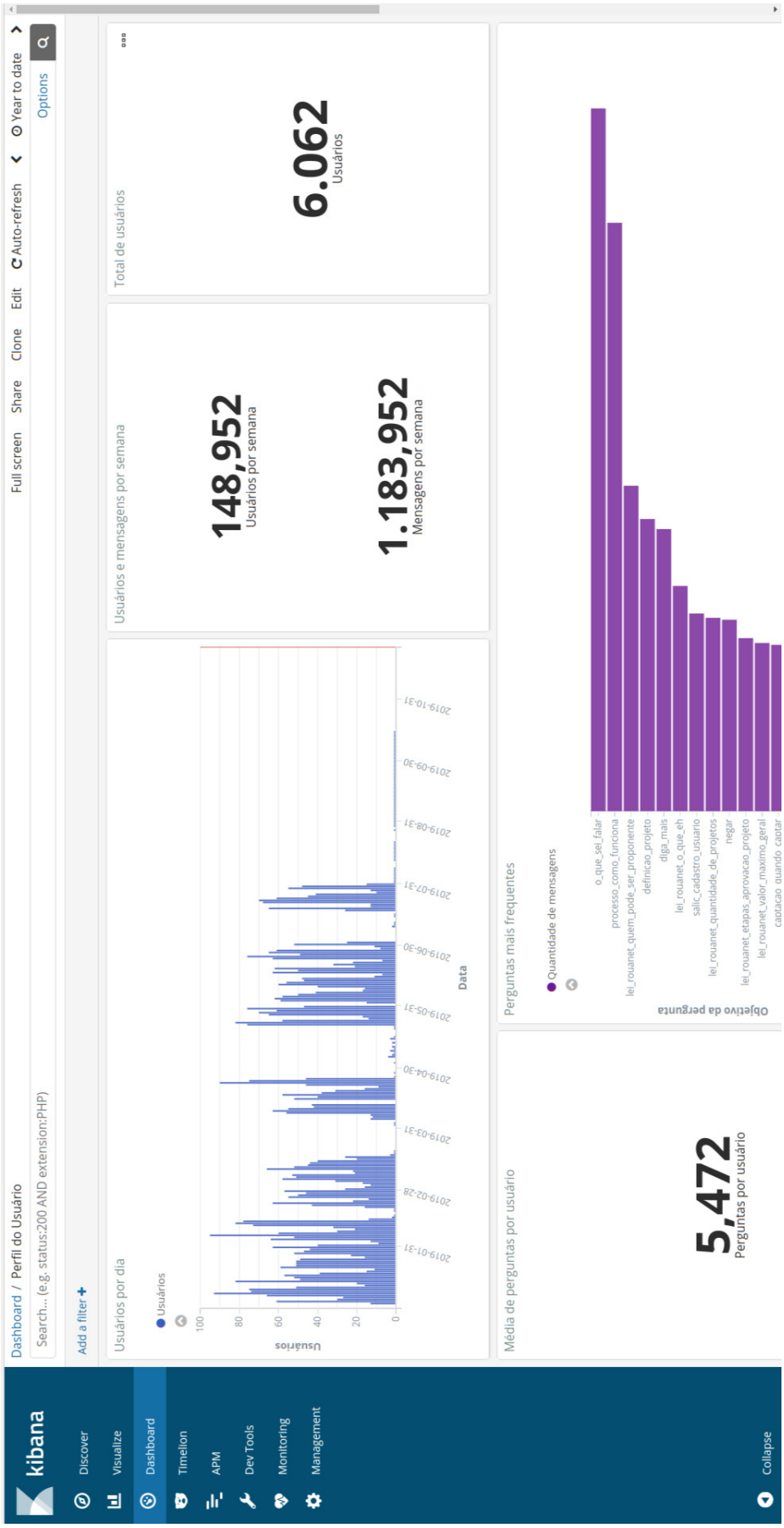
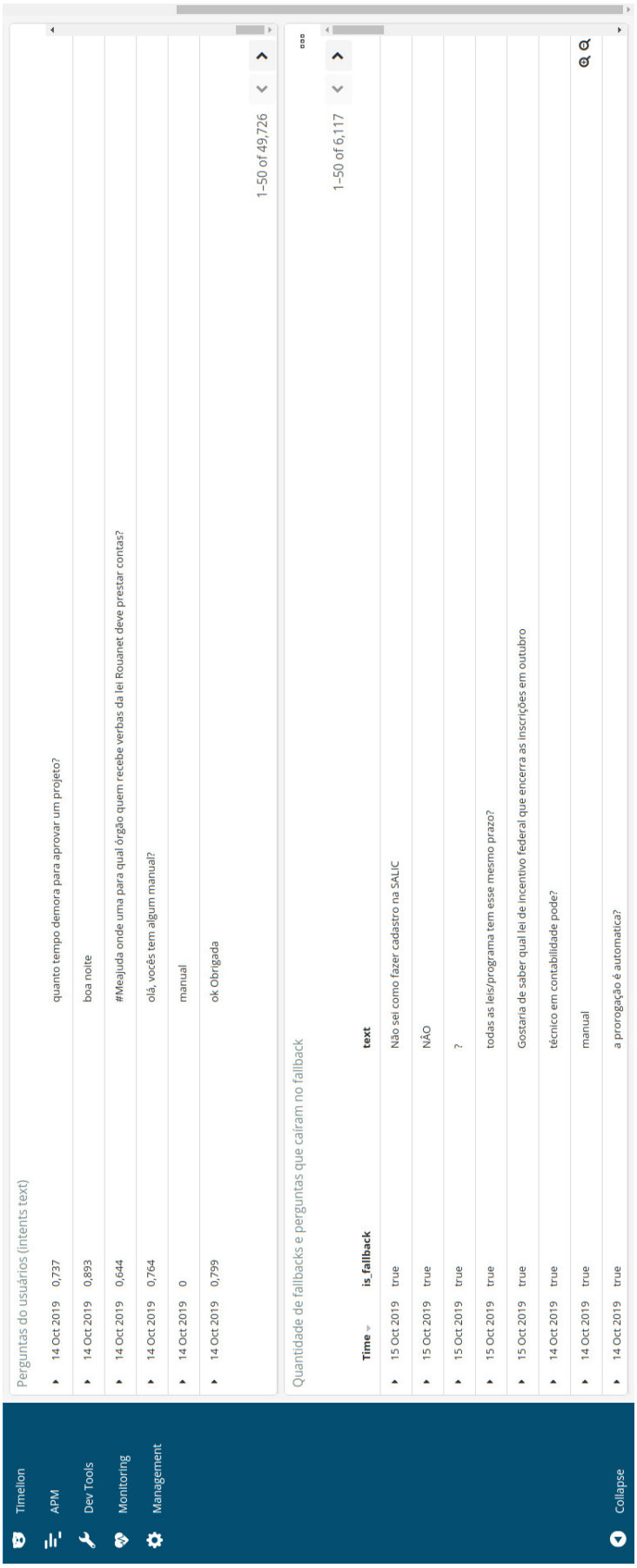


Figura 24 – *Dashboard* Perfil de usuário (Parte 2) - Kibana da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

Figura 25 – *Dashboard* Paloma's - Kibana da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)

Figura 26 – *Dashboard* Paloma's (Parte 2) - Kibana da Tais (UNIVERSIDADE DE BRASÍLIA, 2018a)



ANEXO D – BotFlow - Tais

Figura 27 – BotFlow - Página de *intents* (UNIVERSIDADE DE BRASÍLIA, 2018a)

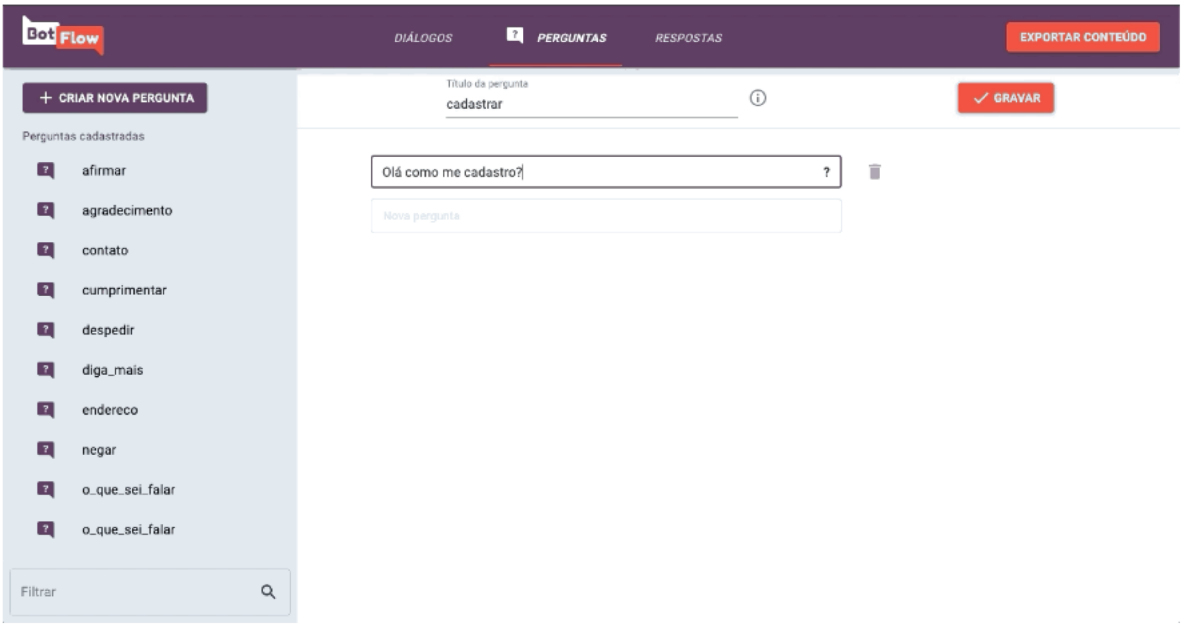


Figura 28 – BotFlow - Página de *utterances* (UNIVERSIDADE DE BRASÍLIA, 2018a)

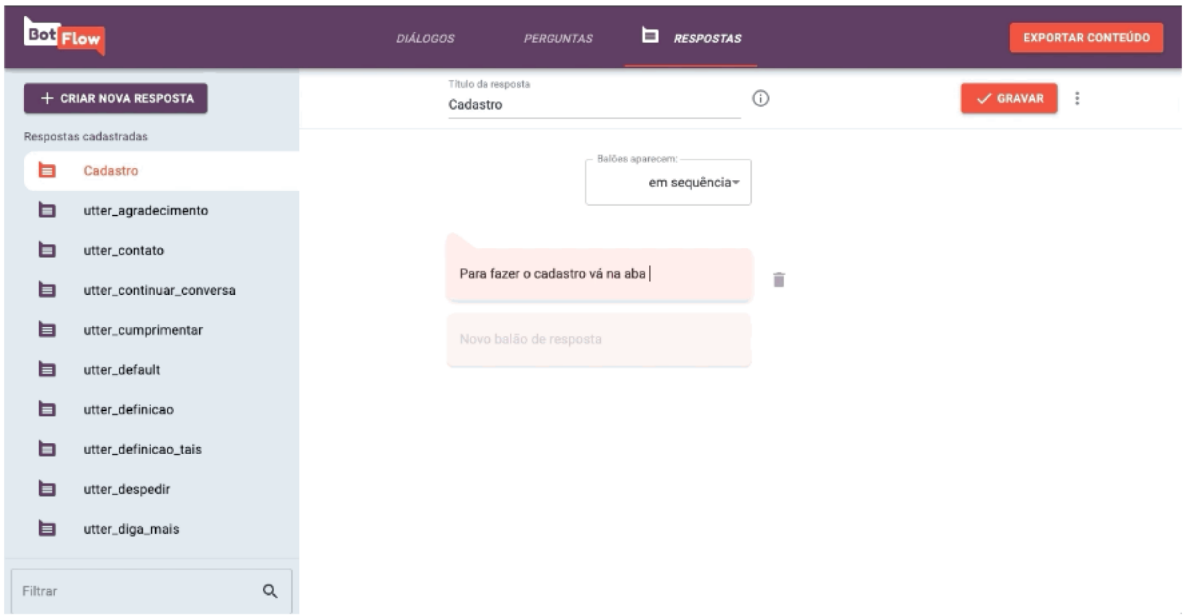


Figura 29 – BotFlow - Página de *stories* (UNIVERSIDADE DE BRASÍLIA, 2018a)