UNIVERSIdade FEEVALE

TIAGO LUIS STEIL

DESENVOLVIMENTO DE UM CHATBOT PARA ATENDIMENTO COM O WATSON

Novo Hamburgo

2019

Tiago luis Steil

DESENVOLVIMENTO DE UM CHATBOT PARA ATENDIMENTO COM O WATSON

Trabalho de Conclusão de Curso

apresentado como requisito parcial

à obtenção do grau de Bacharel em

Ciência da Computação pela

Universidade Feevale

Orientador: Rodrigo Rafael Villarreal Goulart

Novo Hamburgo

2019

Agradecimentos

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

Aos meus pais que sempre me incentivaram durante toda a minha formação, a minha noiva, que sempre esteve ao meu lado e me apoiou em toda a jornada e às pessoas que convivem comigo diariamente, minha gratidão, pelo apoio emocional nos períodos mais difíceis do trabalho.

Resumo

Esse trabalho tem como objetivo apresentar o uso de *chatbots* com Processamento de Linguagem Natural (PLN), utilizando os serviços do Watson da IBM. Os *chatbots* estão sendo cada vez mais utilizados pelas grandes empresas para atender de maneira mais ágil os usuários, reduzindo o tempo de espera entre os atendimentos e a necessidade de intervenção humana. Desta forma, este trabalho busca resolver o problema atual no atendimento de clientes de um software da empresa Safetech informática que é oferecido na modalidade de SAAS. Para isso, foi desenvolvido neste trabalho um *chatbot* capaz de conversar com os usuários que possuem dúvidas sobre o sistema. Para construção deste atendente foram utilizados os serviços do Watson *Assistant*, Watson *Natural Language Understanding* e Watson *Discovery* da IBM, integrados através do software desenvolvido, utilizando diferentes tecnologias. Avaliamos o desempenho do *chatbot* através da análise do histórico das conversas e com a aplicação de questionários junto aos utilizadores do atendente desenvolvido. Os resultados obtidos foram satisfatórios. Conforme os resultados, 65,2% dos usuários aprenderam alguma coisa utilizando o sistema, e em relação as respostas, o *chatbot* respondeu corretamente 50% das perguntas.

Palavras-chave: Chatbots. Watson. IBM. Processamento de Linguagem Natural. DeepQA.

Abstract

This paper aims to present the use of chatbots with Natural Language Processing (NLP) using Watson IBM services. Chatbots are being increasingly used by large companies to more quickly serve users, reducing waiting times between calls and the need for human intervention. In this way, this work seeks to solve the current problem in the customer service of a software of the company Safetech Informática that is offered in the SAAS modality. For this, a chatbot was developed in this work able to talk with users who have doubts about the system. The Watson Assistant, Watson Natural Language Understanding and Watson Discovery services from IBM, were used to construct this attendant, integrated through software developed using different technologies. We evaluated the performance of chatbot by analyzing the history of the conversations and applying questionnaires to the users of the developed attendant. The results were satisfactory. According to the results, 65.2% of users learned something using the system, and regarding the answers, chatbot correctly answered 50% of the questions.

Keywords: Chatbots. Watson. IBM. Natural Language Processing. DeepQA.

Lista de Figuras

[Figura 1 – Quadro do jogo Jeopardy 19](#_Toc13260493)

[Figura 2 – Alto nível da arquitetura DeepQA 22](#_Toc13260494)

[Figura 3 – Dimensões de evidência para uma pista do Jeopardy 23](#_Toc13260495)

[Figura 4 – Intenção e entidade em uma conversa 34](#_Toc13260496)

[Figura 5 – Fluxo de importação dos documentos para o *Discovery* 35](#_Toc13260497)

[Figura 6 – Fluxo do serviço *Tone Analyzer* 36](#_Toc13260498)

[Figura 7 – Representação de alto nível do NLU 37](#_Toc13260499)

[Figura 8 – Arquitetura do Chatbot desenvolvido 39](#_Toc13260500)

[Figura 9 – Diagrama do processo do diálogo 42](#_Toc13260501)

[Figura 10 – Estrutura básica do nodo de diálogo 51](#_Toc13260502)

[Figura 11 – Nodo da intenção #ajuda\_duplicar\_nota 51](#_Toc13260503)

[Figura 12 – Nodos da intenção #ajuda\_inutilizar\_nota 52](#_Toc13260504)

[Figura 13 – Validação dos nodos na árvore de diálogo 54](#_Toc13260505)

[Figura 14 – Estrutura de diálogo do nodo da intenção #ajuda\_cadastro. 57](#_Toc13260506)

[Figura 15 – Respostas do nodo #geral\_saudação. 59](#_Toc13260507)

[Figura 16 – Resposta 1 do *chatbot* a saudação do usuário. 59](#_Toc13260508)

[Figura 17 – Resposta 2 do *chatbot* a saudação do usuário. 60](#_Toc13260509)

[Figura 18 – Fluxo de diálogo da intenção #ajuda\_emitir\_nota. 61](#_Toc13260510)

[Figura 19 – Interface do chatbot. 64](#_Toc13260511)

[Figura 20 – Retorno do Watson *Assistant* ao orquestrador. 67](#_Toc13260512)

[Figura 21 – Inicialização do contexto no nodo Bem Vindo. 69](#_Toc13260513)

[Figura 22 – Alteração da variável de contexto ação. 69](#_Toc13260514)

[Figura 23 – Contexto do diálogo ao pedir um atendente real. 70](#_Toc13260515)

[Figura 24 – Resposta do *chatbot* com utilizando o *Discovery*. 73](#_Toc13260516)

[Figura 25 – Retorno da API do *Discovery*. 74](#_Toc13260517)

[Figura 26 – Tela de criação da intenção no Watson *Assistant*. 75](#_Toc13260518)

[Figura 27 – Arquivo em CSV com a intenção e os exemplos 76](#_Toc13260519)

[Figura 28 – Ferramenta de *Try it out* (Experimente) do *Assistant*. 77](#_Toc13260520)

[Figura 29 – Ensinando intenção com a ferramenta de *Try it out* (Experimente). 78](#_Toc13260521)

[Figura 30 – Ferramenta de *Analytics* do Watson *Assistant*. 79](#_Toc13260522)

[Figura 31 – Treinamento através do *Analytics*. 80](#_Toc13260523)

[Figura 32 – Diagrama do fluxo de melhoria do *chatbot*. 83](#_Toc13260524)

[Figura 33 – Pergunta não encontrada no *Discovery*. 89](#_Toc13260525)

[Figura 34 – Arquivo em *html* de perguntas e respostas. 90](#_Toc13260526)

[Figura 35 – Gráfico da questão 3 do formulário de testes do chatbot. 91](#_Toc13260527)

[Figura 36 – Notas atribuídas ao chatbot no formulário de testes. 92](#_Toc13260528)

[Figura 37 – Resultados do formulário de avaliação. 93](#_Toc13260529)

Lista de Tabelas

[Tabela 1 – Amostra de perguntas mapeadas. 38](#_Toc9783259)

[Tabela 2 – Exemplos fornecidos para treinamento de uma intenção. 39](#_Toc9783260)

[Tabela 3 – Intenções de ajuda mapeadas. 40](#_Toc9783261)

[Tabela 4 – Entidades do *Chatbot*. 41](#_Toc9783262)

[Tabela 5 – Intenções gerais e de controle mapeadas. 42](#_Toc9783263)

[Tabela 6 – Conteúdos ausentes no *Chatbot* 80](#_Toc9783264)

Lista de Abreviaturas e Siglas

|  |  |
| --- | --- |
| AIML | Artificial Intelligence Markup Language |
| API  BI  CPU  DANFE | Application Programming Interface  Business Intelligence  Unidade central de processamento  Documento Auxiliar da Nota Fiscal Eletrônica |
| ERP  IA  QA  LAT | Enterprise Resource Planning  Inteligência Artificial  Question Answering  Lexical answer type |
| LTDA  MIT  ML | Empresa de Sociedade Limitada  Massachusetts Institute of Technology  Machine Learning |
| NF-E | Nota Fiscal Eletrônica |
| NFC-E  NLU  NPM | Nota Fiscal de Consumidor Eletrônica  Natural Language Understanding  Node Package Manager |
| PIQUANT | Practical Intelligent Question Answering Technology |
| SAAS  SDK  SEFAZ | Software as a Service  Software Development Kit  Secretaria de Estado da Fazenda |
| TCC  TREC  TV  UIMA | Terapia Cognitivo-Comportamental  Text Retrieval Conference  Televisão  Unstructured Information Management applications |

Sumário

[Introdução 11](#_Toc13259571)

[1 CHATBOTS 14](#_Toc13259572)

[1.1 TRABALHOS RELACIONADOS 15](#_Toc13259573)

[2 WATSON 18](#_Toc13259574)

[2.1 Framework DeepQA 20](#_Toc13259575)

[2.1.1 Question Analysis 25](#_Toc13259576)

[2.1.2 Hypothesis Generation 27](#_Toc13259577)

[2.1.3 Soft Filtering 29](#_Toc13259578)

[2.1.4 Hypothesis and Evidence Scoring 29](#_Toc13259579)

[2.1.5 Final Merging and Ranking 31](#_Toc13259580)

[2.2 Serviços disponíveis (APIs) 32](#_Toc13259581)

[2.2.1 Watson Assistant 33](#_Toc13259582)

[2.2.2 Watson Discovery 34](#_Toc13259583)

[2.2.3 Watson Tone Analyzer 35](#_Toc13259584)

[2.2.4 Watson Natural Language Understanding 36](#_Toc13259585)

[3 Chatbot Desenvolvido 38](#_Toc13259586)

[3.1 Mapeamento das intenções 43](#_Toc13259587)

[3.2 Criando o fluxo do diálogo 50](#_Toc13259588)

[3.3 Desenvolvimento do Software orquestrador 62](#_Toc13259589)

[3.4 Configuração do Discovery 71](#_Toc13259590)

[3.5 Treinamento do Chatbot 74](#_Toc13259591)

[4 Avaliação e resultados 82](#_Toc13259592)

[4.1 Problemas encontrados 84](#_Toc13259593)

[4.2 Resultados 91](#_Toc13259594)

[5 CONCLUSÃO 94](#_Toc13259595)

[Referências Bibliográficas 96](#_Toc13259596)

[APÊNDICE A – FORMULÁRIO DE TESTES DO CHATBOT 99](#_Toc13259597)

[APÊNDICE B – FORMULÁRIO DE AVALIAÇÃO DO CHATBOT 100](#_Toc13259598)

# Introdução

A empresa Safetech Informática LTDA, é uma empresa de software que possui como seu principal produto um software de ERP (Sistema de Gestão Empresarial) para empresas do ramo calçadista. Comercializa também um software de varejo para gestão de redes de lojas de qualquer área de atuação e também oferece o serviço de *outsorcing* com mão de obra especializada. Outro produto da Safetech é o software de emissão de NF-e (Nota fiscal eletrônica) e NFC-e (Nota fiscal ao Consumidor Eletrônica) que é vendido na modalidade de SAAS (Software como Serviço), e que possui atualmente 396 clientes.

O sistema de NF-e e NFC-e é hospedado em um servidor na Safetech, não havendo assim a dependência de *hardware* por parte do contratante e é vendido para diversos clientes, em diversos segmentos de atuação. Para utilização do sistema, o cliente precisa acessar um endereço web com seu usuário e senha, e então dar início a emissão de notas. O modelo de negócio atual desse *software* é a venda de um pacote de notas para o cliente emitir no sistema, funcionando de maneira semelhante a um sistema pré-pago onde o cliente carrega a sua quantidade de notas e pode emitir as mesmas quando for necessário, não existindo mensalidade para utilização desse *software*.

Atualmente, é cobrada uma taxa para treinamento dos usuários desse sistema, onde o cliente contrata o serviço quando necessita de um treinamento para utilizar a ferramenta. O sistema é de fácil acesso e possui um modelo de negócio atrativo para clientes que não emitem muitas notas, por ter um custo sob demanda. Os clientes possuem diversos níveis de instrução na área de informática, variando entre dificuldades básicas na utilização do sistema e aqueles que acabam esquecendo o treinamento por utilizarem o sistema com pouca frequência.

Outro problema da empresa é que atualmente o suporte desse produto é composto por somente uma pessoa, que realiza os treinamentos e presta suporte aos clientes que entram em contato com dúvidas. Com o crescente aumento no número de clientes que utilizam a plataforma, aumenta o número de treinamentos e de atendimentos necessários, deixando essa pessoa cada vez mais sobrecarregada. Com isso, se faz necessário melhorar esse serviço da empresa, tanto em quantidade de pessoas, quanto em qualidade do atendimento.

Esses problemas poderiam ser sanados ou minimizados com um *chatbot* que ocupasse o papel de um suporte nível 1 (N1) no sistema, sendo a primeira camada de atendimento aos clientes que tiverem dúvidas sobre a utilização do serviço. Esse *chatbot* poderia ser construído e integrado ao sistema para oferecer um suporte as dúvidas básicas e mais frequentes dos clientes da plataforma, reduzindo a necessidade de intervenção humana nos atendimentos mais simples ou triviais, melhorando a qualidade e agilidade do processo.

De acordo com Souza e Moraes (2015) “*Chatbots* (ou *chatterbots*) são agentes conversacionais que têm como objetivo a comunicação, em linguagem natural, com usuários a fim de ajudá-los de alguma forma”. Esses agentes são criados com o objetivo de simular uma conversa com o usuário a ponto de o mesmo acreditar estar falando com uma pessoa. Conforme Maeda e Moraes (2017), essa capacidade de manter um diálogo com uma pessoa em linguagem natural está tornando os *chatbots* muito comuns e cada vez mais utilizados pelo mercado, a exemplo de empresas como Apple, Microsoft e Amazon, que possuem respectivamente a Siri[[1]](#footnote-2), Cortana[[2]](#footnote-3) e Alexa[[3]](#footnote-4), e que cada vez mais lançam produtos nessa direção.

Existem atualmente diversos *frameworks* para criação de *chatbots* disponíveis no mercado, sendo possível até criar *chatbots* sem a necessidade de desenvolvimento de software. Um exemplo disso é o BLIP (2018), que possui uma plataforma web para criação de *chatbots*, aonde o usuário configura todo o fluxo da conversa e pode publicar esse *chatbot* automaticamente em diversos serviços. Esse *chatbot* pode ser criado com um fluxo definido do que ele pode responder, ou pode ser integrado a serviços de processamento de linguagem natural, como o Watson da IBM ou o LUIS[[4]](#footnote-5) da Microsoft. Com essa integração é possível criar *chatbots* mais funcionais e que atendam melhor aos usuários.

Outro serviço disponível para criação de chatbots são as APIs do Watson da IBM (2018), que estão disponíveis para desenvolvedores com uma camada gratuita de utilização. Com as APIs do Watson, é possível criar *chatbots* utilizando serviços de Processamento de Linguagem Natural que a plataforma oferece. Sendo assim, é possível definir fluxos para as conversas que o agente poderá seguir e, utilizar o Processamento de Linguagem Natural para extrair as intenções do usuário das conversas, guiando as ações do *chatbot*. Essas APIs possuem suporte ao idioma português e podem ser integradas a diversas linguagens de programação, conforme Watson Assistant (2018). Por causa disso, foi o serviço escolhido para criação do chatbot desse trabalho.

Diante disso, esta proposta apresenta o desenvolvimento de um *chatbot* integrado ao sistema de emissão de NF-e e NFC-e da empresa Safetech informática, utilizando os serviços de PLN do IBM Watson, para que o mesmo seja capaz de realizar um atendimento de suporte N1 a esses clientes, com o objetivo de reduzir os problemas descritos acima e melhorar a qualidade e agilidade do atendimento.

Este trabalho está dividido em cinco capítulos. No primeiro capítulo são abordados os principais conceitos sobre os *chatbots* e as teorias relacionadas a este tipo de sistema, assim como é realizada uma análise dos trabalhos relacionados ao tema, com diferentes implementações de *chatbots*. No capítulo seguinte é descrita toda a pesquisa feita sobre o Watson da IBM, com a sua origem e características, para posteriormente serem apresentadas as soluções oferecidas atualmente pelo *framework*. No terceiro capítulo, é descrito todo o processo para criação do *chatbot*, desde a estruturação dos diálogos até o treinamento do atendente. No capítulo seguinte, são apresentados os resultados obtidos através dos testes de conversação e dos formulários utilizados. Por fim, no capítulo cinco, é realizada uma breve conclusão, mostrando as vantagens e limitações da solução proposta.

# CHATBOTS

Segundo Huang, Zhou e Yang (2007) um *chatbot* é um agente conversacional que interage com os usuários, utilizando sentenças em linguagem natural, sobre algum determinado domínio ou assunto. De acordo com Dale (2016) o conceito básico de um *chatbot* é sempre o mesmo, atingir algum resultado através de uma conversa dialógica com uma máquina, utilizando linguagem natural. Assim também, para Almeida JR (2017) os chatbots simulam uma conversa inteligente com o usuário, com o objetivo de mantê-la de uma forma natural e coerente.

O estudo e aplicação de chatbots são uma questão que iniciou em 1950, quando Alan Turing publicou o artigo “*Computing Machinery and Intelligence*”, na área da inteligência artificial, propondo que um computador poderia ser considerado inteligente se ele pudesse manter um diálogo com uma pessoa através de um chat, sem que ela soubesse se estava falando com uma pessoa ou com um computador. Criando assim o Teste de Turing, que procura avaliar se um computador se passa por uma pessoa.

Após isso, conforme Araujo (2010) “A primeira geração de chatbots surgiu com o ELIZA, criado em 1966 pelo cientista da computação do MIT Joseph Weizenbaum para simular o diálogo com um analista rogeriano”. Primo e Coelho (2002) desenvolveram o primeiro *chatbot* criado para a língua portuguesa, a Cybelle. O sistema simula um diálogo com os usuários utilizando uma estrutura semelhante a do *chatbot* ELIZA.

Esses chatbots são classificados por Maeda e Moraes (2017) em dois principais tipos: agentes com arquitetura cognitiva e agentes com arquitetura reativa. Os agentes com arquitetura cognitiva são baseados em uma meta, atendendo as tarefas que o usuário solicita em linguagem natural. Um exemplo desses chatbots são os assistentes pessoais de celular que existem atualmente. Já na arquitetura reativa os chatbots imitam uma conversa real de humanos, sem procurar por uma tarefa em específico para ser realizada, sendo projetados para domínio aberto.

De acordo com Bernardi, Sônego e Pozzebon (2018), que analisaram as publicações cientificas sobre *chatbots*, o tema é relativamente recente, quando comparado com outros assuntos, mas é um tema muito relevante devido a sua variedade de aplicações e interdisciplinaridade, existindo um avanço nas pesquisas tanto nas áreas acadêmicas, quando em produtos voltados para o público geral.

## TRABALHOS RELACIONADOS

Pesquisas e trabalhos sobre chatbots para atendimento e respostas a perguntas de usuário são comuns e possuem diversas abordagens. Em Souza e Moraes (2015) foi criado um *chatbot* para responder perguntas dos usuários relacionadas a área de saúde, mais especificamente sobre doenças. Para esse *chatbot*, foi criada uma base em AIML, que conforme Marieto et al (2013) é uma linguagem de marcação para inteligência artificial, com as perguntas sendo extraídas de uma *Frequently Asked Questions* (FAQ) de um site no idioma português, criando respostas padrões para diferentes entradas dos usuários. O trabalho de Bada e Menezes (2012), também envolve a construção de uma base AIML utilizando corpus em português, para esclarecer as dúvidas de alunos dos cursos de educação a distância, porém adicionando mais agentes ao chatbot, para procurar as respostas para o usuário utilizando proposições lógicas da ontologia, quando não encontrar na base.

Chatbots de domínio aberto, no idioma português, foram propostos por Maeda e Moraes (2017), utilizando uma base de legenda de filmes e uma base de mensagens de celular, como corpus para treinamento do chatbot. Nesse trabalho foi utilizada uma rede neural *sequence to sequence* de Sutskever et al. 2014 (apud Maeda e Moraes 2017),para definir a base de conhecimento do agente. Para avaliação do *chatbot*, os autores criaram uma interface web, para os usuários se comunicarem com o mesmo, após o diálogo foram aplicados dois questionários, um para analisar o perfil do usuário e outro com foco em validar a coerência semântica.

Com o corpus de legendas de filmes, os autores observaram pouca coerência e falta de continuidade nas conversas, atribuindo esse resultado a heurística utilizada para definir os turnos das conversas, que foi necessário, pois no corpus de legendas os turnos não são identificados. Para validar essa atribuição, eles utilizaram o corpus de mensagens de celular, utilizando o mesmo modelo de configuração do corpus anterior, obtendo respostas mais coerentes do *chatbot*. Nesse trabalho os autores relataram dificuldades em encontrar corpora de diálogos em português para treinarem o *chatbot*, e concluíram que o resultado obtido com o corpus de mensagens de celular se assemelha ao obtido por Sutskever et al. 2014 (apud Maeda e Moraes 2017), no trabalho do idioma em inglês, porém com o corpus de legendas não conseguiram os mesmos resultados.

O trabalho de Souza (2018), aborda um *chatbot* para atendimento de clientes de um *e-business* com o mesmo objetivo principal deste trabalho, melhorar o atendimento aos clientes, através de um atendimento automático. Para isso, o autor criou uma estrutura semelhante a uma FAQ, mapeando perguntas e respostas pré-definidas, definindo assim os fluxos de conversação.

Para desenvolvimento foi utilizado o *framework* *DialogFlow[[5]](#footnote-6),* que funciona de maneira semelhante ao Watson da IBM, com o mapeamento das intenções do usuário e as entidades da intenção. Para avaliação, o autor também utilizou um questionário, porém aplicado de uma maneira diferente, no final da conversa o *chatbot* questiona o usuário como foi a conversa, com três perguntas dicotômicas e sequenciais, a fim de avaliar a precisão das respostas. O autor conclui que a arquitetura definida para a criação do chatbot cumpriu o propósito, satisfazendo as dúvidas de mais da metade dos usuários, com um comportamento semelhante ao de um humano. Sugere que a base poderia ser ampliada e as respostas para os usuários deveriam ser simplificadas.

Utilizando o Watson da IBM, existe o trabalho de Oliveira (2018) que consiste na criação de um *chatbot* para atuar como professor virtual da disciplina de engenharia de software. Para a criação, o autor utilizou o *framework* do Watson *Assistant* da IBM para mapear as intenções do usuário e as entidades de cada intenção, mapeando um total de 5 intenções, cada uma sendo uma pergunta específica da área. Uma característica implementada pelo autor, foi a de que o usuário consegue personalizar o tipo de resposta que prefere receber (curta ou longa) e o seu nível de dificuldade com a disciplina, para ser possível configurar respostas diferentes de acordo com as preferências do usuário. Outra abordagem foi a de armazenar as perguntas que possuíam um percentual de confiança menor de 80% em um banco de dados para posterior análise e treinamento do chatbot para responder a elas. Foram realizados testes durante a implementação do chatbot, porém o autor não submeteu o *chatbot* a conversas com outros usuários, não sendo possível averiguar se o mesmo traria reais benefícios.

Outro trabalho relacionado é o de Almeida JR (2017), que desenvolveu um *chatbot* para conversar com adolescentes de 15 a 18 anos, que possuem depressão ou que desejam se informar para evitá-la. Foi utilizada a linguagem *ChatScript* para criação do *chatbot* e os diálogos foram modelados para atender a objetivos específicos, com embasamento nas técnicas de TCC (Terapia Cognitivo-Comportamental). O *chatbot* utiliza turnos específicos de conversa, existindo uma ação dele em cada turno, assim, o diálogo é conduzido pelo *chatbot* e o usuário não consegue interagir com o ele enquanto não for o seu turno. Antes da interpretação da mensagem do usuário, foram realizados os processos de normalização do texto, correção ortográfica, *stemming* e *POS-tagging*, utilizando a biblioteca NLPNET em python. Para a avaliação, o autor aplicou testes de conversão e um questionário com 32 adolescentes de duas escolas públicas. Analisando as avaliações, o autor concluiu que o *chatbot* obteve resultados satisfatórios, com 85,94% dos adolescentes tendo ficado satisfeitos em conversar com o *chatbot* e 92,19% aprovando o mesmo.

Diante dos trabalhos descritos acima, é possível observar que existem trabalhos sobre *chatbots* para atendimentos de clientes e para auxilio de usuários nas mais diversas tarefas e áreas. O trabalho de Souza (2018) que trata de um *chatbot* para atendimentos de clientes de um *e-bussines* é semelhante ao proposto neste trabalho, porém com a utilização de outro *framework*, não com o Watson da IBM. O trabalho de Oliveira (2018), por sua vez, utiliza o Watson da IBM, porém foram desenvolvidas poucas estruturas de diálogo (5) e o *chatbot* não foi avaliado por nenhum usuário a fim de atestar a sua real contribuição. O *chatbot* desenvolvido por Almeida JR (2017), auxilia os usuários, porém o mesmo conduz o diálogo perante o usuário, não atuando como um atendente que responde a perguntas.

Todos os trabalhos tratam de alguma forma de automatização do atendimento de usuários e na melhoria de algum aspecto do atendimento com a inclusão dos *chatbots*. Mesmos semelhantes nenhum dos trabalhos encontrados aborda a resolução do problema de atendimento a suporte nível 1 de um sistema com um *chatbot* utilizando o Watson da IBM, solução proposta por esse trabalho.

# WATSON

De acordo com High (2012) o IBM Watson é um sistema cognitivo, com processamento de linguagem natural profundo, que é único devido a combinação das capacidades de PLN, para entender dados que não estão estruturados, geração de hipóteses de resposta e avaliação das mesmas. Aprendizado dinâmico, com base nos resultados, para melhorar as respostas a cada iteração e interação. Nenhuma destas capacidade é exclusiva do Watson, porém a união das mesmas o torna uma solução poderosa.

Um dos motivos da equipe da IBM ter assumido este desafio de pesquisa, segundo Ferrucci *et al* (2010), foi por eles acreditarem que avanços na área de QA (*Question Answering*), podem ajudar a fornecer informações de suporte para tomada de decisão nas mais diversas áreas, como medicina, BI (*Business* *Intelligence*), segurança e suporte ao cliente. Existindo assim uma clara relevância comercial neste projeto para a IBM.

Em 2007 uma equipe de cerca de 20 pesquisadores da IBM, assumiu como desafio a criação de um sistema de computador que fosse capaz de competir em tempo real, no programa de TV americano *Jeopardy,* contra concorrentes humanos e com um nível de acerto nas respostas igual ou maior à média dos participantes que já haviam vencido no programa (FERRUCCI *et al*, 2010). Não sendo somente um teste de laboratório, mas sim, adicionar um participante real no programa que fosse um sistema de computador.

O *Jeopardy* é um programa de TV americano, que está no ar a mais de 25 anos nos Estados Unidos e que coloca três concorrentes humanos um contra o outro, em uma competição que requer respostas rápidas a perguntas ricas em linguagem natural. As perguntas são chamadas de pistas, com um domínio amplo de tópicos e com penalidades para respostas erradas (FERRUCCI *et al*, 2010). Ainda de acordo com Ferrucci *et al.* (2013), essas perguntas geralmente contêm uma linguagem complexa, ambiguidades e trocadilhos. Para todas as questões os competidores disputam a chance de responder através de uma campainha portátil que fica em sua mão. Quem pressiona o botão primeiro tem a oportunidade de responder.

Devido a isso, o *Jeopardy* é um jogo amplamente reconhecido como um jogo divertido e que requer jogadores inteligentes, conhecedores e rápidos. Todos os participantes devem passar por um teste com 50 questões para poderem participar do jogo. O jogo se constitui de um painel, conforme a Figura 1, que é uma grade organizada em seis colunas, cada coluna sendo uma categoria e cinco linhas com valores crescentes em dólares. No início do jogo todas as pistas estão escondidas e o participante escolhe de qual categoria e valor deseja ver a pista. Após a escolha, a pista é revelada para todos os participantes e o apresentador a lê em voz alta, quando a leitura termina uma luz vermelha se acende e é possível pressionar o botão para ter uma chance de responder. (FERRUCCI *et al.*, 2010).

Figura 1 – Quadro do jogo Jeopardy



Fonte: (JEOPARDY!, 2018).

Ao acertar a pergunta para a pista, o valor do quadro é somando ao valor total do participante e ao responder incorretamente, o valor é descontado. O participante fica impossibilitado de pressionar o botão por meio segundo caso aperte antes da luz acender. (FERRUCCI *et al.*, 2010). O que diferencia esse programa dos tradicionais programas de perguntas e respostas é que as pistas são apresentadas como respostas e o competidor deve formular a pergunta correta para a resposta exibida.

Conforme Ferrucci *et al.* (2013), os principais desafios de implementar um sistema para participar do *Jeopardy* são as perguntas de amplo domínio, a alta taxa de precisão e confiança, e a velocidade em que as respostas devem ser dadas. Sendo esses três fatores de importância crítica para o sucesso no jogo. Assim, de acordo com Ferrucci *et al.* (2010), para um computador ser capaz de competir com humanos em tempo real, o sistema deve ser capaz de produzir respostas exatas para as questões de linguagem natural, com alta precisão e velocidade. Respondendo em torno de 70% das questões, com uma precisão aproximada de 80% em 3 segundos ou menos. Desta forma, a confiança é utilizada para saber o momento em que o botão deve ser apertado, sendo que a mesma precisa ser calculada durante a leitura da pista para a questão.

Então, segundo Ferrucci *et al.* (2010), após aproximadamente três anos de esforços da equipe composta com cerca de 20 pesquisadores e engenheiros de software, o framework *DeepQA*, que será detalhado no capítulo 4.1, responsável pelo funcionamento do Watson, passou a obter resultados no mesmo nível dos vencedores do jogo. Conforme Ferrucci *et al.* (2013) o Watson disputou vários jogos contra campeões do jogo, inclusive em um programa televisionado em fevereiro de 2011, em que ganhou dos dois maiores jogadores de todos os tempos. Para Ferrucci (2012), o fato de um computador vencer os melhores competidores humanos no jogo representa um grande avanço no campo de QA de amplo domínio.

Vale ressaltar, de acordo com Das (2014), que o sistema precisa ser autônomo e sem conexão com a internet durante o jogo, para poder competir com concorrentes humanos. Ele precisa analisar a questão e procurar pela resposta nos recursos disponíveis, sendo capaz de responder. Tudo isso em questão de segundos, para ser possível apertar o botão antes dos outros competidores. E como explica Ferrucci *et al.* (2013), o framework do Watson não mapeia a questão para um banco de dados de questões e faz uma simples *query* para encontrar a resposta, mas sim, analisa a linguagem natural existente tanto na questão quanto nas suas bases de conhecimento.

## Framework DeepQA

A fim de criar um ponto de referência inicial de performance, chamado de *baseline*, os autores utilizaram o sistema PIQUANT (Tecnologia de Atendimento inteligente a perguntas rápidas), como cita Ferrucci et al. (2010) (apud Prager, Chu-Carroll, Czuba 2004), também em desenvolvimento pela IBM, mas com início antes do desafio do Jeopardy. Segundo o autor, na época, um dos top três a cinco sistemas de QA na conferência de recuperação de texto (TREC[[6]](#footnote-7)). De acordo com Voorhees e Dang (2005), a conferência TREC foi criada em 1999 com o objetivo de fomentar a pesquisa sobre sistemas que retornam uma resposta e não somente documentos contendo as respostas.

Utilizando uma base de 500 pistas randômicas de episódios passados em um período de 15 anos, os pesquisadores criaram um *baseline* utilizando o PIQUANT. O resultado foi abaixo do esperado para o desafio do Jeopardy com as respostas mais confiantes obtendo um percentual de acerto de 47% e utilizando todas as pistas do *dataset* obteve uma precisão de 13%. Com este *baseline* os autores obtiveram o estado da arte em sistemas de QA para a tarefa do Jeopardy, até o então momento e um ponto de partida para desenvolvimento do novo framework. (FERRUCCI et al., 2010).

No início da criação do sistema, os autores citam as dificuldades que encontraram, como a de retirar do contexto os resultados de pesquisas publicados na área de QA e aplicar em sistemas de ponta a ponta para ser possível comparar resultados. Assim como as pesquisas que geralmente não são focadas nos problemas específicos da área, necessários para o projeto. Com isso, observaram que a área de QA se beneficiaria da evolução colaborativa de uma única arquitetura extensível que permitiria avaliar o resultado dos componentes em um contexto técnico comum. (FERRUCCI et al., 2010).

Assim sendo, foi construído o sistema chamado de DeepQA, que segundo Ferrucci et al. (2010), é uma arquitetura baseada em evidências probabilísticas massivamente paralelas. Para o desafio Jeopardy, são utilizadas mais de 100 técnicas diferentes para analisar a linguagem natural, identificando fontes, encontrando e gerando hipóteses, encontrando e pontuando evidências, e mesclando e classificando hipóteses. De acordo com os autores, muito mais importante do que qualquer técnica em particular que foi utilizada, foi o fato de elas terem sido combinadas no DeepQA, de um modo em que as abordagens de sobreposição trazem suas forças para suportar e contribuir para melhorias na precisão, confiança ou velocidade.

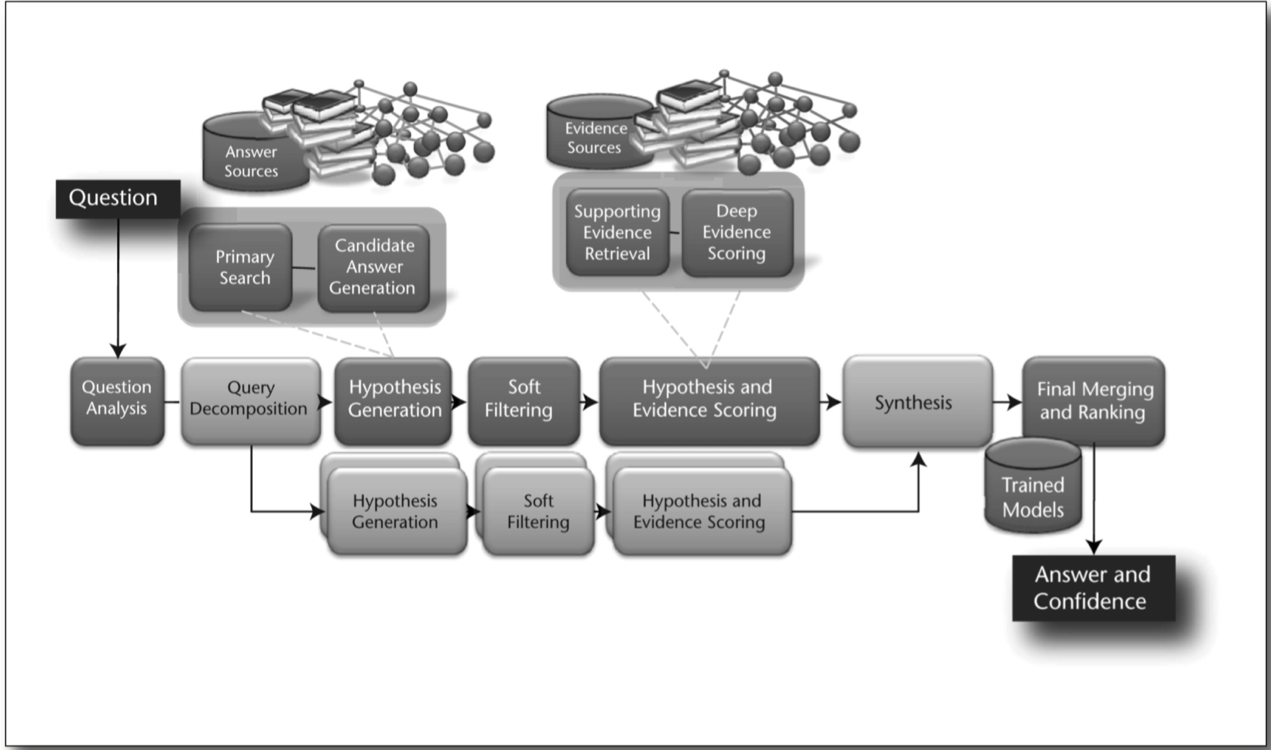
Para Gliozzo et al. (2013), o DeepQA é uma arquitetura de software para analisar conteúdo em linguagem natural em bases de questões e de conhecimento. Descobrindo e avaliando potenciais respostas, reunindo e classificando evidências que comprovem as mesmas em fontes não estruturadas, como documentos de linguagem natural, e estruturadas como em bancos de dados relacionais e bases de conhecimento. Segundo o autor, os componentes do Watson incluem uma grande variedade de soluções com o estado da arte nos campos de PLN, ML, recuperação de informações, web semântica e computação na nuvem.

Estimar a confiança da resposta para a pista foi um ponto crítico para criação do DeepQA e como ele seria moldado. Conforme Ferrucci et al. (2010), não haviam expectativas de que algum componente no sistema fizesse todo o trabalho. Todos os recursos do sistema registram a confiança e algoritmos de *machine learning* combinam esses recursos para decidir se há confiança na resposta final ou não. De acordo com Ferruci et at. (2013), o DeepQA possui uma ampla gama de componentes de busca e pontuação conectáveis que permitem a integração de muitas técnicas analíticas diferentes. Com isso, o ML é utilizado para combinar pontuações de diferentes pontuadores e cada resposta está ligada com a sua evidência de apoio.

Conforme Ferrucci et al. (2010) a primeira etapa para resolver um problema de QA é a aquisição de conteúdo, ou a identificação e busca de conteúdo que será utilizado para fontes de respostas e evidências. Devido ao amplo domínio do Jeopardy, as fontes do Watson possuem uma ampla variedade de enciclopédias, dicionários, artigos de notícia e assim por diante. O DeepQA aplica um processo automático para ampliação do seu corpus, com quatro etapas de alto nível. Primeiro identifica documentos iniciais e recupera documentos relacionados na internet. Após isso, extrai dados brutos de texto contido nos documentos web relacionados. Então os dados brutos são pontuados conforme relação ao documento inicial e por último o corpus é expandido com os dados mais relevantes. O corpus expandido é o utilizado pelo sistema durante o jogo. O sistema também coleta dados de fontes semiestruturadas e estruturadas, como banco de dados, taxonomias e ontologias.

A arquitetura em um alto nível do sistema DeepQA está representada na Figura 2, conforme apresentada por Ferruci et. al. (2010). Com todos os componentes utilizados no sistema e o fluxo que uma pergunta segue até ser obtida a resposta para a mesma.

Figura 2 – Alto nível da arquitetura DeepQA

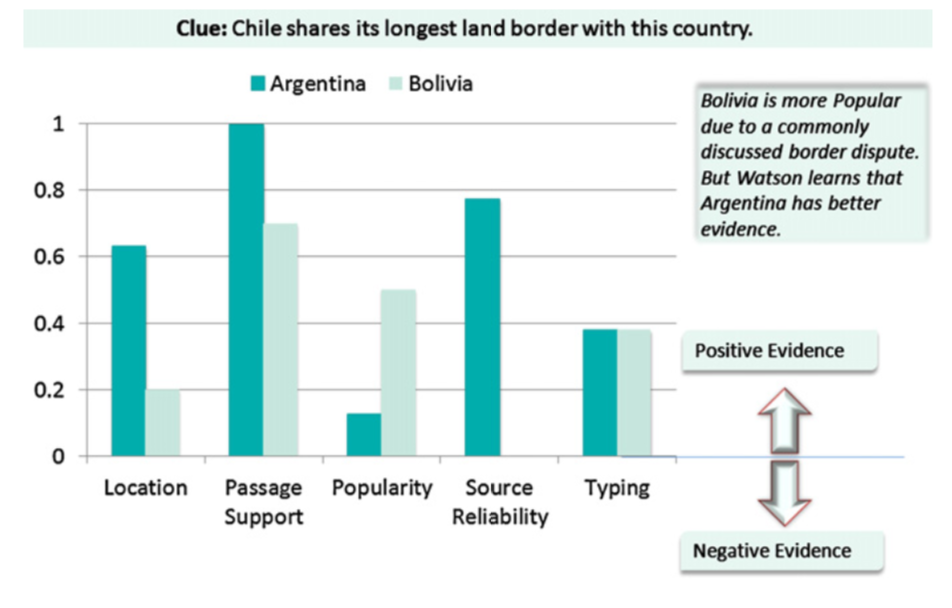


Fonte: (FERRUCCI et al. 2010).

Conforme Ferrucci et al. 2013, o DeepQA analisa uma questão recebida para determinar com precisão o que está sendo pedido e gera diversas possíveis respostas por meio de uma ampla pesquisa em grandes volumes de conteúdo. Para cada uma dessas respostas, uma hipótese é formada, considerando a possível resposta no contexto da questão e do tópico original. Para cada hipótese, o DeepQA gera um encadeamento independente que tenta provar isso. Posteriormente, é realizada uma busca nas fontes de conteúdo, em busca de evidências que suportem ou refutem cada hipótese. Para cada conjunto de evidência-hipótese, o DeepQA aplica centenas de algoritmos, que dissecam e analisam as evidências ao longo de diferentes dimensões de evidências, como classificação de tipo, tempo, geografia, popularidade, suporte de passagem, confiabilidade de origem e parentesco semântico.

Essa análise produz centenas de recursos. Esses recursos são então combinados com base em seu potencial para prever a resposta correta. O resultado final desse processo é uma lista ordenada de possíveis respostas, cada uma com uma pontuação de confiança, indicando o grau em que a resposta é considerada correta, junto com os links para a evidência. A Figura 3 mostra as dimensões de evidências para a pista “O Chile divide sua maior fronteira terrestre com este país” do Jeopardy. Cada algoritmo usa diferentes recursos e técnicas algorítmicas, cada uma com diferentes compensações de precisão e recuperação. Então, para formar um conjunto consumível de dimensões de evidência, os autores agruparam as características de acordo com a taxonomia dos tipos de evidências definidos por eles (por exemplo, localização e popularidade, como mostrado na Figura 3). (FERRUCCI et al., 2013).

Figura 3 – Dimensões de evidência para uma pista do *Jeopardy*.



Fonte: (FERRUCCI et al. 2013).

Estes recursos são combinados e recebem um peso ​​de acordo com o modelo de ML treinado, a fim de avaliar e exibir a contribuição de cada tipo de evidência na produção do índice final de confiança. A Figura 3 ilustra um perfil de evidência comparativa, destacando algumas das dimensões definidas para o Watson. Os perfis de evidências foram usados ​​pelos desenvolvedores do projeto para depuração, e segundo os autores, serão úteis para os usuários finais em muitos aplicativos, para entender e explorar evidências associadas a uma possível resposta. (FERRUCCI et al., 2013).

O DeepQA, de acordo com Ferrucci et al. (2010), foi desenvolvido utilizando o framework Apache UIMA[[7]](#footnote-8). Ferrucci e Lally (2004), caracterizam o UIMA como um sistema que analisa grandes volumes de informações estruturadas e não estruturadas, para descobrir, organizar e apresentar informações relevantes para o usuário final. Com isso, todos os componentes do DeepQA são implementados como anotadores do UIMA, que são componentes de software que analisam o texto e produzem anotações ou afirmações sobre o mesmo. (FERRUCCI et al., 2010).

A performance inicial do Watson, conforme Ferrucci et al. (2010), foi de duas horas para responder a uma única pergunta, utilizando um único processador. Isso fez com que a equipe precisasse expandir o sistema para utilizar processamento assíncrono no UIMA, redimensionando o processamento do Watson em mais de 2500 núcleos. Essa implementação assíncrona permitiu latências de tempo de execução com intervalos de três a cinco segundos, tornando o sistema competitivo. Ferrucci et al. (2010) cita também a utilização do Hadoop[[8]](#footnote-9), mapeando os anotadores UIMA no *framework* de *map-reduce*, para pré-processar o corpus e criar índices de execução mais rápidos. Segundo os autores, o *framework* do Hadoop oferece alta utilização de CPU (Unidade Central de Processamento) e fornece ferramentas para implementar, gerenciar e monitoras processos.

Nas próximas seções serão detalhados os componentes da arquitetura de alto nível do Watson, presentes na Figura 2. A questão a ser respondida, passa por um fluxo ao ser recebida que consiste principalmente em *Question Analysis* (Análise da questão), *Hypothesis Generation* (Geração de hipóteses), *Soft Filtering (*Filtragem Suave), *Hypothesis and Evidence Scoring (*Pontuação de evidências e de hipóteses) e *Final Merging and Ranking (*Agrupamento final e raqueamento). Após essas etapas macro, a resposta e a sua pontuação de confiança são apresentadas pelo sistema.

### Question Analysis

Segundo Ferrucci et. al. (2010) a primeira etapa do processo do sistema DeepQA consiste em analisar a questão. Neste processo, o sistema busca entender o que a questão está perguntando e executa as análises iniciais para determinar como a questão será processada nas próximas etapas do fluxo. O autor afirma que a abordagem do sistema DeepQA estimula a mistura de técnicas especialistas em análises de questões, como *parses* superficiais, análises profundas apud (McCord 1990), formas lógicas, rótulos de papéis semânticos, núcleos de referência, relações, entidades nomeadas, entre outros.

Nesta análise da questão ocorre a classificação da pergunta, para identificar perguntas ou partes da pergunta que precisam de processamento especial. Podendo ocorrer para questões inteiras que possuem certas funcionalidades sintáticas, semânticas ou retóricas, ou para palavras únicas com duplo sentido. Ferrucci et. al. (2010) colocam que a classificação de perguntas pode identificar o tipo da questão, como uma questão de quebra-cabeça, matemática, de definição, entre outras. Identificando trocadilhos, restrições, componentes de definição, ou sub pistas inteiras dentro de uma questão (pista) principal.

Para Ferrucci et. al. (2010) existe uma classificação chamada de LAT (*Lexical Answer Type*), que consiste em uma palavra na pista que indica o tipo da resposta sem qualquer tentativa de entender a sua semântica. Exemplo na pista “Inventado nos anos 1500 para acelerar o jogo, esta manobra envolve duas peças da mesma cor” (tradução nossa), da categoria xadrez, a LAT é a palavra “manobra”. Dito isso, outra etapa da análise é a determinação da LAT da pergunta e das respostas candidatas e, se a LAT da resposta candidata pode ser considerada uma instância da LAT da pergunta. Segundo o autor, determinar se uma resposta pode ou não ser considerada uma instância da LAT da pergunta é um tipo importante de pontuação de confiança e uma fonte comum de erros.

Outra heurística utilizada por Ferrucci et. al. (2010) foi a de determinar o foco da questão, que conforme explica o autor é a parte da pergunta que, se substituída pela resposta, transforma a pergunta em uma afirmação autônoma. Exemplo, o foco da pista “Quando atingido por elétrons, um fósforo emite energia eletromagnética nesta forma” (tradução nossa) é “nesta forma”. O autor afirma que o foco geralmente contém informações úteis sobre a resposta, sendo frequentemente o assunto ou objeto de uma relação na pista. Sendo assim, o foco é uma maneira útil de coletar evidências sobre uma resposta candidata, pois uma questão pode ser transformada em uma declaração verdadeira quando substitui o foco pela resposta candidata.

Ainda na análise da questão, existe também a detecção de relações. Ferrucci et. at. (2010) abordam que a maioria das questões contém relações, podendo ser predicados sintáticos de sujeito-verbo-objeto ou relações semânticas entre entidades. Um exemplo pode ser observado na pista “Eles são os dois estados que você poderia estar entrando novamente se cruzar a fronteira norte da Flórida” (tradução nossa), aonde é possível detectar a relação entre as fronteiras, apresentada da seguinte forma pelo autor, (Flórida, ?, norte). Com isso, o sistema do Watson pode utilizar as relações detectadas para consultar um armazenamento triplo e gerar candidatas as resposta diretamente.

Ferrucci et. al (2010) afirmam que atualmente a capacidade do Watson de utilizar bancos de dados curados para consultar diretamente as respostas é para menos de 2% das pistas, devido à amplitude de relações no domínio do *Jeopardy* e da variedade pelas quais elas são expressas. Com isso, a utilização de bancos de dados existentes está atrelada a capacidade de analisar a questão e detectar as relações cobertas por cada banco de dados. Conforme o autor, o benefício da detecção de relações está atrelado a frequência das relações, pois relações que ocorrem menos reduzem drasticamente a ajuda em encontrar a resposta. Para Ferrucci et. al. (2010) a detecção de relações em amplo domínio continua sendo uma grande área de pesquisa em aberto.

Outro requisito importante citado por Ferrucci et. al. (2010) é a decomposição das pistas, pois algumas questões são melhor respondidas após a decomposição. O DeepQA utiliza *deep parsing* baseado em regras e classificação estatística para identificar quando uma pista deve ser decomposta em sub pistas e qual a melhor maneiras de dividi-las. A hipótese apresentada por Ferrucci et al (2010) é de que a interpretação correta da questão e da(s) respostas(s) derivada(s) terão maior pontuação após a coleta de todas as evidências relacionadas e todos os algoritmos relevantes terem sido considerados. Para o autor, mesmo que a pista não precise ser decomposta para ser respondida, o método pode ajudar a aumentar a pontuação de confiança relacionada a resposta.

Após a decomposição das pistas, o sistema DeepQA as resolve com a aplicação recursiva das questões no sistema, do início ao fim, sintetizando as respostas finais com um componente de combinação de respostas personalizável. Esse processamento da decomposição é exibido na Figura 2 em “*Query Decomposition*”, com o caminho em cinza fraco exibindo o *loop* que as questões decompostas seguem. Os componentes de síntese personalizáveis para as respostas são utilizados, de acordo com o autor, para permitir que algoritmos de síntese especializados sejam incorporados facilmente ao sistema. (FERRUCCI et. al., 2010).

### Hypothesis Generation

A etapa de *Hypothesis Generation* (geração de hipóteses), como caracteriza Ferrucci et. al. (2010), consiste em transferir os resultados da etapa anterior (análise de perguntas) e produzir respostas candidatas a serem a correta, pesquisando nas fontes disponíveis do sistema. Neste processo, cada resposta candidata é conectada de volta ao texto original da pergunta e, a partir disto considerada uma hipótese, que o sistema tem que provar correta com algum grau de confiança. A pesquisa para geração de hipóteses é chamada de “*Primary Search*” (busca primária) pelos autores, para distingui-la da pesquisa por evidências, gerada posteriormente. Assim como em outras etapas do sistema, são utilizadas diferentes abordagens e algoritmos para a busca primária e geração das respostas candidatas.

A etapa de *Primary Search* (busca primária), segundo Ferrucci et. al. (2010), resume-se no processo de encontrar o máximo possível de conteúdo, que possa servir de suporte para a resposta, com base nos resultados da análise da questão. Assim, o foco está diretamente ligado a busca de conteúdo, com a expectativa de que as etapas de análise mais profunda de conteúdo possam extrair respostas candidatas e pontuá-las, junto com qualquer outra evidência que possa ser encontrada para suportar ou refutar a resposta candidata. Com o objetivo de obter o melhor equilíbrio entre precisão e uso de recursos computacionais, os autores realizaram testes empíricos para encontrar o número de resultados de pesquisa e geração de respostas candidatas que pudesse produzir os melhores resultados do sistema. Dessa forma, o sistema encontra a resposta correta como uma das candidatas para 85% das perguntas em algum lugar entre as 250 respostas candidatas com maior pontuação.

Dentre as diferentes técnicas de busca que são utilizadas por Ferrucci et. al. (2010) na busca primária, são citados os softwares de busca de texto Indri[[9]](#footnote-10) e Lucene[[10]](#footnote-11), geração de várias consultas diferentes para uma mesma pergunta, pesquisa de passagem e pesquisa de documentos. Também são realizadas pesquisas em bases de conhecimento de triplas utilizando *SPARQL*, que segundo Santarém (2014) é um conjunto de especificações que fornece linguagens e protocolos para consultar e manipular conteúdo estruturado disponível na web. As pesquisas em bases de dados de triplas, nesta etapa de busca primária, são baseadas em entidades nomeadas na pista. Por exemplo, encontrar todas as entidades no banco de dados relacionadas com as entidades que constam na pista. Para os casos em que uma relação semântica é detectada na pista, segundo o autor, a consulta é mais focada, sendo possível gerar a resposta candidata utilizando alguma instância conhecida.

Após a busca por conteúdo para suporte à resposta, o sistema, de acordo com Ferruci et. al. (2010), parte para a etapa de geração de respostas candidatas. Os dados que são obtidos na busca primária possuem diferentes formatos e, com isso, eles precisam receber diferentes tratamentos para ser possível obter como resultado uma resposta candidata apropriada. Segundo o autor, são aplicadas diferentes técnicas de acordo com o tipo de resultado. Em fontes de dados orientadas a título de documento, o título é utilizado como resposta candidata, sendo que o mesmo título pode gerar mais de uma resposta candidata, pois o sistema retira trechos do título para gerar variações da resposta. Quando o documento possui *hyperlinks* para outros documentos, o sistema realiza uma análise nos *links* para gerar respostas candidatas.

As pesquisas como consulta inversa ao dicionário e consulta em fontes de armazenamento triplo, tem como resultado direto uma resposta candidata, como afirma Ferruci et. al. (2010). Por outro lado, conforme o autor, os resultados da pesquisa de passagem, exigem uma análise mais profunda do texto, antes de identificar as respostas candidatas, como por exemplo, a aplicação de detecção de entidades nomeadas para extrair as respostas candidatas da pesquisa de passagem.

Ferruci et. al. (2010 alegam que se a resposta correta não for gerada nesta etapa, como uma das candidatas, o sistema não tem nenhuma expectativa de responder a pergunta. Com isso, o autor salienta que essa etapa do sistema privilegia a busca de respostas candidatas, em detrimento a precisão das mesmas. Pois o intuito é que o fluxo restante do sistema gere a resposta correta, mesmo com um conjunto grande de candidatos a resposta. Sendo assim, o autor argumenta que um dos objetivos do projeto do sistema DeepQA é tolerar o ruído de informações nos estágios iniciais do fluxo do sistema, aumentando a precisão nas etapas conseguintes. Ferucci et. al. (2010) afirmam que o Watson gera várias centenas de candidatas a resposta neste estágio.

### Soft Filtering

Como descrito por Ferrucci et. al. (2010) a etapa de *Soft Filtering (*filtragem suave, tradução nossa), consiste em equilibrar o uso dos recursos computacionais durante as etapas do framework. Nas etapas iniciais, em que existem mais respostas candidatas a serem a correta, são aplicados algoritmos que utilizam menos recursos computacionais para reduzir o número de respostas, em seguida, são aplicados os algoritmos de pontuação com maior custo computacional a uma número reduzido de candidatos, com foco maior na precisão. Ferrucci et. al. (2010) caracterizam essa etapa como fundamental para equilibrar os recursos computacionais e a precisão das respostas.

Logo após, o sistema DeepQA combina essas pontuações geradas pela etapa de filtragem suave em uma pontuação única para a etapa. O sistema adota um limite mínimo de pontuação nesta etapa para filtragem das respostas, as que obtém uma pontuação acima do limite, passam para a próxima etapa, a de *Hypothesis and Evidence Scoring* (pontuação de hipóteses e evidencias, tradução nossa). As que não passam, são direcionadas diretamente para a etapa de *Final Merging* (combinação final, tradução nossa). Conforme Ferrucci et. al. (2010), a filtragem suave é uma função parametrizável e atualmente o Watson permite que aproximadamente 100 candidatos passem pelo filtro. Ainda segundo o autor, o modelo de pontuação da filtragem suave, assim como o limite mínimo de pontuação, são determinados através de aprendizado de maquina nos dados de treinamento.

### Hypothesis and Evidence Scoring

Na sequência, as respostas candidatas que passam pelo limite imposto na filtragem suave são processadas novamente. Este processo, segundo Ferrucci et. al. (2010) é uma rigorosa avaliação que envolve a coleta de evidências adicionais para cada resposta hipotética, e a aplicação de várias pontuações para validar as evidências de suporte. Assim, esse processo consiste de duas etapas, coleta de evidências e pontuação. A coleta de evidência busca informações com o objetivo de coletar conteúdo que justifique a resposta e a pontuação fica responsável por determinar o grau em que a evidência suporta a resposta.

A etapa da recuperação de evidências de suporte adicionais também utiliza uma variedade de técnicas para coleta. De acordo com Ferrucci et. al. (2010) uma das técnicas mais eficazes é a pesquisa de passagem utilizando uma derivação da pesquisa primária, em que a resposta candidata é adicionada como um termo obrigatório na busca. Essa técnica recuperará passagens que contenham a resposta candidata sendo utilizada no contexto dos termos da pergunta original. Ainda segundo o autor, existem outras fontes que são utilizadas para buscar esses dados. Após recuperadas, as evidências de apoio são encaminhadas aos componentes que processam a pontuação.

Ferrucci et. al. (2010) explicam que os algoritmos de pontuação determinam o grau de certeza em que uma evidência recuperada sustenta uma resposta candidata. Assim como em todas as etapas do *framework* descritas, a estrutura do DeepQA apoia e incentiva que nesta etapa sejam utilizados muitos componentes diferentes, ou pontuadores como citado pelo autor, para serem considerados diferentes aspectos das evidências no momento da pontuação. Na etapa de pontuação é momento em que a maior parte da análise de conteúdo é executada. Para estimular o uso de diversos pontuadores, o DeepQA fornece um formato padrão para o registro das hipóteses e pontuações de confiança. Isso permite que os desenvolvedores do *framework* desenvolvam e implantem os componentes.

Por meio desta organização o Watson é capaz de empregar mais de 50 componentes de pontuação diferentes, que conforme Ferrucci et. al. (2010) aplicando diferentes algoritmos para produção das pontuações, utilizando as evidências presentes em fontes distintas de informação. Os autores também citam que os pontuadores consideram diversos fatores para geração das pontuações, como o grau de correspondência entre as estrutura de predicado e argumento de uma passagem e da questão, confiabilidade da fonte de passagem, localização geoespacial, relações temporais, classificação taxonômica, relações lexicas e semânticas nas quais a resposta candidata é conhecida, a correlação da resposta candidata com termos presentes na pergunta, sua popularidade (ou obscuridade) entre outras não citadas.

Essas diferentes formas de pontuação são utilizadas dentro do *framework* DeepQA para ser possível analisar as evidências encontradas sobre diversos ângulos. Respostas candidatas que possuam pontuações semelhantes em algumas análises podem divergir significantemente em outras, indicando assim a resposta correta. Ferrucci et. al. (2010) descrevem que todos os apontadores implementados no Watson tem impacto independente sobre o desempenho do *framework* e nenhum algoritmo domina. O autor cita também uma das contribuições importantes do DeepQA é a capacidade de absorver estes algoritmos, por meio das ferramentas que foram criadas.

Com o objetivo de ajudar os desenvolvedores a identificarem como o Watson usa as evidências para decidir a resposta correta, as pontuações geradas nesta etapa são agrupadas de acordo com a sua relação. Esse agrupamento gera um perfil de evidência, que segundo Ferrucci et. al. (2010) fornece uma visão mais intuitiva das informações disponíveis. Assim, os dados dos pontuadores que foram executados para as evidências ficam organizados em grupos, que são uma combinação de pontuações e que possibilitam uma análise das respostas que concorrem em ser a correta.

### Final Merging and Ranking

A última etapa do *framework* tem como objetivo avaliar as centenas de hipóteses baseadas nas pontuações, para identificar as mais bem fundamentadas nas evidências. De acordo com Ferrucci et. al. (2010), para ganhar no Jeopardy, é necessária a habilidade de identificar a resposta precisa, com uma confiança suficiente para apostar que a resposta está correta. Para isso, essa etapa do DeepQA se divide em duas partes, agrupamento de hipóteses e estimativa com classificação da pontuação de confiança.

As respostas candidatas precisam ser agrupadas, pois segundo Ferrucci et. al. (2010), muitas das respostas candidatas para a mesma pergunta podem ser equivalentes. Mesmo se a camada superficial das respostas diferir muito umas das outras, elas podem estar se referindo a mesma resposta final. Como no exemplo do autor, as respostas “Abraham Lincoln” e “*Honest Abe*” mesmo diferentes são equivalentes. Os autores citam que sem a combinação das respostas, os algoritmos de classificação estariam comparando e tentando discriminar múltiplas respostas que no final, representam a mesma resposta. Com isso as respostas são agrupadas antes da estimativa de confiança e do ranking.

Para o agrupamento destas respostas são utilizados algoritmos em conjunto. Alguns citados por Ferrucci et. al. (2010) são: algoritmos de correspondência, normalização e resolução de núcleo. Somente após essa etapa de combinação das respostas que o *framework* segue para o próximo passo, que é estimar a confiança e classificar as hipóteses utilizando essas pontuações combinadas.

Na etapa de estimativa de confiança, que ocorre após o agrupamento das respostas, Ferrucci et. al. (2010) adotaram uma abordagem de aprendizado de maquina. Esse método requer a execução do sistema em um conjunto de perguntas com respostas conhecidas, e o treinamento de um modelo baseado nas pontuações. Esse modelo gerado é adicionado a um sistema, que utiliza vários modelos criados para lidar com diferentes tipos de pergunta. Segundo o autor, essa abordagem permite melhorar o sistema de forma iterativa, com a possibilidade da aplicação de outros modelos mais sofisticados. Assim, o *framework* aplica esse sistema para estimar a confiança de todos os agrupamentos de respostas gerados, para ser possível retornar a resposta com a maior confiança após a classificação.

## Serviços disponíveis (APIs)

Com a criação do Watson e a sua ótima performance nos testes e no programa *Jeopardy*, a IBM transformou o computador e seus algoritmos em serviços separados, que podem ser utilizados por qualquer pessoa ou empresa. De acordo com Gliozzo et al. (2013), a IBM esta investindo agressivamente na transformação do IBM Watson de um protótipo de pesquisa em um sistema altamente adaptável a nível de mercado para ser utilizado por dezenas de empresas.

Os serviços disponibilizados pelo Watson podem ser utilizados acessando a plataforma web da IBM[[11]](#footnote-12), com uma conta gratuita. Ao acessar, existem vários serviços de IA que podem ser criados pelo usuário em sua conta, para utilizar com os seus dados ou com os exemplos previamente fornecidos pela IBM para aprendizado e testes. Estes serviços podem ser utilizados por desenvolvedores, realizando chamadas às API do Watson, especificando o serviço que desejam utilizar. Cada serviço possui uma configuração distinta para utilização, assim como parâmetros de entrada diferentes.

No Watson são oferecidos diversos serviços que podem ser utilizados para a criação de um *chatbot*, dentre eles estão, Watson *Assistant*, Watson *Discovery*, Watson *Tone Analyzer* e Watson *Natural Language Understanding*. Cada serviço disponibilizado atualmente no ambiente da IBM possui um propósito especifico na área de processamento de linguagem natural, manipulando os dados de maneiras distintas. Todos esses serviços operam de maneira separada, possuindo informações de entrada e saída diferentes e não necessitando um do outro para funcionamento.

Os serviços e o seu funcionamento serão detalhados nas próximas seções. Para isso, serão utilizados alguns exemplos de sentenças e palavras para demonstrar como os serviços processam essas entradas e quais os dados que são devolvidos como resultado.

### Watson Assistant

O Watson *Assistant* é uma API desenvolvida especificamente para a criação de *chatbots* e utiliza técnicas de processamento de linguagem natural para extrair as intenções das sentenças analisadas, com a possibilidade de extrair dados que complementam essas intenções e são chamadas de entidades. As entidades servem para completar as intenções do usuário, sendo possível coletar dados sobre uma intenção e criar tratamentos diferentes para a mesma intenção. (WATSON ASSISTANT, 2018).

De acordo com Watson Assistant (2018), as intenções são objetivos que os desenvolvedores devem prever que os seus usuários terão quando conversarem com o *chatbot*. Com isso, para cada objetivo que o usuário possa ter ao utilizar o serviço, deve ser definida uma intenção. Ao serem definidas as intenções, devem ser fornecidas sentenças de exemplo, que reflitam o objetivo da intenção. Essas frases são utilizadas para treinamento do Watson. Já as entidades, segundo Watson Assistant (2018), representam um termo ou objeto que fornece contexto a uma intenção.

A Figura 4 apresenta o diálogo entre duas pessoas, Nelson e Marie. Na imagem Nelson está realizando uma pergunta para Marie, que pode ajudá-lo. Para ser possível construir um *chatbot* capaz de ajudar Nelson é necessário identificar a intenção do mesmo com a pergunta, assim como as entidades presentes nela. Neste caso, a intenção do usuário é “*find\_a\_place*” (encontrar um lugar, tradução nossa) e a entidade é “*transp\_landmark*” (meio de transporte, tradução nossa). Assim, o *chatbot* precisa ser treinado para identificar que a intenção do usuário é encontrar um local e a entidade é um local específico, neste caso, a estação de trem. Identificando esses elementos dentro de uma sentença do usuário é possível definir a ação que será tomada pela aplicação.

Dentro do Watson *Assistant* existe também a criação e configuração do diálogo, que é o fluxo que incorpora as intenções e entidades, e deve ser seguido pelo *chatbot* durante a interação com o usuário, este diálogo cria uma árvore de decisão com vários nodos. Cada nodo mapeia uma intenção, com a(s) resposta(s) que podem ser retornadas para o usuário. No diálogo existe também um contexto, que armazena todas as informações coletas pelo *chatbot* durante a interação com o usuário em forma de variáveis para ser possível utilizar posteriormente no próprio fluxo da conversa, ou em alguma outra funcionalidade ou serviço que o *chatbot* oferecerá. (WATSON ASSISTANT, 2018).

Figura 4 – Intenção e entidade em uma conversa



Fonte: (Azraq et. al. 2017).

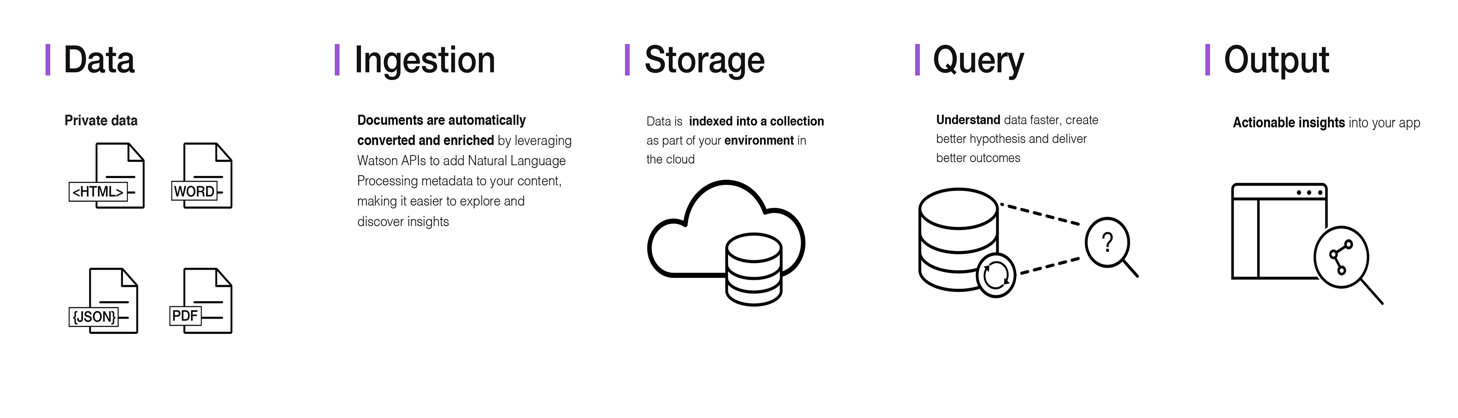
O Watson *Assistant* é uma das ferramentas que será utilizada para desenvolvimento da solução deste trabalho, pois suporta o idioma Português do Brasil e torna possível controlar o fluxo do diálogo, mapeando intenções e os dados vinculados a elas. Para criação de um atendente é necessário identificar as intenções do usuários e os dados relacionados a elas, e para isso o serviço do Watson *Assistant* é o mais indicado dentro das ferramentas disponíveis na API.

### Watson Discovery

Watson *Discovery* também possui PLN para analisar dados, realizando essa análise nos materiais que forem importados para dentro da plataforma. Esses dados, são documentos, que podem estar na forma estruturada ou não estruturada. As informações são analisadas pelo serviço, que treina com os dados e cria a possibilidade de extrair as informações e respostas relevantes para as perguntas dos usuários. Dentro de um *chatbot* esse serviço pode ser utilizado para carregar uma base de informações de uma FAQ por exemplo, e através de uma análise retornar a resposta mais precisa para o usuário. (WATSON DISCOVERY, 2018).

Uma representação completa da arquitetura do Watson *Discovery* pode ser observada na Figura 5. De acordo com Gliozo et. al. (2017) o serviço fornece um *pipeline* de recursos para ingestão, enriquecimento e armazenamento de grandes quantidades de dados não estruturados. A Figura 5 apresenta os componentes na ordem em que são utilizados, da esquerda para a direita, primeiro os documentos com os dados desejados são importados no serviço, que aplica o enriquecimento disponível nos dados. Na sequência, os dados ficam armazenados na nuvem da IBM, dentro do serviço. Então, esses dados ficam disponíveis para criação de consultas e podem ser utilizados por exemplo, dentro de um aplicativo.

Figura 5 – Fluxo de importação dos documentos para o *Discovery*



Fonte: (WATSON DISCOVERY, 2019).

Com a ferramenta é possível importar dados não estruturados, de diversas fontes e gerar consultas utilizando estes dados. Os dados são importados, convertidos, enriquecidos e normalizados. Com isso, é possível criar relações personalizadas nos dados importados com a NLU (*Natural Language Understandig*). (WATSON DISCOVERY, 2018).

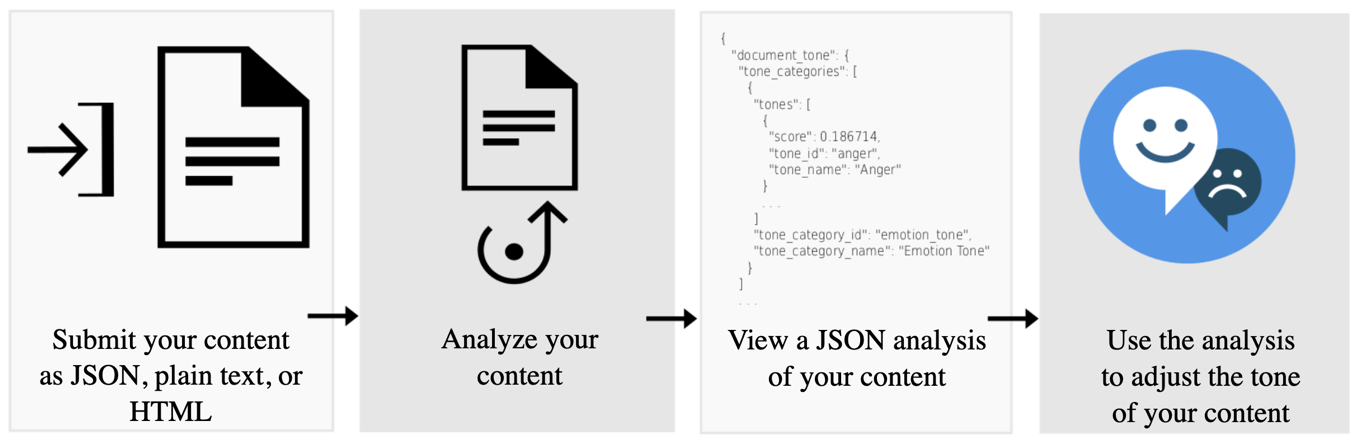
### Watson Tone Analyzer

Watson *Tone Analyzer* é um serviço que utiliza análise linguística para detectar tons emocionais e de linguagem no texto por escrito, sendo possível analisar o tom em documentos ou em sentenças enviadas para o serviço. (WATSON TONE ANALYZER, 2018). Através disto, é possível determinar as emoções e o estilo de comunicação, retornando quando o usuário esta irritado ou bravo, por exemplo, com uma pontuação atribuída a emoção. O serviço pode ser utilizado a nível de documento ou de sentença.

Como cita Watson Tone Analyzer (2018), o serviço pode ser utilizado para entender como as comunicações por escrito de uma empresa ou usuário são percebidas por quem as lê, dessa maneira é possível melhorar o tom das comunicações com os clientes. Além disso, as empresas podem utilizar o serviço para aprender o tom das comunicações dos seus clientes, para ser possível responder a cada cliente da forma apropriada, ou para entender e melhorar as suas conversas no geral.

Na Figura 6 é possível observar o fluxo que um documento ou sentença segue dentro do serviço quando enviado. O Watson *Tone Analyzer* recebe o arquivo ou sentença, analisa o conteúdo e retorna um JSON com todas as emoções extraídas sobre os dados. O resultado pode então ser utilizado dentro de outra aplicação. Um dos casos de uso interessantes para o serviço, conforme Watson *Tone Analyzer* (2018), são os *chatbots,* pois o serviço torna possível que o agente detecte tons dos clientes e monte respostas adequadas com essa informação. Por exemplo, é possível responder à tristeza com “Sinto muito que você esteja se sentindo assim” ou a satisfação com “Estou feliz que esteja satisfeito com o nosso serviço”.

Figura 6 – Fluxo do serviço *Tone Analyzer*



Fonte: (WATSON TONE ANALYZER, 2018).

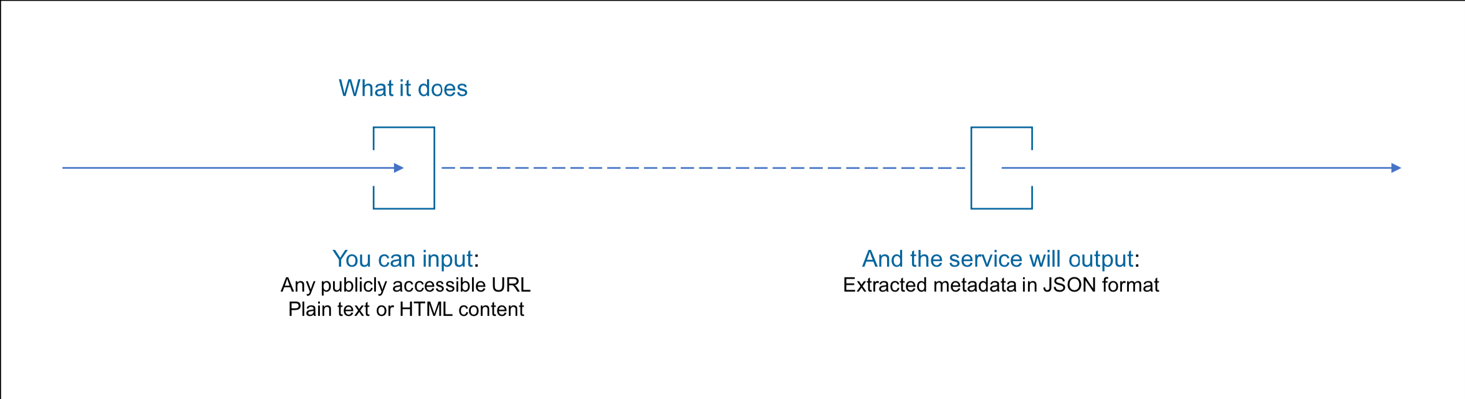
O serviço de análise do tom das conversas tem a capacidade de incluir empatia em um *chatbot*, ou tratar situações em que seja necessária a intervenção de um atendente humano. Assim sendo, esse serviço possui grande importância na implementação de um *chatbot*, e também deve ser incorporado ao sistema de atendimento proposto neste trabalho.

### Watson Natural Language Understanding

Watson *Natural Language Understanding* é um serviço do Watson onde é possível realizar PLN em textos fornecidos para o serviço, analisando os recursos de semântica da entrada de texto. Com ele é possível obter categorias, conceitos, emoções, entidades, palavras-chave, metadados, relações e a impressão. A análise pode ser feita a nível de palavra ou de sentença. Uma diferença do *Watson Natural Language Understanding* é que o serviço pode ser estendido de forma personalizada com modelos customizados que identificam entidades customizadas e relações exclusivas do domínio do usuário. (WATSON NATURAL LANGUAGE UNDERSTANDING, 2018).

Por meio da Figura 7 é possível visualizar uma representação de alto nível do funcionamento da API do Watson *Natural Language Understanding*. Neste serviço é possível informar como entrada qualquer informação em texto, endereço eletrônico de um site ou um arquivo no formato HTML. Recebendo esta entrada, o serviço realiza a análise e retorna os dados em formato *JSON* (*JavaScript Object Notation*), que é uma formatação leve para troca de dados, facilmente interpretada e gerada por máquinas (JSON, 2019).

Figura 7 – Representação de alto nível do NLU



Fonte: (VERGARA, Sebastian et al. 2017).

Segundo Vergara et. al. (2017) os desenvolvedores são capazes de extrair diversos tipos de metadados do texto de entrada utilizando o Watson *Natural Language Understanding*. Sendo eles, categorias, conceitos, emoções, entidades, palavras-chave, relações, papéis semânticos e sentimentos. O serviço também pode ser personalizado para identificar entidades específicas do domínio do conteúdo que for enviado. Estas características do serviço o tornam muito útil em cenários que exigem uma rápida análise de textos não estruturados, sem a necessidade do conhecimento aprofundado no campo de processamento de linguagem natural.

# Chatbot Desenvolvido

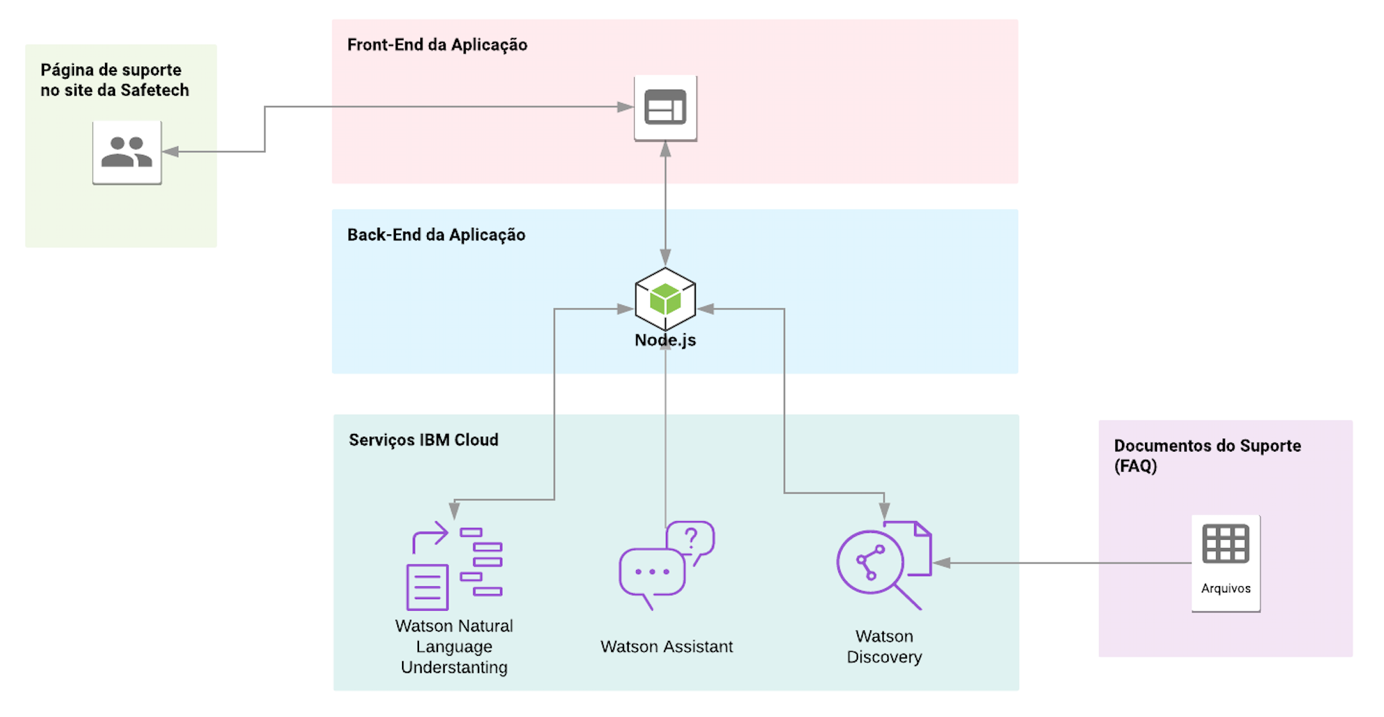
O chatbot criado neste trabalho consiste de um sistema capaz de responder as dúvidas dos clientes do sistema, referentes ao negócio em que está inserido. O sistema permite que sejam adicionados arquivos a base do *chatbot*, com conteúdo explicativo sobre o sistema e sobre como resolver problemas específicos, com o objetivo de aumentar a base de conhecimento do mesmo. O sistema manterá um registro de todas as conversas efetuadas com o *chatbot*, para que seja possível analisar as respostas fornecidas, a fim de refinar o fluxo do diálogo e as conversas fornecidas, mapeando perguntas não entendidas. O idioma utilizado na comunicação com os usuários é o português.

O acesso ao sistema é efetuado através do software de emissão de nota fiscal da Safetech informática, que está disponível para os clientes da empresa no endereço [mail.sft.inf.br:8088/nfe-rs](http://mail.sft.inf.br:8088/nfe-rs). No portal existe uma seção de ajuda, e nesta seção, foi adicionada a possibilidade do usuário realizar o auto atendimento conversando com o chatbot. Este acesso abre uma página, que possui a interface de usuário para comunicação com o sistema. Nesta página o usuário faz as perguntas, como em um chat normal e visualiza as respostas fornecidas pelo sistema. O chatbot busca responder as perguntas dos usuários sem a necessidade de um atendente humano da equipe de suporte técnico.

Para a implementação da interface de usuário do chatbot, foram utilizadas as linguagens *HTML5*[[12]](#footnote-13), *CSS*[[13]](#footnote-14) e *JavaScript*[[14]](#footnote-15). Linguagens estas, amplamente utilizadas para criação do *front-end* de aplicações web modernas, permitindo a criação de uma interface em que o usuário digita a sua pergunta ou resposta em um campo na tela, e visualiza de forma separada as mensagens do sistema e as suas. Para controle do *back-end* foi criado um servidor utilizando um ambiente *Node.js[[15]](#footnote-16).* O que significa que todo o código do servidor foi escrito na linguagem *Javascript.* A linguagem foi escolhida devido a existência de uma SDK[[16]](#footnote-17) (*Software development Kit*) para uso com as bibliotecas do Watson da IBM e por ser projetada para o desenvolvimento de aplicações escaláveis em rede (NODE.JS,2019). Servidor este que é chamado de orquestrador neste trabalho, pois é o elo de ligação do *front-end* com o *back-end* e com as APIs do Watson.

A arquitetura utilizada para o chatbot é apresentada na Figura 8, sendo possível observar que o *chatbot* desenvolvido possui 3 camadas principais. O *Front-end,* responsável por toda a parte visual da aplicação que é exibida para os utilizadores do sistema, local em que ocorre o acesso dos usuários, consiste em receber as entradas de texto que o usuário fornece e realizar as solicitações ao servidor de *back-end*. Após o envio, o *front-end* fica aguardando uma resposta do servidor, e ao receber, formata a mesma para exibir de forma legível ao usuário no chat.

Figura 8 – Arquitetura do Chatbot desenvolvido



Fonte: do autor.

O orquestrador, que é o *back-end* da aplicação criado em *Node.js*, foi nomeado de orquestrador porque integra a aplicação na qual o usuário interage, com os serviços de IA (Inteligência Artificial) disponíveis na nuvem da IBM. A cada entrada de texto do usuário, o orquestrador processa o texto em um fluxo definido no algoritmo, para verificar quais serviços deve invocar antes de retornar uma resposta para a pergunta do usuário. Cada serviço possui uma configuração e parâmetros distintos para funcionamento. Com isso, o *back-end* trata da parte de receber os dados do usuário, que vem do *front-end,* e verifica qual serviço do Watson deve ser consultado e em que ordem. O orquestrador então recebe a resposta dos serviços, analisa e formata a mesma a fim de retornar ao *front-end* a resposta final do chatbot, que será exibida para o usuário.

Os serviços utilizados no desenvolvimento do *chatbot* foram o Watson *Natural Language Understanting*, *Watson Assistant* e *Watson Discovery*. O serviço do *Watson Tone Analyzer* teve a implementação proposta, porém não foi utilizado pois mesmo retornando o resultado da análise de emoção nas sentenças no idioma português, ainda não analisa entradas de texto em português, sendo necessária uma tradução nas sentenças antes de enviar ao serviço, o que não seria interessante neste trabalho devido as alterações de intenção que a tradução possa causar.

Um *chatbot* desenvolvido utilizando os serviços da IBM necessita primordialmente que o serviço do *Watson Assistant* seja utilizado. Esse serviço é o que constitui as partes principais de uma *chatbot* desenvolvido nesta plataforma, pois é nele que são configurados os 3 elementos principais do *chatbot*. São eles: (i) Intenções e os seus exemplos, para ser possível identificar o que o usuário deseja com a entrada de texto que ele informou. (ii) As entidades, que são complementos de informação das intenções, para ser possível coletar dados sobre uma pergunta ou afirmação do usuário. (iii) E, por fim, existe o diálogo, que é a estrutura em que se configura a forma como as entradas de texto do usuário serão processadas pela ferramenta.

Atualmente dentro da ferramenta podem ser criados diversos *skills* (habilidades) e *assistants (*assistentes). As *skills* são as habilidades que um chatbot possui, ou seja, os assuntos ou tópicos que ele tem capacidade de compreender e processar. O *assistant* é um agregador de *skills,* podendo um *chatbot* possuir uma ou mais habilidades. Neste trabalho, a *skill* é a capacidade do *chatbot* em processar assuntos sobre o sistema de emissão de notas da empresa e o *assistant* é o serviço que disponibiliza a *skill* para utilização via API. Esses componentes são utilizados para organização dos *chatbots* na plataforma da IBM. Além disso, eles tornam os agentes criados com essa ferramenta mais escaláveis e com a possibilidade de aproveitamento de *skills* em diferentes agentes conversacionais*.* Dentro da ferramenta podem ser criados diversos *skills* e diversos *assistants.*

Atuando como uma ferramenta de análise de linguagem natural, o Watson *Natural Language Understanting* processa entradas de texto fornecidas a ele, retornando diversas informações sobre o texto analisado, como por exemplo, nomes de pessoas presentes na sentença. A ferramenta funciona de maneira independente comparado ao *Assistant,* pois ela retorna o resultado da análise sobre o texto, não uma ação ou resposta baseada neste texto como é o caso o *Assistant*. Com isso, o serviço pode ser utilizado para efetuar uma análise completa na entrada de texto do usuário, a fim de identificar palavras-chave que sejam importantes para a resposta, ou que de alguma maneira possam fornecer dados para o chatbot compor a resposta.

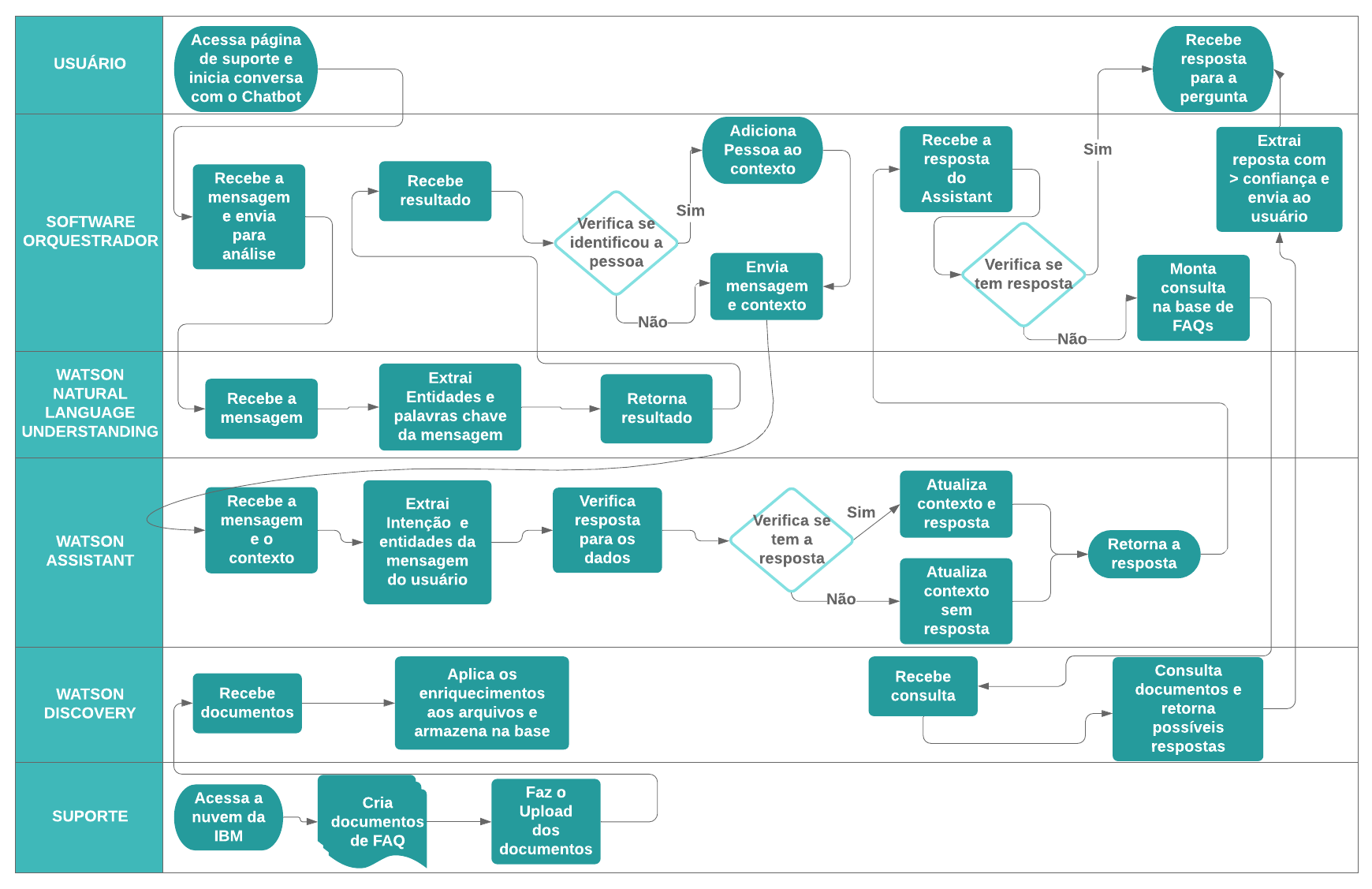
O serviço do *Watson Discovery* é a ferramenta do *chatbot* que pode ser chamada de banco de dados, devido a ser o local onde os arquivos de suporte do sistema são importados para análise do serviço. Após a análise do serviço é possível efetuar *querys* (consultas) nos documentos através de uma entrada de texto, retornando os trechos de conteúdo dos arquivos, chamados de passagens, que possuam maior proximidade com a entrada fornecida. Este serviço é disponibilizado para upload de arquivos e análises sobre os dados e a linguagem dos mesmos e com a utilização do orquestrador é possível integrar a mesma ao chatbot, o que transforma o serviço em um banco de dados do *chatbot*, pois todos os manuais ou documentos previamente escritos para suporte dos usuários podem ser reaproveitados.

A proposta do *chatbot* desenvolvido é o orquestrador enviar as entradas de texto do usuário para ao serviço do *Watson NLU,* antes de enviar ao *Assistant,* para ser possível enriquecer o contexto do diálogo com dados sobre a entrada de texto e enviar para o serviço do *Watson Assistant*. Neste serviço estão mapeadas as principais dúvidas dos usuários*,* que são processadas na árvore de diálogo, para retornar uma resposta, ou uma ação que o orquestrador deva tomar antes de fornecer a resposta ao usuário. As dúvidas ou problemas que não forem encontrados desta maneira, o orquestrador faz uma consulta aos documentos que foram importados no *Watson Discovery*. Caso a análise nos documentos encontre alguma resposta que possua uma pontuação elevada de correspondência com a questão, o *back-end* retornará a mesma para o *front-end* e em consequência para o usuário.

O diagrama de processo da Figura 9, explica de forma macro o fluxo que segue uma pergunta do usuário ao ser submetida ao chatbot. Na esquerda do diagrama estão os elementos principais que compõem e participam do fluxo de uma conversa do *chatbot.* Na direita dos elementos estão descritos as ações que este elemento executa, assim como a etapa em que é executada. Pode ser observado que as entradas de texto sempre passam pelo orquestrador, que sempre utiliza os serviços do *Watson NLU* e *Watson Assistant,* assim como sempre é fornecido algum retorno para o usuário. O serviço do *Watson Discovery* pode não ser utilizado em um fluxo de pergunta-resposta, pois é utilizando somente quando a resposta não é encontrada nas que foram mapeadas no serviço do *Assistant.*

No fluxo é possível observar como os serviços distintos do *Watson* da IBM foram integrados para chegar no resultado final do chatbot deste trabalho. Cada serviço disponível proporciona uma forma de análise linguística diferente nos textos e sentenças fornecidos. Desta forma, a capacidade do *chatbot* tem alta correlação com a quantidade de serviços envolvidos no desenvolvimento do mesmo. Assim como observado nos capítulos anteriores, em que o Watson foi desenvolvido para uma competição, não existe uma maneira de resolver todos os problemas da análise linguística e da resolução de questões com somente um algoritmo ou serviço. A integração correta entre os serviços disponíveis pela IBM, assim como uma correta configuração é uma forma de fazer um *chatbot* ser assertivo ou não, podendo ser realizada de diferentes maneiras, dependendo do objetivo.

Figura 9 – Diagrama do processo do diálogo



Fonte: do autor.

Nas seções seguintes, veremos com mais detalhes todo o trabalho que foi realizado para o desenvolvimento do *chatbot*. Iniciando na coleta de dados sobre o sistema em que o *chatbot* presta suporte, indo até a geração de dados para treinamento e configuração. Detalhando a configuração dos serviços da IBM e o desenvolvimento do software orquestrador, assim como a definição das intenções, entidades e do fluxo de diálogo. Construção e definição dos documentos que foram utilizados para base de dados, até os testes realizados após a implementação do serviço para mapear falhas e pontos de melhoria do *chatbot*.

## Mapeamento das intenções

O *chatbot* disposto neste trabalho foi desenvolvido para auxiliar o suporte de uma empresa com o objetivo de reduzir os atendimentos humanos. Para isso, se fez necessária a obtenção de dados sobre o sistema atual a fim de mapear as dúvidas mais frequentes, assim como demais indicadores sobre o produto que o *chatbot* presta suporte. Assim sendo a equipe de suporte, que atende o sistema diariamente, registrou os dados de atendimento aos clientes que entravam em contato com dúvidas ou problemas sobre o sistema durante o período de um mês. Os dados foram coletados de forma sintética, sendo registrada a data do atendimento, o cliente, uma descrição resumida do problema e o canal em que o cliente entrou em contato.

Além desta coleta de dados sobre os atendimentos prestados, a equipe de suporte também criou uma série de vídeos com treinamentos sobre o sistema. Esses vídeos foram divididos em tópicos dentro dos módulos do sistema, com o mesmo conteúdo ensinado aos usuários do sistema no treinamento inicial. O objetivo da equipe com a criação dos vídeos foi de gerar material de suporte sobre o sistema, que poderia ser consultado posteriormente por algum cliente que necessitasse de treinamento. O que ocorre é que muitos clientes são pequenas empresas e utilizam pouco o sistema, consequentemente acabam esquecendo de como realizar os processos básicos, e procuram a equipe de suporte com dúvidas.

Analisando os dados obtidos no levantamento dos atendimentos prestados aos usuários do sistema durante um mês, foi possível constatar que a equipe de suporte registrou 255 atendimentos no total, que se somado ao fato de que existem uma média de 265 clientes que utilizam o sistema e emitem notas durante o mesmo período, pode-se concluir que o número de atendimentos durante o mês é quase o mesmo número de clientes ativos. O número de notas emitidas no sistema durante o mês é muito superior ao número de clientes do sistema, porém, considerando o número de clientes ativos no período com o número de atendimentos é como se cada cliente que usou o sistema, precisou entrar em contato antes.

Para geração de dados para o *chatbot,* foram assistidos os vídeos criados pela equipe de suporte, que possuem um treinamento sobre o sistema para sanar as dúvidas mais frequentes dos usuários. Como os dados são necessários em texto, foi realizada uma transcrição manual dos vídeos para texto, organizando o conteúdo na forma de tópicos, para tornar possível mapear as intenções e entidades necessárias para o chatbot, assim como criar as respostas padrões para as perguntas mapeadas. Desta maneira, as dúvidas mais frequentes dos usuários foram mapeadas no *Watson Assistant,* enquanto as dúvidas menos frequentes e mais difíceis de serem mapeadas serão consultadas no *Watson* *Discovery*. Ao todo, foram analisados 11 vídeos, totalizando 17 minutos e 33 segundos, que geraram 19 tópicos e foram organizados dentro de um arquivo de texto.

Uma amostra do resultado da compilação das dúvidas mais frequentes dos usuários, utilizada para configuração do *Watson Assistant* na versão inicial pode ser vista na Tabela 1. Como pode ser observado na tabela, as perguntas têm uma intenção bem definida do que o usuário deseja fazer, ou de qual é a sua dificuldade. A equipe de suporte do sistema elaborou todas as respostas para essas perguntas mapeadas dos usuários. Para utilização no *chatbot,* a resposta foi utilizada na íntegra, somente com as modificações para responder ao usuário como se fosse o *chatbot* respondendo.

Tabela 1 – Amostra de perguntas mapeadas.

|  |  |
| --- | --- |
| Pergunta | Resposta |
| |  |  | | --- | --- | | Como cadastrar produtos? | Clicar em Cadastros;  Clicar em Produto;  Preencher os campos correspondentes a: Código, Descrição, NCM (buscando no cadastro do sistema) e Unidade de Medida. Os demais campos não precisam ser preenchidos;  Clicar em Salvar; | | Onde colocar as informações adicionais do produto? | Clicar em Faturamento;  Clicar em Digitação de Notas;  Na 4ª Guia – Produtos e Serviços, onde são inseridas as informações do produto, terá um campo Informações Adicionais do Produto; Neste campo, você poderá complementar alguma informação sobre o produto. Ex: cor, modelo, referencia. | | Como proceder quando aparecer a mensagem de certificado vencendo? | Quando aparecer o aviso que o certificado esta expirando, favor entrar em contato com o seu contador para lhe informar e solicitar um novo certificado A1;  É necessário que seja enviado o novo arquivo para o suporte da Safetech suportenfe@safetech.inf.br para que seja atualizado; | | Como duplicar uma nota? | Clicar em Faturamento;  Clicar em Consulta de Notas;  Na tela estarão todas as notas emitidas até a presente data.  Selecione o botão Duplicar Nota (último botão do lado direito da tela);  Abrirá a nota e você poderá fazer as devidas alterações; | | |

Fonte: do autor.

Segundo Mathur (2017) as entradas de texto que são fornecidas às interfaces de conversação virtuais podem ser classificadas de uma maneira abrangente em dois grupos, *short tail (*cauda curta) e *long tail (*cauda longa). *Short tail* se refere as entradas de texto que aparecem mais frequentemente e podem ser mapeadas para uma série de intenções pré-definidas, como as perguntas de uma FAQ. De maneira oposta, a *long tail* contém perguntas que ocorrem com menos frequência e em virtude disto podem não ter uma resposta facilmente definida, necessitando uma busca na base de conhecimento para encontrar conteúdo relevante. Em vista disto, as questões mais frequentes dos usuários foram classificadas e tiveram as intenções mapeadas, enquanto que para as questões que não forem encontradas será realizada uma busca nos documentos do suporte para buscar conteúdo para a resposta.

O Watson Assistant é o serviço utilizado para tratar essas questões da *short tail,* por ser o mais adequado para este tipo de questão, porque mapeia as intenções do usuário em grupos chamados de *intents (*intenções)*.* Após fornecida a intenção é necessário informar frases de exemplo para o treinamento do Watson com aquela intenção. Com a intenção criada no serviço, o *Assistant* é capaz de identificar quando a frase do usuário estiver expressando aquela intenção, com uma pontuação de confiança atribuída. Quando a intenção for detectada o serviço toma a ação que estiver especificada na árvore do diálogo para a mesma, que será detalhada na próxima seção. Dentro da ferramenta as intenções recebem o carácter # antes da definição do nome da intenção e as entidades recebem o caractere @, também antes do nome.

Para organização, todas as intenções mapeadas na plataforma, referentes ao suporte do sistema, receberam o prefixo “ajuda” após o caractere que define a intenção e antes do nome da intenção. A demais intenções recebem o prefixo “geral” antes do nome da intenção. Essa técnica será utilizada para separar as intenções principais do sistema, que são as de auxilio sobre o sistema de NF-e, e das intenções de controle necessárias para o *chatbot,* que serão detalhadas posteriormente. Desta forma, uma das intenções criada no serviço do *Watson Assistan*t foi “#ajuda\_duplicar\_nota”, presente na amostra de dúvidas mapeadas na Tabela 1. Este nome de intenção será vinculado a frase do usuário quando a análise retornar que o usuário deseja duplicar uma nota.

Ao cadastrar uma intenção no *Watson Assistant* é possível informar os exemplos de sentenças do usuário que indicam a intenção, como pode ser observado na Tabela 2. A ferramenta não obriga que os exemplos sejam informados nesta etapa, mas sem esta informação não será possível mapear nenhuma conversa do usuário para essa intenção. Não foi encontrada a especificação de um número mínimo de exemplos que devem ser fornecidos, porém a capacidade do serviço em identificar a intenção está altamente relacionada com a quantidade e a qualidade dos exemplos providos. Para todas as perguntas mapeadas como principais, foram criados exemplos semelhantes de variações na forma de perguntar, conforme os exemplos da Tabela 2.

Tabela 2 – Exemplos fornecidos para treinamento de uma intenção.

|  |  |
| --- | --- |
| Intenção | Exemplos |
| #ajuda\_duplicar\_nota | Como duplicar uma nf-e? |
| Como duplicar uma nota? |
| como eu faço para copiar uma nota fiscal? |
| de que maneira eu copio uma nota fiscal? |
| é possível duplicar uma nota? |
| eu tenho um nota e gostaria de fazer outra igual |
| posso copiar uma nota? |

Fonte: do autor.

Para a versão inicial do *chatbot* foram mapeadas no total 18 intenções de ajuda, que podem ser visualizadas na Tabela 3 e que foram geradas através das perguntas mapeadas como mais frequentes. Assim também, foram geradas 12 intenções de controle, nomeadas como gerais e presentes na Tabela 4. Portanto, para a versão inicial do agente foram mapeadas 30 intenções no total, que o mesmo será capaz de detectar para ser possível determinar a ação do *chatbot*. Para estas 30 intenções foi informado uma média de 10 exemplos para cada, com variações da pergunta original, porém com o mesmo sentido e objetivo final.

Tabela 3 – Intenções de ajuda mapeadas.

|  |  |  |
| --- | --- | --- |
| #ajuda\_baixar\_xml | #ajuda\_consulta\_nf | #ajuda\_info\_adicionais\_nf |
| #ajuda\_cadastro | #ajuda\_duplicar\_nota | #ajuda\_info\_adicional\_produto |
| #ajuda\_cancelar\_nota | #ajuda\_emitir\_nota | #ajuda\_inutilizar\_nota |
| #ajuda\_carta\_correcao | #ajuda\_enviar\_nf\_email | #ajuda\_prazo\_cancelamento |
| #ajuda\_certificado\_vencendo | #ajuda\_geral | #ajuda\_re\_imprimir\_danfe |
| #ajuda\_cliente\_recebeu\_email | #ajuda\_imprimir\_danfe | #ajuda\_solicitacao\_de\_notas |

Fonte: do autor.

Algumas das dúvidas mapeadas na versão inicial foram convertidas em intenções agrupadas. Como é o caso das intenções “#ajuda\_cadastro” e “#geral\_forma\_contato”. Esse agrupamento foi criado devido a intenção do usuário com a pergunta ser a mesma, porém em entidades diferentes. Em outras palavras, quando o usuário deseja realizar um cadastro no sistema a sua intenção é a de cadastrar, indiferente do objeto que ele deseja cadastrar. Porém, ele pode cadastrar mais de um objeto, que neste caso pode ser uma pessoa jurídica, uma pessoa física ou um produto. Nestes casos a intenção foi mapeada como uma ajuda em cadastro e o *chatbot* precisará saber em que cadastro a ajuda é necessária.

Para ser possível descobrir detalhes sobre uma intenção do usuário, como no exemplo do cadastro ou da forma de contato, são definidas as entidades. Entidades são informações que complementam as intenções e nestes casos específicos podem ser utilizadas para identificar o que o usuário deseja cadastrar distinguindo sobre qual objeto se trata o cadastro. Ou ainda como o mesmo deseja entrar em contato, no caso da intenção “#geral\_forma\_contato”. Logo, para descobrir essas informações foi necessária a criação de 3 entidades, que quando identificadas pelo *chatbot* tornam possível discernir com precisão o que o usuário deseja, e sobre o que ele deseja.

No Watson *Assistant* as entidades são cadastradas de maneira semelhante as intenções, porém ao invés de exemplos para cada entidade, são fornecidos sinônimos para a palavra mapeada como uma entidade. Com isso, o cadastro das entidades é composto de nome da entidade, valor da entidade e sinônimos para aquela entidade. A Tabela 4 foi construída para apresentar as entidades que foram mapeadas na versão inicial da ferramenta. Desta maneira, foi possível obter mais dados sobre a intenção do usuário, dados que são necessários para os exemplos com mais de uma opção dentro da mesma intenção, conforme citado acima.

Tabela 4 – Entidades do *Chatbot*.

|  |  |  |
| --- | --- | --- |
| Entidade | Valor | Sinônimos |
| @forma\_contato | E-mail | Email, mail |
| @forma\_contato | Skype | Mensagem |
| @forma\_contato | Telefone | Número, ligação, plantão, celular, fixo, móvel, fone |
| @forma\_contato | WhatsApp | Whats app, wpp |
| @pessoa | Física | Física, consumidor |
| @pessoa | Jurídica | Empresa, fábrica, loja, fornecedor, revendedor,representante,jurídica |
| @produto | Produto | Item, mercadoria |

Fonte: do autor.

Dentro das intenções também foi mapeada na versão inicial uma intenção chamada de “#ajuda\_geral”. O objetivo desta intenção era de identificar quando o usuário precisava de ajuda ou estava com dúvidas e a dúvida não fosse encontrada nos exemplos anteriormente mapeados das intenções. Para isso, foram fornecidos exemplos de sentenças genéricas pedindo ajuda ou informando algum erro no sistema. Quando o *chatbot* não identificasse a intenção do usuário, ele poderia detectar a mesma como geral e indicar para o serviço apontar ao orquestrador que deveria ser realizada uma busca na base de documentos. Essa configuração do diálogo será apresentada a seguir, assim como o resultado desta abordagem.

Conforme Cummins (2018), uma das boas práticas para desenvolvimento de um *chatbot* é criar um mecanismo em que o usuário possa indicar de alguma maneira que ele não deseja mais ser atendido pelo sistema, mas sim por um atendente real. Essa necessidade pode ocorrer por vários motivos, como quando o *chatbot* não consegue fornecer uma explicação satisfatória para o usuário, ou quando é necessária uma intervenção técnica para resolução do problema ou quando o cliente estiver com raiva. Para atender essa funcionalidade foi mapeada uma intenção chamada de “#ajuda\_pedir\_uma\_pessoa” que contém os exemplos para identificar quando o usuário deseja falar com uma pessoa real. Essa intenção pode ser visualizada na Tabela 5, juntamente com as demais intenções de controle criadas.

Tabela 5 – Intenções gerais e de controle mapeadas.

|  |  |
| --- | --- |
| #geral\_capacidades\_atendente | #geral\_humano\_ou\_robo |
| #geral\_feedback\_negativo | #geral\_pedir\_uma\_pessoa |
| #geral\_feedback\_positivo | #geral\_saudacao |
| #geral\_finalizar | #geral\_sobre\_chatbot |
| #geral\_forma\_contato | #resposta\_nao |
| #geral\_horario\_atendimento | #resposta\_sim |

Fonte: do autor.

Na próxima seção será possível verificar que a primeira interação do *chatbot* com o usuário é uma saudação, seguida de sua apresentação. A tendência natural de um diálogo é de o usuário responder ao chatbot com outra saudação. Por isso, é necessário identificar quando a mensagem do usuário se trata de uma saudação, para lhe responder de maneira adequada a mensagem e não tentar mapear a mensagem como uma dúvida. Por isso, foi criada a intenção “#geral\_saudacao” com treinamentos de saudações, para o *chatbot* conseguir identificar quando o usuário estiver realizado uma saudação. De maneira oposta, foi criada a intenção “#geral\_finalizar” para identificar quando o usuário deseja finalizar a conversa.

As intenções “#geral\_feedback\_negativo” e “#geral\_feedback\_positivo” foram mapeadas no agente conversacional com o objetivo de fornecer respostas ao usuário quando o mesmo provê um retorno sobre o chatbot. Ou seja, caso o usuário agradeça ao *chatbot*, o sistema pode fornecer uma resposta de agradecimento pelo retorno, ou quando o *feedback* (retorno) for negativo, o *chatbot* pode explicar ao usuário que está em construção e sempre aprendendo para melhor ajudar o usuário. O propósito da implementação destas duas intenções é o de responder corretamente ao usuário quando ele agradece ou critica o *chatbot,* para que frases do usuário que tenham estas intenções não sejam mapeadas incorretamente.

Ainda com o objetivo de melhorar a experiência do usuário na conversa com o *chatbot,* foram implementadas as intenções “#geral\_capacidades\_atendente” e “#geral\_humano\_ou\_robo”. A primeira trata das perguntas que tem relação com os conhecimentos do atendente virtual, sendo possível listar para o usuário no que o *chatbot* pode ser útil. A segunda intenção foi criada para ser possível responder aos usuários que questionam sobre o atendente na conversa ser uma pessoa real ou um robô, nesses casos o sistema irá responder que é um robô criado para melhorar o atendimento ao cliente.

A equipe do suporte também propôs que o chatbot deveria ser capaz de responder aos usuários sobre as formas e horários de atendimento. Para isso, inicialmente foi mapeada a intenção “#geral\_horario\_atendimento” com exemplos de quando o usuário deseja saber o horário em que o suporte trabalha na empresa, assim como os dias. Foi criada também a intenção para mapear quando o usuário pergunta sobre como entrar em contato, essa com o nome de “#geral\_forma\_contato”. Em suma, o *chatbot* será capaz de responder para o usuário em quais horários a equipe da empresa está disponível para atendê-lo e através de quais canais de atendimento ele pode entrar em contato.

Uma das abordagens utilizadas para reduzir o tamanho de algumas respostas para o usuário final, foi a de confirmar algumas informações com o usuário durante o diálogo. Um exemplo desta situação é a intenção chamada de “#ajuda\_inutilizar\_nota”, para esta intenção existem duas respostas diferentes devido a regra de negócio atual. Caso a nota fiscal já esteja com o status de finalizada no Sefaz (Secretaria de Estado da Fazenda), não é possível inutilizar a nota, devendo a mesma ser cancelada no sistema e não inutilizada. Entretanto, para os casos em que a nota ainda não estiver finalizada no Sefaz, a mesma pode ser inutilizada no sistema.

Para isso ser possível é necessário que o chatbot questione o usuário se a nota fiscal já está finalizada no Sefaz, ao identificar que a intenção do mesmo é a de “#ajuda\_inutilizar\_nota”. Posteriormente, quando o usuário responder, é necessário que o agente detecte quando o usuário esta confirmando a pergunta, ou quando o usuário está negando. Devido a essa necessidade, foram criadas as intenções “#resposta\_sim” e “#resposta\_nao” que tornam possível a abordagem citada, pois o chatbot é treinado para detectar quando a resposta do usuário for um sim ou um não. Essas intenções são consideradas de controle e utilizadas posteriormente no diálogo em mais intenções que necessitem de uma confirmação do usuário.

## Criando o fluxo do diálogo

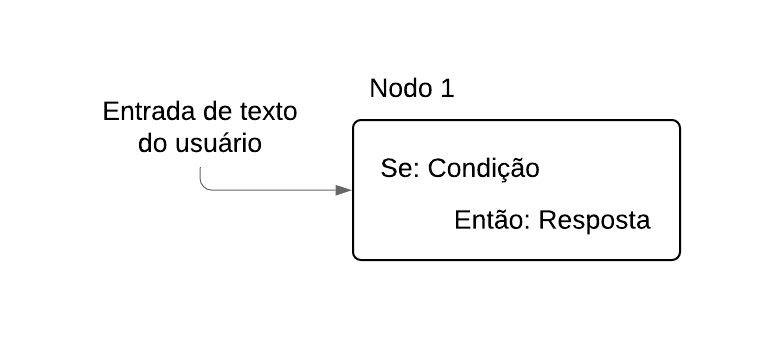
O terceiro componente presente no Watson *Assistant* é o diálogo, responsável por interligar as intenções e as entidades mapeadas e criadas até este ponto da construção do assistente virtual. A etapa do diálogo dentro do Watson *Assistant* consiste em estruturar um fluxo para a conversa dentro da ferramenta, para fornecer uma resposta as entradas de texto do usuário dependendo da intenção que a ferramenta detecta. Para construção deste diálogo o *framework* dispõe de uma ferramenta gráfica on-line, local em que é possível construir o diálogo. A qualidade desta etapa, assim como as capacidades de resposta do *chatbot* estão altamente relacionadas com a quantidade e qualidade das intenções mapeadas na seção anterior.

Neste trabalho está sendo utilizada a versão *Lite (*leve) do serviço do Watson *Assistant,* com isso existem algumas limitações que se aplicam a criação do diálogo. Uma delas é a limitação de 10.000 mensagens por mês, em que cada chamada para a API conta como uma mensagem. Outra redução desta versão é o limite de 5 *skills* de conversação, com no máximo 100 nodos de diálogo em cada uma. Também não existe um controle de versões sobre as *skills* criadas. Os históricos das conversas com os usuários, que podem ser utilizados para treinamento têm uma duração de 7 dias nesta versão da plataforma. O serviço do Watson *Assistant,* nesta versão,não pode ficar mais de 30 dias sem atividade que é deletado.

O diálogo na ferramenta do Watson *Assistant* é composto de uma ramificação de um ou mais nodos, cada um contendo, no mínimo, uma condição e uma resposta. Um nodo dentro do diálogo é um fluxo de conversação que define como o agente responde quando reconhece as intenções e entidades previamente definidas. Com o editor de diálogos da ferramenta do Watson *Assistant* é possível criar conversas com os usuários, fornecendo respostas com base nas intenções e entidades que são reconhecidas nas entradas de texto. A Figura 10 apresenta um nodo de diálogo, com uma representação básica da sua estrutura.

Como pode ser observado na Figura 10 um nodo de diálogo possui uma condição e uma resposta. A condição caracteriza as informações que devem estar presentes no texto que o usuário forneceu, para que o nodo em questão seja acionado. Essas informações geralmente são as intenções previamente mapeadas, mas elas também podem ser um tipo de entidade, um valor de entidade, o valor de uma variável no contexto ou alguma condição programada. Já a resposta, como o próprio nome já diz, é a frase que o serviço retorna para responder ao usuário. A resposta é um texto de retorno, mas pode ser configurada para exibir uma imagem, uma lista de opções ou para acionar ações previamente programadas.

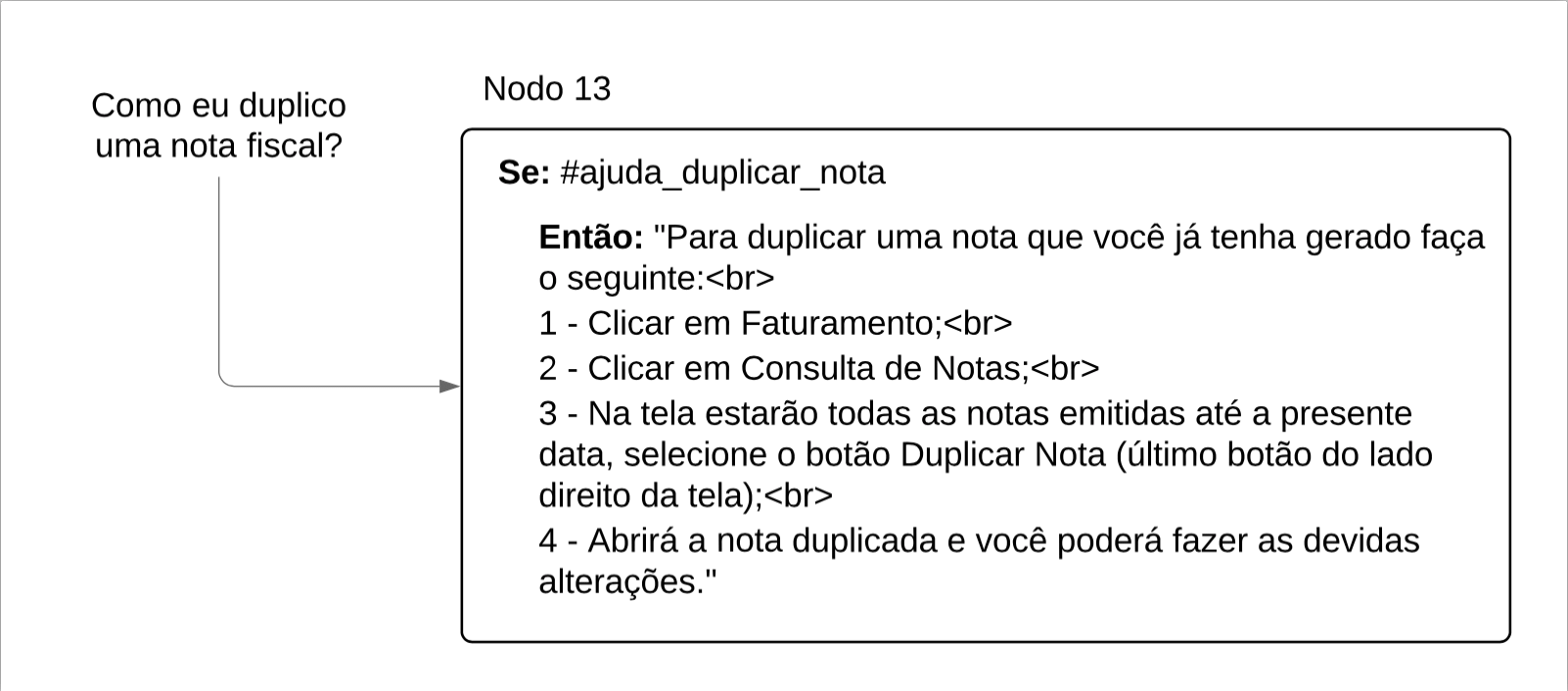
Figura 10 – Estrutura básica do nodo de diálogo



Fonte: do autor.

Um nodo único com condição e resposta simples e direta, pode ser configurado para manipular as solicitações dos usuários. Portanto, para as intenções dos usuários que não necessitam de mais dados, e devem fornecer ao usuário uma resposta simples e direta, foram configurados nodos simples com condição e resposta dentro da ferramenta. Como pode ser observado na Figura 11, com a representação do nodo da intenção “#ajuda\_duplicar\_nota”. Assim, o nodo sempre será acionado quando o processamento de linguagem natural do serviço detectar que a sentença fornecida pelo usuário contém a intenção de “#ajuda\_duplicar\_nota”.

Figura 11 – Nodo da intenção #ajuda\_duplicar\_nota

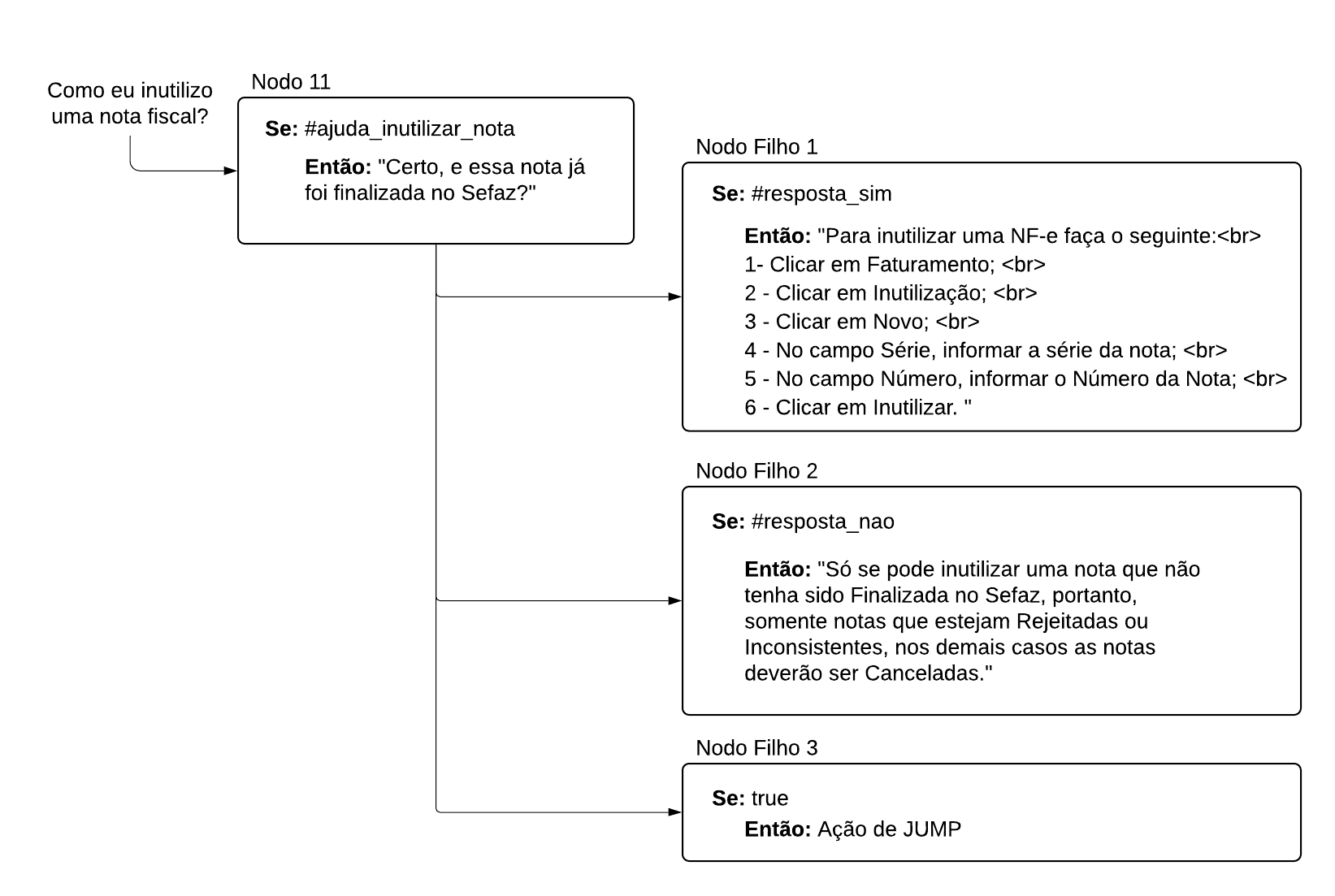


Fonte: do autor.

Como pode ser observado na Figura 11, a resposta fornecida para o usuário contém carácteres HTML no final das frases, neste caso o “<br>”. Essa abordagem foi necessária para formatar a resposta corretamente na tela do usuário final, pois a resposta do serviço é retornada em formato de texto via API, sem formatação. Então, para ser possível exibir corretamente as respostas para os usuários é utilizado HTML para formatar todas as respostas que necessitam de tratamento especial ou formatação de texto, como quebra de linha e negrito.

Um nó simples com condição e resposta atende a vários tipos de perguntas, porém para as perguntas que necessitam de mais informações antes de fornecer a resposta, foi necessário a utilização de outra técnica. No serviço existe a possibilidade da criação de nodos filhos, ligados com um nodo específico. Os nodos filhos solicitam a informação complementar para realizar uma ação, quando o nodo pai detecta a intenção correta do usuário. Um exemplo em que esta técnica foi necessária foi na intenção “#ajuda\_inutilizar\_nota”, que conforme descrito na seção anterior possui uma regra especifica. Com isso, a configuração dos nodos para essa intenção dentro do diálogo foi mapeada conforme a Figura 12.

Figura 12 – Nodos da intenção #ajuda\_inutilizar\_nota



Fonte: do autor.

Na Figura 12, primeiro o processamento de linguagem natural do serviço identifica a intenção de inutilizar a nota no texto do usuário, então faz a pergunta conforme o nodo 11 da Figura 12. Então o serviço fica esperando a próxima entrada do usuário, posicionado no início do nível dos nodos filhos, ou seja, esperando o texto do usuário para validar os nodos filhos. No caso desta intenção, o serviço verifica se a resposta do usuário remete a um sim ou a um não, e nestes dois casos fornece a resposta adequada. O nodo filho 3 possui a condição chamada de *true (*verdadeira) que é uma condição especial que sempre aciona o nodo caso a validação visite o mesmo.

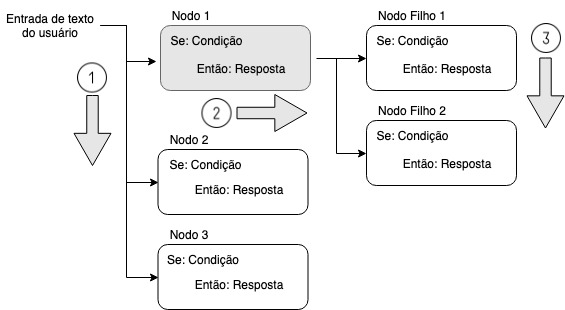
A condição especial *true* é adicionada no final da validação dos nodos filhos para sempre ser acionada quando nenhuma das condições dos primeiros nodos filhos for satisfeita. Caso isso ocorra, o serviço pode repetir a pergunta do nodo pai ao usuário explicando que não foi possível identificar corretamente a informação que necessita. Após isso, o serviço realiza a ação de *jump (*pular), que consiste em se posicionar novamente em outro ponto do diálogo para validar novamente a condição dos nodos em ordem. Desse modo é criado um *loop* (laço) com os nodos, pois o serviço faz a pergunta e espera a resposta até o usuário responder o complemento de informação necessário para fornecer a resposta correta.

Os diálogos criados dentro do Watson *Assistant* são processados como uma árvore, iniciando no primeiro nodo, ou seja, no topo da árvore. À medida que o serviço vai descendo na árvore ele avalia as condições dos nodos, para acionar aquele em que a condição seja atendida. Posteriormente o serviço acessa o nodo que satisfaz a condição para verificar a ação ou resposta, pesquisando também se existe algum nodo filho para validar, utilizando a mesma ordem e regra de validação dos nodos principais. Essa lógica é seguida do primeiro ao último nodo da ramificação que o serviço está seguindo. A Figura 13 ilustra a árvore do diálogo existente no Watson *Assistant*, com a ordem em que os nodos são validados.

Na Figura 13, as setas com a numeração indicam a ordem em que a árvore dos nodos é processada. Assim, seguindo a seta de número 1, é possível verificar que os nodos 1, 2 e 3 que estão no nível mais alto da árvore são validados primeiramente e em ordem. A seta de número 2 indica que a condição do nodo 1 foi avaliada como positiva para a entrada do usuário, com isso, a resposta ou ação presente no nodo 1 será retornada pelo serviço e os nodos subsequentes de número 2 e 3 não serão processados. Como o nodo 1 possuí nodos filhos os mesmos também serão processados pelo serviço seguindo a mesma regra e ordem dos nodos no nível mais alto da árvore. Ao validar os nodos filhos e fornecer uma resposta ou realizar uma ação o serviço finaliza o fluxo do diálogo em questão.

Como pode ser observando na Figura 13 a ordem das ramificações do diálogo é importante, porque as condições dos nodos presentes no mesmo são avaliadas em ordem. Quando um nodo avaliado na árvore contém uma condição em que a entrada do usuário se encaixe, o nodo é acionado e qualquer nodo que venha depois dele na árvore, que não sejam seus filhos, não é mais ativado. Levando em consideração esse comportamento do diálogo, a ordem em que os nodos são configurados dentro do diálogo tem grande interferência nas respostas que o chatbot fornecerá e por isso deve ser levada em conta na organização no diálogo.

Figura 13 – Validação dos nodos na árvore de diálogo



Fonte: do autor.

O primeiro nodo do diálogo configurado para o *chatbot* deste trabalho é um nodo com a condição especial chamada de *welcome* (boas vindas). Essa condição especial é avaliada como *true* sempre e somente quando o diálogo inicia e é invocada sem nenhum texto ser fornecido. Em todas as outras rodadas do diálogo subsequentes a primeira, esse nodo é avaliado como *false* (falso)*.* Esse nodo é utilizado no diálogo para saudar o usuário e apresentar o *chatbot*, como neste caso, com a frase “Olá. Sou o chatbot da Safetech. Como posso te ajudar?”. Assim, o orquestrador, que será detalhado na próxima seção, inicia o diálogo enviando uma requisição sem texto para o serviço do *Assistant*, para que o mesmo retorne a saudação como resposta.

Após o nodo inicial, os nodos do atendente criado neste trabalho foram organizados seguindo uma regra pré-determinada, levando em consideração a lógica em que os nodos do diálogo são processados. Portanto, as intenções que foram mapeadas com base nas dúvidas mais frequentes e receberam o prefixo de “ajuda” nas intenções, foram incluídos no topo da árvore. Como são dúvidas que já estão mapeadas e com as respostas fornecidas, devem ser as primeiras que o sistema valida. Após essas intenções de ajuda, foi posicionada a intenção de “#ajuda\_geral”, que faz a busca nos documentos quando identifica que é uma dúvida não mapeada. O objetivo desta abordagem é pesquisar a dúvida do usuário nos documentos.

Após esta intenção de ajuda geral foram inseridos os nodos para todas as intenções mapeadas como gerais, exceto as que foram criadas somente para controle. Com isso, o *chatbot* valida primeiro os nodos com as intenções de ajuda e dúvidas sobre o produto do sistema, atribuindo uma prioridade maior aos nodos que tratam do sistema e das regras do mesmo, devido a essas intenções estarem no topo da árvore. As intenções gerais que foram mapeadas para melhorar o fluxo da conversa ficam nos nodos finais da árvore, para não serem acionados com frequência, pois o sistema poderia fornecer respostas erradas caso identificasse a intenção do usuário em alguma delas, que por serem gerais, tem mais chance de corresponder com a sentença do usuário.

O último nodo do diálogo é um nodo que possui outra condição especial existente no serviço. Essa condição é chamada de *anything\_else* (Qualquer outra coisa) e sempre é validada como *true* caso o processamento dos nodos de diálogo chegue até ela, por isso a mesma é posicionada no final do diálogo. Dessa forma, sempre que todos os nodos do diálogo forem processados e nenhum tiver a condição que corresponda com a entrada de texto do usuário, esse nodo será acionado. Quando ativado, esse nodo tem o papel de retornar uma resposta para o usuário indicando que não foi possível entender o que ele deseja, e se é possível reformular a frase. Um exemplo de sentença que esse nodo retorna é “Eu não entendi. Você pode tentar reformular a frase?”. Com isso, o usuário fornece uma nova entrada de texto que será processada novamente na árvore de diálogo.

Um nodo dentro do diálogo do serviço é composto por 4 componentes principais que podem ser configurados dentro da ferramenta web do Watson *Assistant.* O primeiro deles é a condição, aonde se define o que o *framework* precisa identificar para acionar o nodo. O segundo componente é opcional e oferece a possibilidade de edição no contexto do diálogo, como por exemplo, atribuir o valor a uma variável. O próximo componente é a resposta que será fornecida pelo nodo quando o mesmo for acionado. Por último, existe a ação que será tomada após a resposta configurada ser fornecida ao usuário.

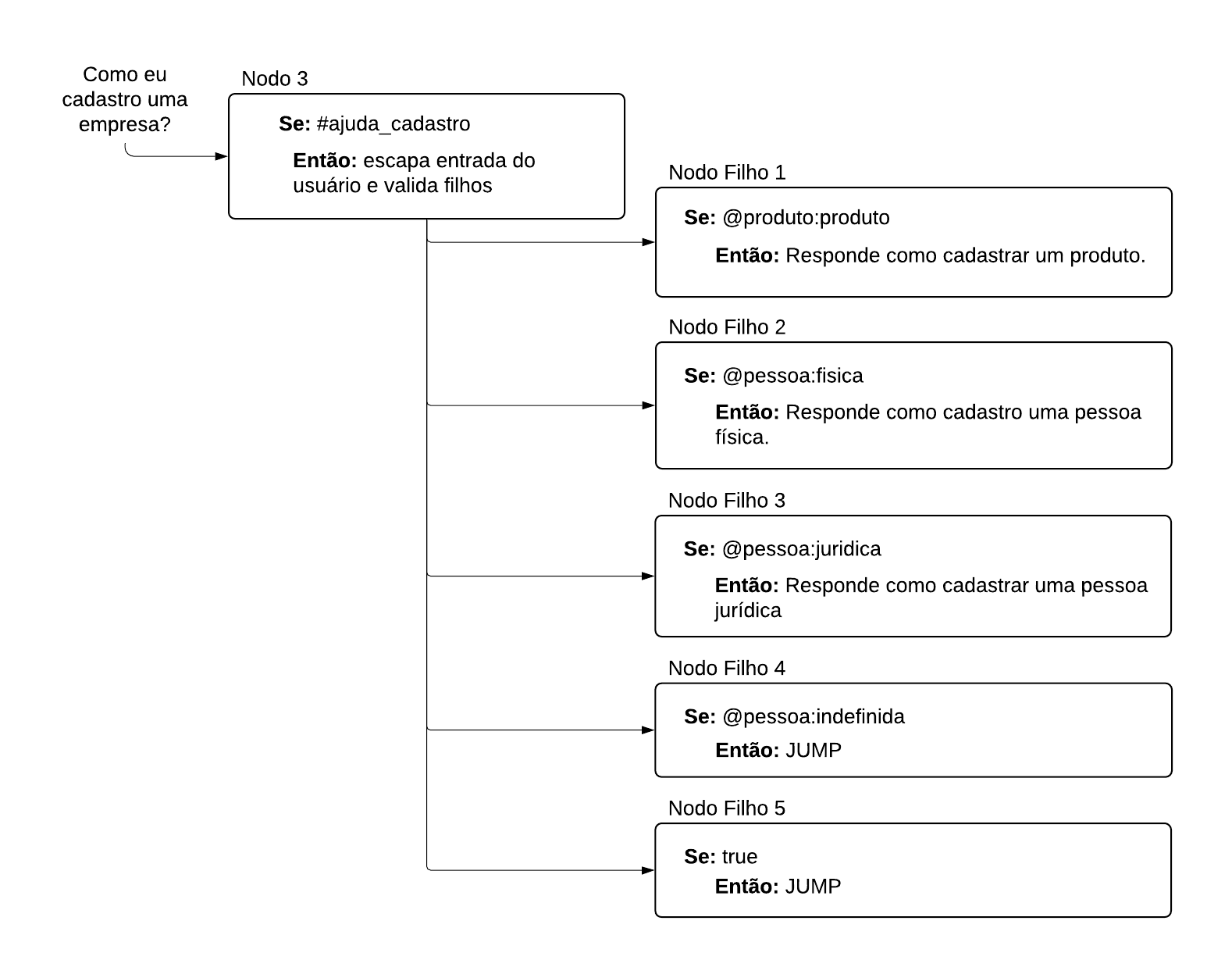
A configuração da condição dentro do nodo serve para definir quando o mesmo será utilizado dentro da conversa. Para compor uma condição se dispõe de algumas opções dentro do serviço, que podem ser combinadas no momento da definição da condição. Dentre as opções existentes temos: variáveis de contexto, entidades, valores de entidades, intenções ou condições especiais. Essas opções já existem dentro do serviço por terem sido definidas previamente ou por serem padrões do *framework*. Dessa maneira, utilizando uma opção ou uma combinação de opções é possível definir a situação exata em que o nodo deve ser acessado.

Sempre que um nodo é acessado é possível alterar as variáveis existentes no contexto da conversa, ou verificar o valor que as mesmas possuem para utilizar nas condições. O contexto mantém todas as informações sobre o diálogo que o usuário estabelece com o serviço, como os nodos acessados, a quantidade de turnos da conversa e as variáveis definidas pelo desenvolvedor. A variável com a quantidade de turnos da conversa é um contador incrementado automaticamente, que o *Assistant* utiliza para identificar em que turno a conversa está. Através do contexto é possível armazenar informações sobre os nodos acessados, que podem ser utilizados para complementar uma condição ou uma resposta. Sempre que um nodo é acessado pelo *framework* é possível buscar ou alterar o valor de uma variável no contexto. O contexto também é utilizado pelo serviço para identificar separadamente os diálogos que estão ocorrendo ao mesmo tempo no *chatbot*.

Além das variáveis de contexto é possível utilizar as intenções ou entidades em uma condição formulada para um nodo. Para utilizar uma intenção como condição, basta informar o nome definido para a intenção na condição que será imposta para o nodo ser acionado. Um exemplo é a intenção “#ajuda\_imprimir\_danfe”, que se informada diretamente na condição de um nodo, o mesmo será acionado sempre que o processamento de linguagem natural identificar que o usuário deseja imprimir uma Danfe (Documento Auxiliar da Nota Fiscal Eletrônica). A utilização das entidades na condição do nodo funciona de maneira semelhante a intenção, com a diferença de que as entidades podem ser utilizadas de duas maneiras diferentes dentro da condição.

As entidades dentro de uma condição podem ser utilizadas de maneira geral ou específica. De maneira geral, o serviço analisa a sentença do usuário para verificar se a entidade esta presente no texto. De forma específica ele valida se um valor exato definido para uma entidade está presente. Um exemplo em que foram utilizadas entidades nas condições dos nodos deste trabalho foi na definição das condições dos nodos filhos da intenção “#ajuda\_cadastro”. Neste caso, o nodo pai mapeia a intenção de que o usuário deseja cadastrar algo no sistema, e as entidades utilizadas nas condições dos nodos filhos verificam qual cadastro o usuário deseja realizar. A estrutura dos nodos desta intenção está representada na Figura 14.

Figura 14 – Estrutura de diálogo do nodo da intenção #ajuda\_cadastro.



Fonte: do autor.

Como pode ser observado na Figura 14, o nodo 3 é acionado quando a intenção do usuário é de cadastrar algo no sistema. Para ser possível definir qual cadastro que o usuário deseja realizar foram utilizadas as entidades mapeadas anteriormente. O nodo pai (Nodo 3) foi configurado para quando a condição for atendida ele “escapar” diretamente para a validação dos nodos filhos, sem fornecer nenhuma resposta. Os nodos filhos 1, 2 e 3 verificam na sentença do usuário uma entidade com valor específico. No caso da frase do usuário que foi fornecida, o nodo filho 3 tem a condição atendida porque o sistema identifica que o usuário deseja realizar um cadastro, que é de uma pessoa, e especificamente, jurídica.

As entidades são definidas com o caractere “@” antes do nome da mesma quando se deseja referenciar o valor da entidade dentro do serviço. Quando é necessário verificar um valor singular de uma entidade é utilizado o caractere “:” seguido do valor específico para a entidade. Como no caso da sentença da Figura 14, a condição do nodo filho acionado é “@pessoa:jurídica”. Assim, o Watson *Assistant* busca a entidade pessoa na frase do usuário, com o valor de pessoa jurídica. O nodo filho 4 foi criado para desambiguação das entidades e será detalhado nas próximas seções. O nodo 5 foi adicionado com a condição especial de *true*, para questionar o usuário novamente quando não identificar a informação necessária para fornecer a resposta.

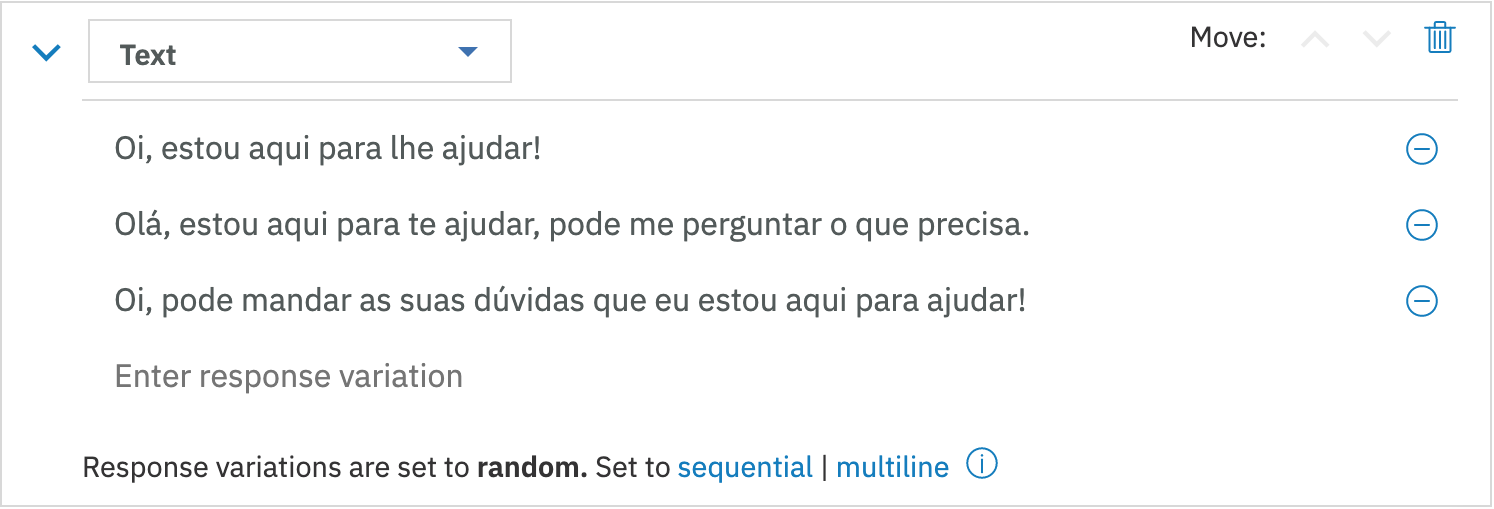
As respostas que um nodo pode fornecer quando a sua condição é satisfeita também possuem algumas variações e opções. Os tipos de resposta disponíveis atualmente no serviço são: texto, opção, pause e imagem. No tipo de resposta opção, são fornecidas respostas fixas para o usuário clicar na opção desejada ou indicar por texto qual a opção escolhida. O tipo pause serve para indicar ao *software* orquestrador que deve ser inserida uma pause nas mensagens, com o objetivo de melhorar a experiência do usuário. O tipo imagem retorna uma imagem para o usuário. Os tipos opção, pause e imagem não foram utilizados neste trabalho.

O tipo de resposta texto foi o utilizado em todas as respostas dos nodos deste trabalho, pois fornece o necessário para responder ao usuário corretamente. Além disso, utilizando as respostas sempre com o formato de texto, existe a necessidade do tratamento de somente um tipo de resposta no *software* orquestrador. Ao definir a resposta como do tipo texto, existe a opção da mesma ser exibida de 3 maneiras diferentes, como sequencial, randômica ou multilinhas. Desta maneira a resposta sempre será um texto e dentro de um nodo é possível informar variações de resposta para o diálogo acontecer de uma maneira mais natural, evitando as repetições.

No caso das respostas do tipo sequencial, todas as cadastradas como possíveis para o nodo são apresentadas para o usuário em ordem, conforme o nodo for sendo acessado. Para o tipo randômico, também são definidas mais respostas para o nodo, porém a exibição para o usuário acontece de forma aleatória sempre que o nodo é acessado. O tipo multilinhas pode ser utilizado nas situações em que o texto da resposta se torna grande, sendo necessário a quebra da resposta em mais de uma linha. A Figura 15 apresenta as variações de resposta que são apresentadas para o usuário de forma randômica, sempre que a intenção “#geral\_saudação” é identificada.

Através desta configuração de respostas do nodo, o diálogo fica mais natural para os usuários que acessam a ferramenta. As Figuras 16 e 17 mostram a variação da resposta do *chatbot* quando o usuário envia uma saudação para o mesmo. A configuração randomizada foi selecionada dentro do tipo de resposta do nodo e o serviço controla a resposta que deve ser enviada, não necessitando tratamento no orquestrador ou na aplicação de *front-end*. Nas figuras o usuário fornece a mesma entrada de texto e recebe respostas diferentes aleatoriamente.

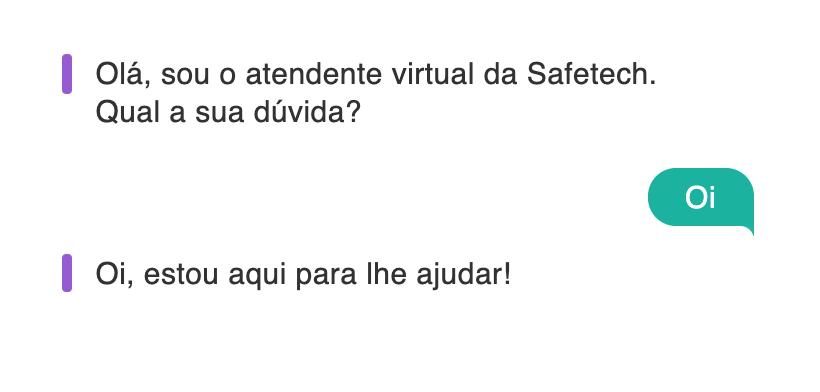
Figura 15 – Respostas do nodo #geral\_saudação.



Fonte: (IBM Watson Assistant, 2019).

O último componente de um nodo são as ações, que definem o que fazer após fornecer a resposta para o usuário. Dentre as opções disponíveis atualmente no *framework* estão: aguardar a resposta do usuário, ignorar a resposta do usuário ou pular para outro nodo do diálogo. A opção aguardar a entrada do usuário é a opção padrão e mais utilizada no *chatbot* deste trabalho. Pois o nodo fornece a resposta para o usuário, induzindo a próxima resposta, e espera a próxima entrada de texto do mesmo para finalizar a resposta, como no caso da intenção #ajuda\_inutilizar\_nota em que o usuário precisa responder sim ou não. Ou quando o nodo fornece a resposta diretamente para o usuário e fica esperando a próxima sentença para validar os nodos da árvore novamente.

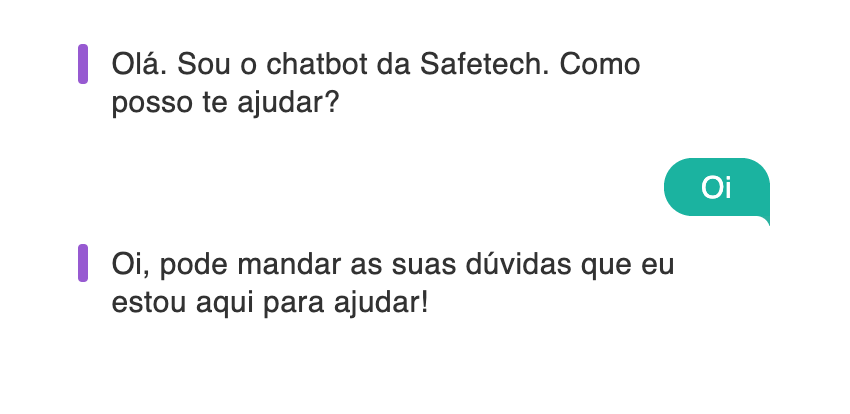
Figura 16 – Resposta 1 do *chatbot* à saudação do usuário.



Fonte: do autor.

A opção de ignorar a resposta do usuário é utilizada nos casos como da intenção #ajuda\_cadastro em que a resposta do usuário será processada pelos nodos filhos e o nodo pai não fornece nenhuma resposta para o usuário, devido as respostas estarem nos nodos filhos. Essa opção fica disponível somente quando o nodo possui filhos. No caso da opção pular, já mencionada anteriormente, a mesma existe para ser possível pular de um nodo para outro diferente, localizado em qualquer local dentro da árvore. Neste trabalho é utilizada para criar os *loops* nos nodos filhos quando a informação do usuário não mapear para nenhuma condição definida nos nodos, forçando o usuário a reformular a sua frase ou responder corretamente, como no exemplo da Figura 12, condição #ajuda\_inutilizar\_nota.

Figura 17 – Resposta 2 do *chatbot* a saudação do usuário.



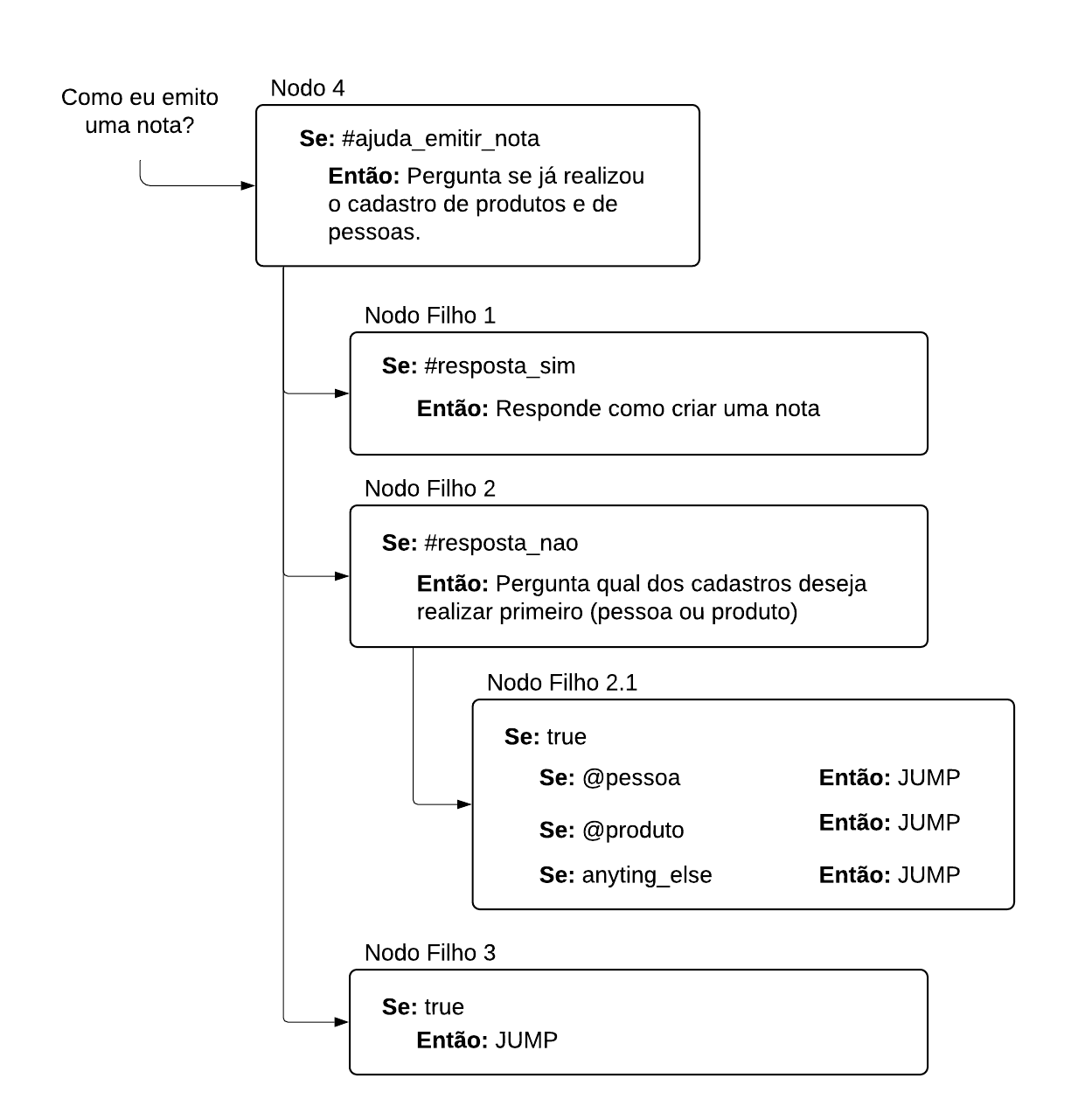
Fonte: do autor.

Durante o desenvolvimento do diálogo na ferramenta do Watson *Assistant*, foi identificado que a intenção de “#ajuda\_emitir\_nota”, que trata quando o usuário deseja criar uma nova nota fiscal no sistema, estava com um texto de resposta muito grande para exibição dentro da conversa. Com isso, para melhorar o fluxo do diálogo, foi necessária a criação de alguns nodos filhos para reduzir as informações antes de apresentar para o usuário. No caso desta intenção, para emitir uma nota fiscal é necessário que o cliente e os produtos estejam previamente cadastrados no sistema. Assim, optou-se pela abordagem de questionar o usuário se todos os cadastros foram realizados antes de fornecer a resposta de como emitir a nota fiscal.

Todas as opções disponíveis para criar as condições que podem ser utilizadas para definir quando um nodo deve ser acionado, também podem ser utilizadas dentro das respostas de um nodo. Esse tipo de resposta é chamado de *multiple responses* (resposta condicional) dentro do *framework.* Ao utilizar essa técnica é possível fornecer respostas distintas para uma pergunta do usuário, com base no valor de alguma variável ou entidade na sentença fornecida. Desta maneira um único nodo pode prover diferentes respostas de acordo com uma condição, o que torna possível simplificar e reduzir a árvore de diálogo quando utilizada esta configuração, pois um único nodo pode fornecer diferentes respostas.

A Figura 18 ilustra o fluxo do diálogo criado para a intenção de emitir uma nova nota fiscal. Quando a frase de exemplo “Como eu emito uma nota fiscal?” é fornecida para o serviço, o nodo 4 é acionado para questionar o usuário se o mesmo já efetuou os cadastros de produto e pessoas. Caso a resposta do usuário seja uma confirmação, o *chatbot* fornece a resposta com os passos necessários para emitir uma nota fiscal. Quando a resposta do usuário mapeia para uma negação, o nodo filho 2 é acionado questionando o usuário qual cadastro ele não realizou, ou qual gostaria de realizar primeiro. Após questionar o usuário, o serviço aguarda a resposta do mesmo para validar os nodos filho do nodo atual.

Figura 18 – Fluxo de diálogo da intenção #ajuda\_emitir\_nota.



Fonte: do autor.

O nodo filho 2.1, filho do nodo filho 2, implementa o tipo de resposta condicional, buscando identificar a presença de algum valor das entidades pessoa ou produto na sentença do usuário. Devido as respostas de como cadastrar pessoas ou produtos já estarem descritas nas respostas dos seus respectivos nodos, não é necessário duplicá-las neste nodo filho, mas sim direcionar para os respectivos nodos. Para isso, se realiza a ação de *jump* direto para os nodos de resposta do cadastro de pessoa ou de produtos. O último teste da condição das respostas do nodo filho 2.1 utiliza a condição especial *anyting\_else* para voltar à pergunta do nodo filho 2 quando não identificar nenhuma das entidades necessárias na sentença do usuário.

O último nodo da Figura 18 também realiza uma ação de *jump,* porém para questionar o usuário novamente quando não identificar se o mesmo já realizou os cadastros ou não. Em síntese, esse fluxo da intenção “#ajuda\_emitir\_nota” foi criado com o objetivo de interagir mais com os usuários quando for detectado que o mesmo deseja emitir uma nota. Além disso, a resposta final para a pergunta do usuário é reduzida, devido a uma parte dela ter sido separada em outros nodos. Esse fluxo de diálogo mostra diversos recursos da ferramenta do Watson *Assistant* sendo combinados para fornecer uma resposta para uma intenção do usuário.

Nesta seção é possível observar diversos recursos disponíveis dentro da ferramenta *on-line* do Watson *Assistant* para criação de diálogos. As intenções dos usuários que podem ser classificadas como mais simples, com perguntas e respostas bem definidas são facilmente mapeadas e atendidas com o uso desta ferramenta. Para as intenções dos usuários que possuem uma complexidade maior ou que necessitam de mais informações para fornecer uma resposta correta, também são atendidos através do *framework*, porém requerem um trabalho maior de estruturação e configuração manual dos nodos na árvore do diálogo.

## Desenvolvimento do Software orquestrador

Para integração entre os serviços disponíveis no ambiente on-line da IBM utilizados neste trabalho, foi necessário o desenvolvimento de duas camadas de software. Uma delas, responsável pela comunicação com as APIs dos serviços da IBM, desempenha o *back-end* da aplicação nomeada de *orquestrador*. A outra camada, responsável pelo *front-end*, trata da integração com o usuário final, construindo a interface que exibe as mensagens retornadas pelo orquestrador para a leitura do usuário. A mesma também precisa enviar as mensagens fornecidas pelo usuário para o orquestrador.

Como já mencionado anteriormente, o *chatbot* foi construído para ser utilizado via *web* dentro do sistema de emissão de notas fiscais eletrônicas da empresa. O *front-end* da aplicação está dividido em 3 arquivos, necessários para a criação da interface. Os arquivos são: “index.html”, “style.css” e “script.js”. Esta camada da aplicação pode ser considerada simples devido ao fato de conter poucos elementos em tela. Pois para atender aos usuários o sistema necessita acima de tudo que a interface receba as mensagens do usuário e envie para o *back-end*. Em seguida aguarda o recebimento da mensagem do orquestrador, para então exibir a resposta formatada para o usuário.

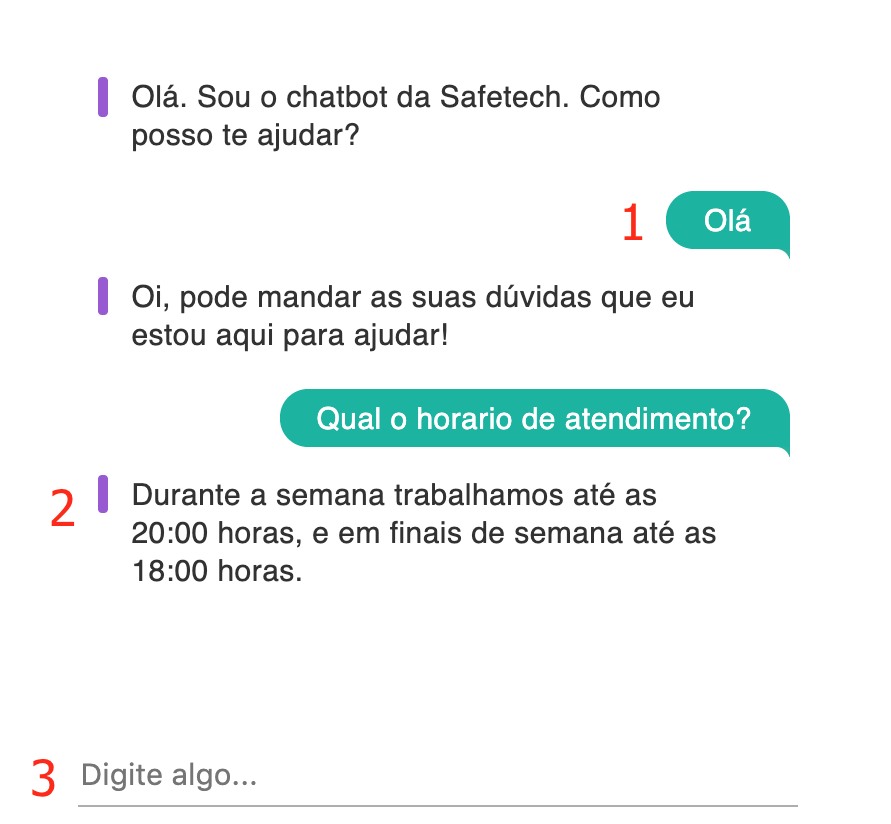
O arquivo “index.html” recebe a extensão “.html” por ser escrito na linguagem *web* HTML. Esse arquivo contém os códigos que criam os elementos da interface que é apresentada ao usuário. O arquivo “style.css” foi criado na linguagem *web* CSS e formata os elementos em HTML adicionados na tela, adicionando cores, definindo a fonte e o tamanho dos elementos. O arquivo “style.css”, assim como o arquivo “script.js” foram adicionados na página através do arquivo “index.html”. O objetivo desta interface é adicionar nesta tela somente os elementos necessários para a interação com o usuário, com foco maior no conteúdo do diálogo e das respostas.

Para o controle das ações que a tela deve executar para ser possível o diálogo, foi criado o arquivo “script.js”, na linguagem *javascript.* Este arquivo é responsável por monitorar a página. Assim que o usuário escrever uma mensagem dentro da caixa de texto disponível e digitar a tecla “*enter*”, o texto da mensagem é enviado para o software orquestrador por meio de uma requisição *post.* O *front-end* envia uma mensagem para o orquestrador e aguarda o retorno do mesmo com a resposta, para então apresentá-la ao usuário. Sempre que uma mensagem é recebida ou enviada para o *back-end* ela é adicionada na tela pelo *script* para ser mantida a conversa.

Essas mensagens são apresentadas para o usuário com uma formatação diferente, diferenciando as mensagens do Watson (atendente) e do usuário (cliente). O *script* sempre adiciona na tela as mensagem que recebe do *back-end*, na área da página das mensagens que já foram trocadas, rolando para a última mensagem na tela para ser possível a visualização por parte do usuário. O código do *script* também mantém uma variável para armazenar o contexto da conversa, que é retornado pelo serviço do Watson *Assistant* e deve ser mantido durante toda a conversa. Sempre que a página do *chatbot* é acessada, o *front-end* envia uma mensagem sem conteúdo para o orquestrador, para receber a saudação do atendente devido a configuração do diálogo no Watson *Assistant*.

A interface que resulta da codificação destes arquivos pode ser visualizada na Figura 19. É possível observar que se trata de uma interface simples contendo somente as mensagens trocadas entra o usuário e o atendente, além de uma caixa de texto (identificada na Figura com o número 3) para que seja possível que o usuário envie as suas perguntas. As sentenças alinhadas a esquerda pertencem ao atendente criado neste trabalho e as da direita ao usuário que está conversando com o robô. O diálogo inicia com uma mensagem de saudação e apresentação do *chatbot,* que o *framework* do Watson envia devido a configuração do nodo *welcome* presente no inicio do diálogo. A mensagem identificada pelo número 1 é uma mensagem que foi enviada pelo usuário, enquanto a mensagem identificada pelo número 2 é uma resposta do *chatbot* ao usuário, a respeito de uma dúvida previamente mapeada e configurada.

Figura 19 – Interface do chatbot.



Fonte: do autor.

O desenvolvimento do software orquestrador inicia com a definição dos pacotes necessários para a execução deste projeto. O *Node.js* dispõe de um repositório com diversos módulos que podem ser utilizados dentro do projeto que estiver sendo criado. Existe um gerenciador destes pacotes, chamado de NPM (*Node Package Manager*) que é instalado junto com o *Node.js* e que obtém e gerencia esses módulos. Os módulos são bibliotecas de código em *javascript* que podem ser adicionadas ao projeto (NODE.JS NPM, 2019). Então, para este trabalho se fez necessária a utilização de 2 pacotes disponíveis no NPM, além da SDK do Watson que também é utilizada através de um pacote.

Os pacotes que foram necessários neste projeto são: *Express[[17]](#footnote-18)* e *Body Parser*[[18]](#footnote-19)*.* O primeiro é utilizado para criação de um servidor de *HTTP* que é responsável por receber as requisições realizadas pelos usuários no *front-end.* O *Body Parser* serve para converter o *body* (corpo) da requisição para vários formatos diferentes, para ser possível utilizá-lo dentro do software para troca de mensagens entre os serviços. No caso deste trabalho o formato em que o orquestrador se comunica com a aplicação será o *JSON,* por ser uma formatação leve para troca de dados, e compatível com todos os serviços da IBM utilizados. Sendo assim, com a utilização destes dois pacotes foi criada a estrutura necessária para processar as requisições ao orquestrador.

Além dos dois pacotes utilizados para criação do servidor, é empregado o pacote com as APIs do Watson da IBM, chamado de watson-developer-cloud[[19]](#footnote-20). Com a utilização desta biblioteca os serviços do Watson *Assistant*, Watson *Discovery* e Watson *Natural Language Understanding*, previamente criados e configurados no ambiente *web* da IBM, podem ser configurados para utilização. Todos esses serviços da IBM são acessados através de credencias que são geradas para o desenvolvedor quando os serviços são criados. Para utilização neste trabalho todos os serviços foram criados na versão *free* disponível.

O orquestrador consiste de um servidor preparado para requisições HTTP de *post* na rota “/conversation/” no endereço do servidor que estiver executando a aplicação. Uma visão do processo macro que essa rota executa está exposto na Figura 9, apresentada na introdução deste capítulo. Ao receber uma requisição o primeiro teste realizado pelo orquestrador é verificar se o texto da requisição está informado, caso não esteja, nenhum tratamento ou serviço adicional é invocado, somente o do Watson *Assistant* com o texto nulo, para receber a saudação. Somente na primeira interação com o usuário que o texto pode ser nulo, essa consistência é obtida na página do chat. Através deste comportamento o inicio do diálogo é configurado.

Assim como o diálogo que foi configurado dentro do *Assistant*, o orquestrador também demandou um fluxo de processamento organizado em sequência. Esta ordem foi necessária pois um dos objetivos do trabalho foi o de agregar mais de um dos serviços de processamento de linguagem natural da IBM na criação do *chatbot*. Desta maneira, foi definido que o primeiro serviço a ser utilizado é o NLU, que tem a possibilidade de enriquecer o contexto com informações. Após o NLU, o serviço do *Assistant* é utilizado para processar o texto dentro da lógica criada na árvore de diálogo, indicando se o último dos serviços, o *Discovery,* deve ser utilizado. Assim sendo, foi possível integrar todos os serviços neste trabalho com a utilização desta abordagem.

Quando existe um texto na requisição enviada para o orquestrador, que são todos os turnos de diálogo subsequentes ao primeiro, o primeiro passo efetuado é a configuração da chamada do serviço do NLU. Os parâmetros enviados para o NLU consistem no texto que deve ser analisado, juntamente com quais análises o usuário deseja efetuar. Para este atendente é solicitada a extração das entidades e das palavras chave que constarem no texto, com uma indicação para retornar também a emoção extraída de cada uma delas. Foi enviado como parâmetro também uma indicação do número máximo de duas intenções e duas palavras chave. Com esta configuração o serviço do Watson *Natural Language Understanding* é invocado pelo orquestrador.

A resposta enviada pelo NLU para o orquestrador é processada com o objetivo de encontrar o nome da pessoa na frase. Para isso, a resposta do serviço é analisada para verificar se alguma das entidades detectadas é uma pessoa, que na resposta do *framework* é identificada como uma entidade do tipo “*Person*”. Quando esta entidade é encontrada na resposta, ela é extraída e adicionada em uma variável no contexto com o nome de “*Person”* e com o valor extraído. Em suma, é realizada uma análise na frase do usuário para identificar se o nome dele foi informado e neste caso extrair e adicionar essa informação em uma variável do contexto. Desta maneira foi possível observar que o texto do usuário pode ser analisado em busca de informações específicas antes de ser enviado para o processamento da árvore de diálogo.

Depois de tratada a resposta do serviço do NLU, a sentença do usuário juntamente com o contexto, que pode ter sido enriquecido, são enviados ao Watson *Assistant*. O método responsável por estachamada tem o nome de “*message”* epara utilização do atendente deste trabalho são enviados 3 parâmetros: (i) o texto que o usuário digitou e (ii) o ID (identificador) do ambiente configurado na IBM, chamado de *workspace.* Por ultimo (iii) é necessário enviar também a variável com o contexto atual da conversa. A resposta retornada pelo serviço do Watson *Assistant* é enviada na integra pelo orquestrador para o *front-end*. Nesta resposta consta um campo que possui o texto que o serviço mapeou como sendo a resposta que possuí a maior correspondência com a pergunta enviada. Sendo assim, a aplicação extrai somente esse texto e atualiza no *chat* como sendo a resposta do atendente.

Esse retorno do *Assistant* contém diversas informações que o serviço obteve por meio do processamento de linguagem natural efetuado sobre a sentença. Esta resposta também contém o resultado da validação dos nodos da árvore de diálogo, retornando o nodo que foi acionado. Todos os nodos que foram percorridos para obter o resultado do processamento da árvore também são retornados dentro do contexto. Um exemplo do retorno enviado pelo *framework,* ao receber uma sentença, pode ser observado na Figura 20. Esse retorno corresponde ao resultado para o processamento da sentença “Qual o horário de atendimento?” e está formatado como JSON devido ao tratamento do orquestrador.

Figura 20 – Retorno do Watson *Assistant* ao orquestrador.



Fonte: do autor.

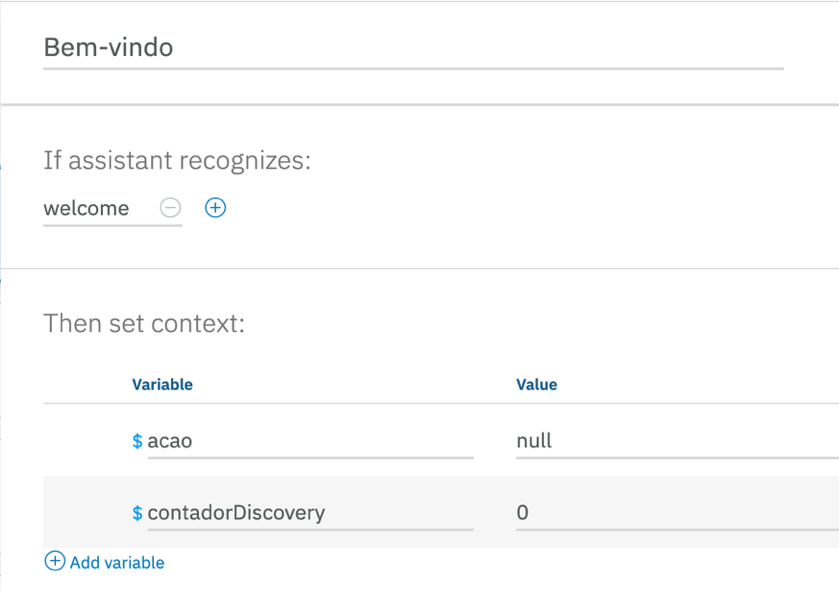
Por meio da Figura 20 é possível perceber que existe um *array* chamado de “*intents*” na linha 2 do arquivo, que retorna todas as intenções que o serviço identificou na sentença do usuário. No caso desta frase do exemplo a intenção mapeada foi a “geral\_horario\_atendimento” com uma confiança de 93%. Há também um *array* com as entidades que foram mapeadas, mas que neste exemplo não foram encontradas. Nas linhas 9 e 10 é possível observar que é retornada uma variável chamada de “*input*” com a resposta do nodo em texto. Nas linhas 12 à 26 é retornado um *array* contendo todos os tipos de resposta que o nodo enviou, caso seja configurada alguma resposta diferente de texto. O *array* com o contexto do diálogo também está presente no retorno, porém foi reduzido na figura, para ser detalhado posteriormente.

Quando uma resposta é retornada pelo *Assistant,* a primeira ação que o orquestrador realiza antes de enviar o retorno para o *front-end* é a de verificar se o contexto contém alguma ação indicada para ser executada. Ou seja, no *Assistant* foi configurada a regra para indicar quando uma ação deve ser executada e o orquestrador executa as ações. A implementação realizada neste trabalho trata duas ações que podem ser indicadas pelo *Assistant*: (i) a ação de encaminhar a conversa a um atendente real e (ii) a ação de realizar uma busca nos documentos do *Discovery*, pois a resposta não estava mapeada no diálogo. Em vista disso, o orquestrador foi configurado para analisar a resposta do *Assistant* antes de retornar a resposta.

Para ser possível analisar as variáveis do contexto que o *Assistant* retorna para o orquestrador é necessário gravá-las em algum momento durante o diálogo. O *Assistant* prove a possibilidade da alteração das variáveis de contexto quando um nodo é acessado dentro da árvore do diálogo. Com isso, foi possível definir as variáveis relevantes para as ações do orquestrador e manipulá-las dentro do fluxo do diálogo. A Figura 21 ilustra a condição do nodo “Bem-vindo”, primeiro nodo da árvore, que possui a condição especial de “*welcome*”, que é executada sempre e somente quando o diálogo inicia. Devido a essa configuração este nodo foi escolhido para resetar as variáveis de contexto utilizadas neste trabalho. Pode ser observado que a variável “ação” recebe o valor *null* (nulo) e a variável “contadorDiscovery” recebe o valor 0.

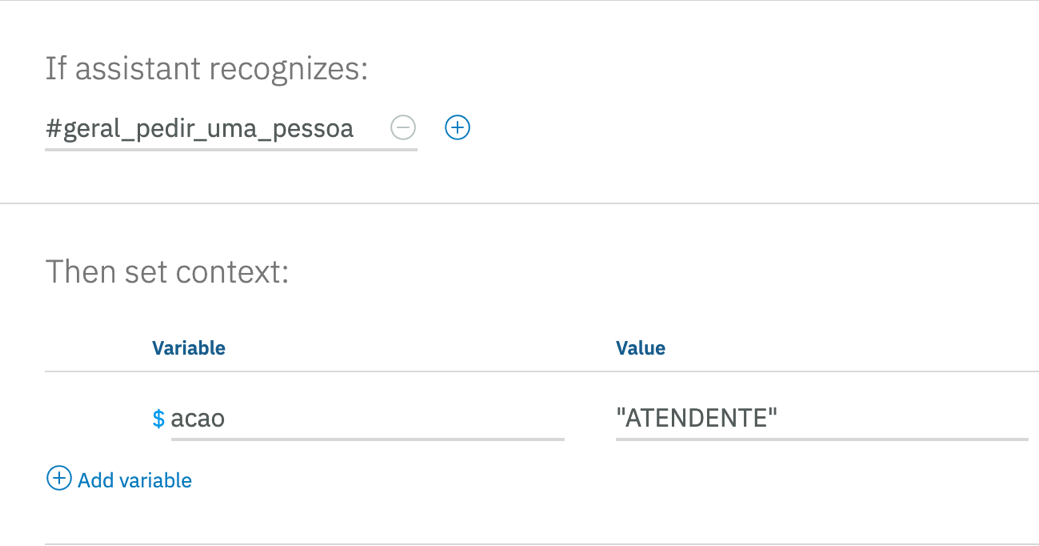
Desta maneira, sempre que o diálogo inicia as variáveis são inicializadas para controlar corretamente as ações necessárias. A partir do nodo inicial a variável “acao” passa a existir dentro do contexto do diálogo podendo ser utilizada pelo orquestrador. Quando um nodo que mapeia uma intenção que requer uma ação é acionado, a variável “acao” é manipulada pelo nodo para comunicar o orquestrador que existe uma ação a ser tomada antes de retornar a resposta ao usuário. A Figura 22 demonstra a alteração da variável de contexto “acao” quando a intenção do usuário é a de falar com um atendente real, na intenção “geral\_pedir\_uma\_pessoa”. Portanto, esta é lógica utilizada para indicar para o orquestrador quando uma ação deve ser tomada.

Figura 21 – Inicialização do contexto no nodo Bem Vindo.



Fonte: do autor.

Figura 22 – Alteração da variável de contexto ação.



Fonte: do autor.

Quando a resposta do *Assistant* chega ao orquestrador é executada uma função que analisa o contexto antes que enviar a resposta ao *front-end*. A função verifica se a variável “acao” possui algum valor dentro do contexto. Caso não exista, o processo é finalizado retornando a resposta do *Assistant*. Para os casos em que existe, a função verifica qual o seu valor para então definir o que deve ser realizado. As duas ações possíveis são: (i) atendente ou (ii) *discovery*. Sempre que a ação for para encaminhar a um atendente o orquestrador envia um e-mail para o responsável do suporte entrar em contato com o cliente e a resposta do nodo comunica o mesmo de que alguém entrará em contato para resolver o seu problema. A Figura 23 demostra o contexto de um retorno do *Assistant* em que a variável de contexto “acao” está gravada.

Na linha de código 43 da Figura 23 é possível observar que o contexto retornado pelo Watson Assistant indica que a conversa deve ser encaminhada a um atendente. Na linha 10 é possível observar a quantidade de turnos que foram realizados no diálogo, sendo que um turno é computado para cada par de pergunta e resposta. Das linhas 12 a 39 é retornada uma variável com todos os nodos que foram percorridos durante o diálogo, nos turnos anteriores até o atual. Desta forma o contexto armazena as variável necessárias para a integração entre os serviços utilizados no chatbot deste trabalho, além de conter informações importantes sobre o diálogo.

Figura 23 – Contexto do diálogo ao pedir um atendente real.



Fonte: do autor.

Nos casos em que a ação do *Assistant* indica que deve ser efetuada uma busca nos documentos, o orquestrador realiza um processo diferente. Esse processo consiste em inicializar o serviço do Watson *Discovery* e realizar uma *query* (consulta) nos documentos disponíveis no serviço. O serviço também possui uma parametrização para indicar em qual coleção de documentos deve ser feita a busca. Além de indicar a coleção, é enviado o texto do usuário não mapeado pelo *Assistant* na íntegra, juntamente com o parâmetro que indica que devem ser retornadas as passagens de texto de dentro dos documentos, que possuem a maior relevância com a pergunta do usuário. Também é limitado o número de resultados para 3 respostas.

Após configurados os parâmetros é executado o método “*query*” do Watson *Discovery*, para realizar a busca nos documentos. A resposta do *Discovery* vem na forma de um *array* (lista) com as três passagens com maior relevância com os parâmetros enviados para consulta. A função então extrai a resposta com maior pontuação e concatena no retorno que deve ser enviado ao *front-end*. Na versão inicial, não foi criado nenhum parâmetro para indicar o limite mínimo de pontuação das respostas que serão retornadas, retornando sempre a resposta com melhor pontuação.

Assim sendo, a versão inicial de consulta aos documentos do *Discovery* monta a *query*, executa e retorna o melhor resultado para o usuário. Inicialmente se optou pela abordagem de que quando o *Assistant* não conseguisse mapear a intenção do usuário para um problema já identificado, o texto que fosse encontrado nos documentos do *Discovery* poderia ajudar. O retorno do serviço, chamado de passagem de texto, é um trecho retirado de dentro de algum dos documentos, da parte com maior relação com a consulta enviada. Como esse retorno é de qualquer local do texto, não foi realizado nenhum tratamento antes de exibir ao usuário, somente é informado que foi realizada uma busca nos documentos, apresentando o resultado.

Quando nenhuma resposta for encontrada dentro dos documentos presentes no Watson *Discovery,* o orquestrador adiciona uma mensagem no texto que será retornado ao usuário. Essa mensagem indica que foi realizada uma busca nos documentos, mas que nada foi encontrado. Ao final deste processo a função do orquestrador limpa as variáveis de contexto, indicando que o *Discovery* já foi consultado e retorna a mensagem ao *front-end*. Mediante o exposto, foi possível integrar o serviço do Watson *Discovery* dentro do fluxo deste *chatbot*. Sendo viável realizar o controle através do orquestrador, com as regras de negócio dentro do *Assistant* para indicar quando o serviço deve ser chamado.

## Configuração do Discovery

A primeira etapa para utilização do serviço do Watson *Discovery* foi a da criação de uma coleção de documentos na ferramenta *web* da IBM. Uma coleção dentro do *framework* é um conjunto de documentos armazenados dentro do mesmo local, para consulta em conjunto. As coleções existem dentro do serviço para ser possível separar documentos que não tenham conteúdos semelhantes e não devem ser consultados ao mesmo tempo. No caso deste trabalho, foi criada uma coleção que armazena os documentos com conteúdo referente ao sistema. Como todos os documentos deste *chatbot* tratam do mesmo assunto, somente 1 coleção se faz necessária, para ser possível consultá-la e analisar todos os documentos relevantes em conjunto.

Para o atendente deste trabalho, foram selecionados inicialmente 3 arquivos para serem adicionados ao *framework*. Dentre os arquivos estão 2 manuais em formato *pdf* e um documento em formato *docx.* Os documentos em *pdf* são manuais de como utilizar o sistema de emissão de notas fiscais e cupons fiscais (NFC-e) da empresa com um passo-a-passo bem detalhado de como utilizar o sistema. Esses arquivos misturam imagens e texto para explicar ao usuário como utilizar a ferramenta e foram carregados no serviço para verificar se a consulta em suas informações seriam uma fonte útil de conteúdo para as respostas do *chatbot*.

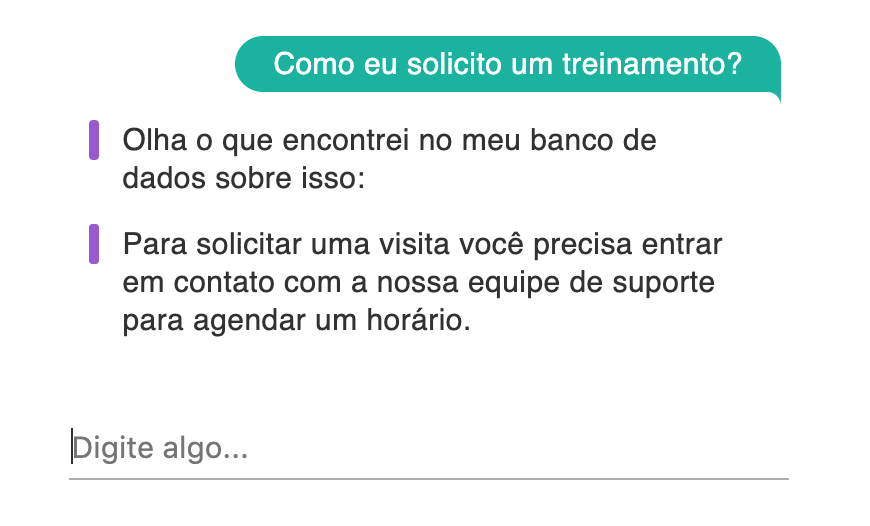
Além destes dois manuais em *pdf* foi utilizado também um documento em formato de texto. Esse documento possui somente texto como seu conteúdo e esta organizado na forma de uma FAQ, com perguntas e respostas definidas. Uma parte das perguntas presentes neste documento são as que não foram mapeadas para dentro do *Assistant,* por possuírem perguntas ou respostas com muito texto ou por não serem as mais frequentes. Foram adicionadas também as perguntas que foram utilizadas para mapeamento das intenções do *Assistant,* porém com respostas mais elaboradas. O objetivo desta abordagem foi o de criar um documento para servir como uma base de conhecimento do *chatbot* em que todas as respostas ou explicações referentes ao sistema fossem adicionadas.

Os arquivos utilizados neste trabalho que tiveram o upload realizado no Watson *Discovery*, foram todos adicionados ao serviço utilizando a configuração padrão. Após efetuar a carga dos documentos dentro do ambiente *web* da IBM a coleção do *Discovery* com estes arquivos fica pronta para executar *querys* (consultas) através da API. A ferramenta oferece a opção de realizar anotações nos documentos, utilizando alguns campos previamente disponíveis ou campos personalizados, utilizando as versões pagas. Neste trabalho não foi realizada nenhuma anotação nos documentos carregados no *Discovery*.

Depois de realizado o *upload (*carga) dos documentos definidos, o serviço realiza uma identificação e enriquecimento nos dados carregados automaticamente. A identificação padrão do serviço sempre extrai o texto dos documentos, para tornar possível que sejam realizadas consultas. O enriquecimento aplicado aos documentos, trata de extrair entidades, analisar o sentimento geral dos documentos, classificar as categorias e conceitos (assuntos) existentes. Esses 4 enriquecimentos sempre são aplicados ao texto dos documentos adicionados no *framework* automaticamente como uma forma de apresentar um resumo analítico dos dados ao usuário.

Na Figura 24 é possível observar uma interação do usuário com o *chatbot,* em que foi preciso consultar os documentos para retornar uma resposta. Nesta situação a pergunta do usuário foi “Como eu solicito um treinamento?” que é uma questão não mapeada pelo *Assistant* em nenhum nodo do diálogo. A resposta do *chatbot* nestes casos é composta de um texto que é retornado pelo *Assistant,* indicando que foi realizada uma busca nos documentos de suporte, exemplo “Olha o que encontrei no meu banco de dados sobre isso:”. Essa introdução é seguida da resposta encontrada na passagem dos documentos presentes na coleção do *Discovery.* Deste modo é possível observar que a busca nos documentos pode fornecer uma resposta ao usuário, quando não for encontrada uma resposta no *Assistant.*

Figura 24 – Resposta do *chatbot* com utilizando o *Discovery*.



Fonte: do autor.

O arquivo na Figura 25 apresenta o retorno da API do Watson *Discovery,* que foi enviado ao orquestrador quando a *query* da sentença presente na Figura 24 foi efetuada. É possível observar na linha 4 da Figura 25 que é repassado um *array* chamado de “*passages*”. Esse agrupamento de elementos retorna as 3 passagens dos documentos do *Discovery* que possuem a maior pontuação de relação com a pergunta enviada. Da linha 6 a 11 é possível observar os dados presentes em uma passagem individual retornada. Nos dados está presente o identificador do documento, uma pontuação de relação, o texto da passagem, a posição inicial e final da passagem em relação ao documento geral e o formato do texto dentro do documento.

Figura 25 – Retorno da API do *Discovery*.



Fonte: do autor.

Utilizando essa resposta da API do *Discovery* foi possível obter trechos dentro dos documentos de suporte que podem responder ao usuário quando a pergunta não estiver mapeada no *Assistant.* A passagem de texto dentro de um documento do *Discovery* pode não ser uma resposta formatada corretamente como uma resposta do chatbot a uma pergunta, mas o objetivo é da informação presente nesse pedaço do documento auxiliar de alguma maneira o usuário que estiver com uma dúvida ou problema. Portanto, o *Discovery* serve como uma ferramenta de apoio ao *chatbot,* sendo utilizado como uma base de conhecimento para ser possível ajudar os usuários em questões que são mais complexas de mapear em uma única intenção e consequentemente em uma resposta definida.

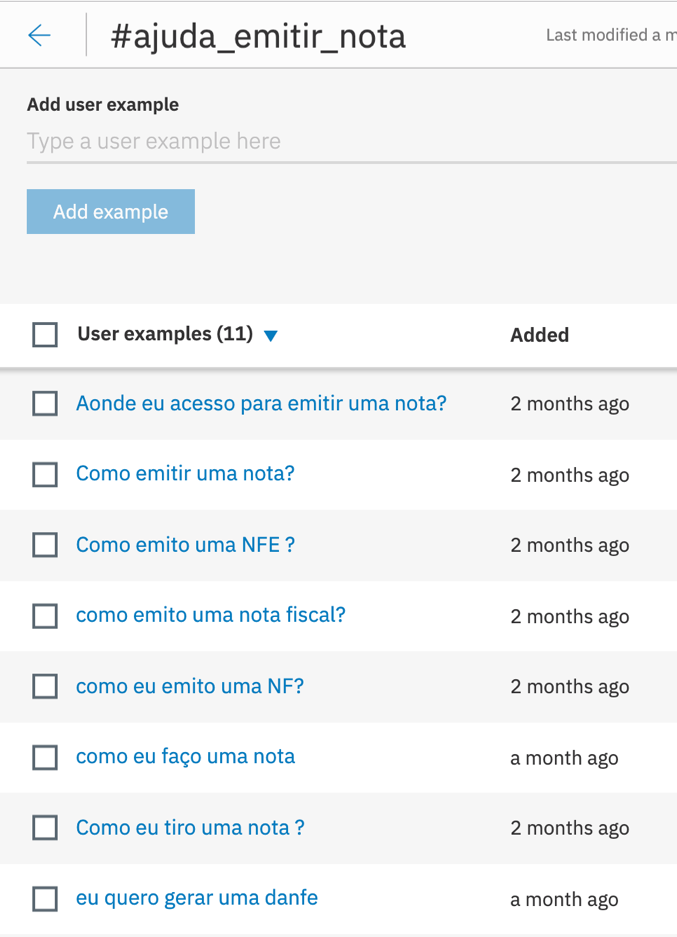
## Treinamento do Chatbot

Um *chatbot* criado com o *framework* do Watson *Assistant,* tem como objetivo identificar a intenção do usuário com a sentença enviada, fornecendo a resposta que estiver configurada nos nodos de diálogo. Para isso foram mapeadas todas as intenções principais dos usuários com exemplos de frases semelhantes, mas que possuem o mesmo objetivo final. Essas frases fornecidas inicialmente podem ser consideradas como um treinamento do assistente na criação da intenção. Porém as frases fornecidas são somente algumas das variações possíveis que podem ocorrer em um diálogo, com isso existe a chance do serviço não ser capaz de mapear uma intenção quando o usuário enviar uma pergunta ao *chatbot*.

Para aprimorar um assistente criado com esse *framework* da IBM, existem então algumas maneiras de treinamento. Essas diferentes formas de treinamento tem o objetivo de aprimorar a capacidade do *Assistant* em identificar a intenção dos usuários com uma sentença. Dentre as maneiras existentes que foram utilizadas neste trabalho estão: (i) cadastro direto das intenções na ferramenta web com o fornecimento manual dos exemplos, (ii) *upload* de intenções na ferramenta utilizando um arquivo no formato “csv”, (iii) a utilização da ferramenta de conversa teste no ambiente de desenvolvimento e (iv) a possibilidade de utilizar o histórico de conversas dos usuários para ensinar o *chatbot.*

Para a forma de treinamento manual, que equivale ao cadastro inicial da intenção e dos seus exemplos é utilizada a ferramenta web do Watson *Assistant* da IBM. A interface disponível para criação das intenções permite que o desenvolvedor informe o nome da intenção e forneça os exemplos de sentenças que devem mapear para aquela intenção. Esta interface pode ser observada na Figura 26, com o cadastro da intenção “#ajuda\_emitir\_nota” do *chatbot* deste trabalho. Neste cadastro é possível definir o nome da intenção dentro do *Assistant* e os exemplos que o serviço utilizará para realizar o treinamento de linguagem natural. O algoritmo utilizado para processar os exemplos não é especificado pelo serviço.

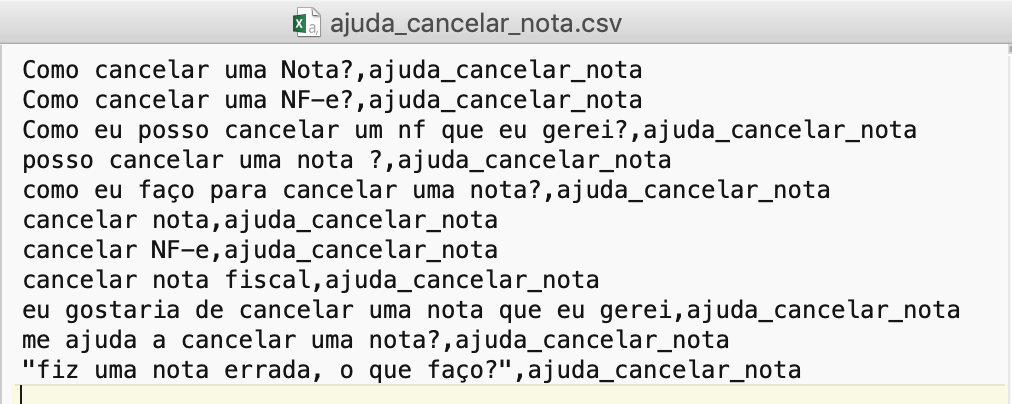
Figura 26 – Tela de criação da intenção no Watson *Assistant*.



Fonte: (IBM Watson Assistant, 2019).

Na criação do *chatbot* deste trabalho o cadastro das intenções foi realizado integralmente por meio da ferramenta apresentada na Figura 26. O cadastro através desta interface *web* permite que os exemplos sejam facilmente fornecidos, treinando o *Assistant* diretamente para a intenção informada. Outra maneira de criação das intenções e fornecimento das frases de treinamento é a importação de arquivos de texto no formato “csv”, que contenham a frase de exemplo e o nome da intenção, separados por uma virgula, conforme a Figura 27. Todas as intenções existentes no Watson *Assistant* também podem ser exportadas facilmente para o formato “csv”.

Figura 27 – Arquivo em CSV com a intenção e os exemplos



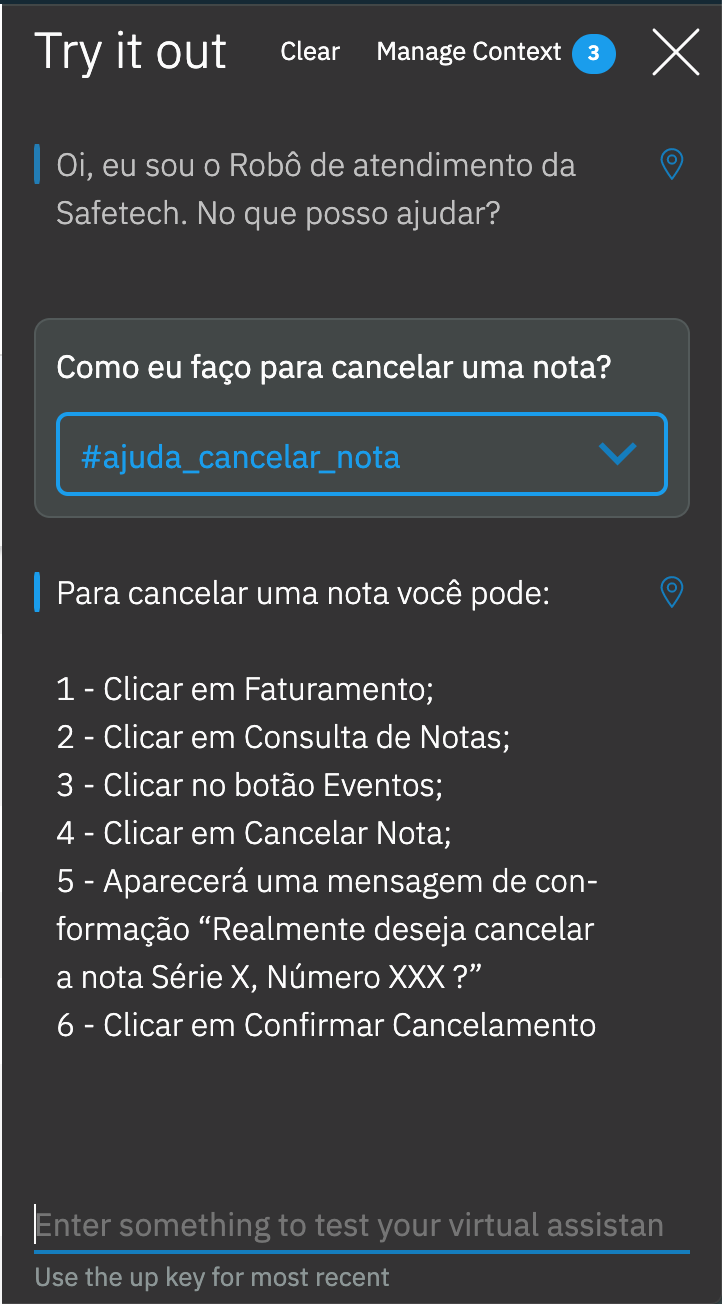
Fonte: do autor.

Essa importação e exportação de arquivos com as intenções e exemplos de sentenças que tem correlação com a intenção, possibilitam a criação de outros recursos no *chatbot*. Um dos que foi utilizado neste trabalho é a opção de exportar todas as intenções e as frases de exemplo para um arquivo, que serve como um *backup* da base de dados do Watson *Assistant*. Outra possibilidade que pode ser utilizada é a de possuir uma fonte diferente de geração de intenções ou de sentenças para as intenções e importá-las para dentro do serviço quando criadas. Dessa maneira, seria possível criar outra forma de coletar as informações das intenções para melhorar o treinamento do *framework*.

Dentro do ambiente *web* do Watson *Assistant* existe outra forma de treinamento do assistente durante a criação do mesmo. Essa forma de treinamento é uma ferramenta que cria um diálogo com o *chatbot* para ser possível os desenvolvedores testarem e aprimorarem a ferramenta. No *framework* essa ferramenta é chamada de “*try it out*” (experimente) e pode ser acessada clicando em um botão que fica disponível na parte superior da ferramenta *web*. Assim, é possível testar o atendente que está sendo criado durante qualquer etapa do desenvolvimento, seja na criação das intenções, das entidades ou durante a criação da árvore do diálogo.

Este recurso permite que uma *skill* que está sendo desenvolvida para um *chatbot* seja testada e validada antes mesmo de ser integrada a sua interface final. Pois é possível verificar se as intenções estão sendo acessadas corretamente, com interações que podem ser realizadas diretamente pela página *web* do serviço. A tela que este ambiente apresenta para homologação do *chatbot* pode ser visualizada na Figura 28. Esse recurso foi amplamente utilizado no *chatbot* deste trabalho durante o desenvolvimento das intenções, entidades e criação da árvore do diálogo. A utilização desta ferramenta permitiu com que fossem identificadas diversas intenções que careciam de exemplos sem a necessidade de executar a aplicação do *chatbot.*

Figura 28 – Ferramenta de *Try it out* (Experimente) do *Assistant*.

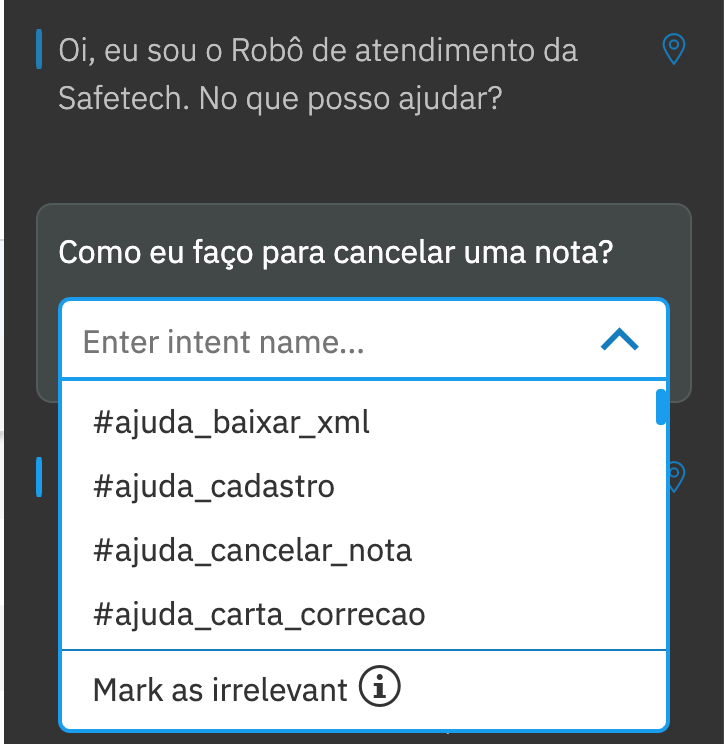


Fonte: (IBM Watson Assistant, 2019).

Na Figura 28 é possível observar que as mensagens retornadas pelo serviço ficam posicionadas a esquerda na tela do *chat* e as mensagens do usuário a direita. Neste exemplo foi fornecida a frase “Como eu faço para cancelar uma nota?” logo após a saudação do *chatbot*. No retângulo que fica logo abaixo da pergunta enviada é possível observar que o serviço indica que a intenção que ele mapeou para a frase foi a “#ajuda\_cancelar\_nota” que neste caso está correta. Como essa ferramenta simula um diálogo com o usuário final, ela já fornece também as respostas configuradas para os nodos de diálogo da árvore. Na parte superior da imagem a ferramenta disponibiliza a opção de “*clear”* (limpar) para iniciar o diálogo novamente e a opção de visualizar as variáveis existentes no contexto através do botão “*Manage Context*”.

Para os casos em que o *framework* não mapeia corretamente a intenção do usuário dada uma frase, é possível sinalizar a intenção correta. Esse comportamento pode ser visualizado na Figura 29, em que a lista de opções abaixo da sentença do usuário foi expandida. Neste exemplo caso a frase da Figura 28 não tivesse mapeada para a intenção correta, poderia ter sido selecionada a intenção correta da lista de opções. Existe também a possibilidade de indicar que a frase não deve ser mapeada para nenhuma das intenções, através da opção “*mark as irrelevant*”. Desta maneira é possível treinar o atendente informando variações de frases com o mesmo objetivo, para identificar se os exemplos de uma intenção estão corretos.

Figura 29 – Ensinando intenção com a ferramenta de *Try it out* (Experimente).

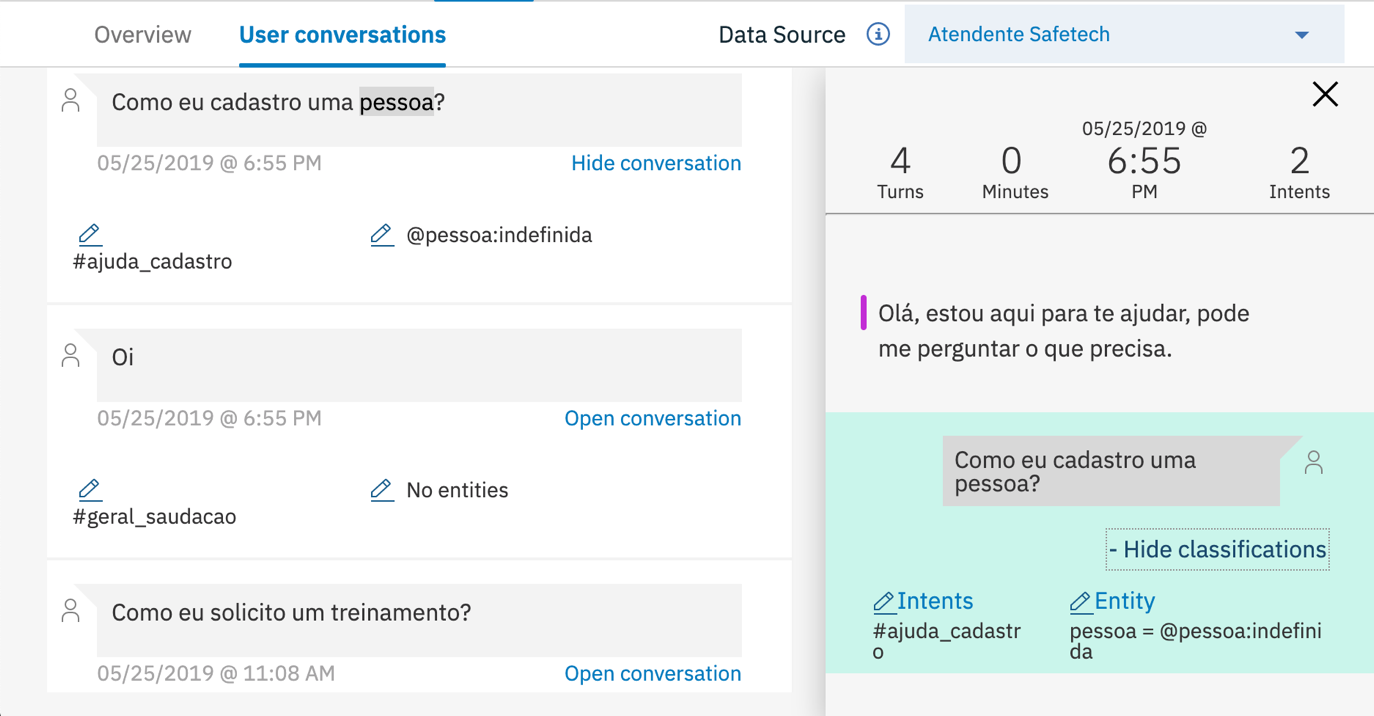


Fonte: (IBM Watson Assistant, 2019).

Outro recurso disponível dentro do Watson *Assistant* são os registros das conversas que o serviço armazena e que já foram citados anteriormente. A parte importante deste mecanismo que cabe ao treinamento do atendente é a possibilidade de aprimorar o *chatbot* nesta tela, mediante indicação e alteração de intenções identificadas. Esse recurso torna possível melhorar o atendente através da análise das conversas que já foram trocadas com os usuários, não sendo necessário armazenar nenhum log dessas conversas para realizar o trabalho manualmente depois. Esse método também foi utilizado no desenvolvimento do *chatbot* deste trabalho após os testes com a versão inicial da ferramenta.

A parte negativa deste recurso na versão *free* é que os dados com os registros das conversas duram somente 7 dias. Devido a essa limitação na versão *free,* utilizada neste trabalho, as análises e treinamentos sempre foram feitos logo após a rodada de testes com os usuários. Outra característica é que somente as conversas efetuadas através das chamadas de API são registradas, ou seja, não registra os logs dos testes. Na Figura 30 é possível visualizar a interface em que ficam esses registros dentro do *Assistant,* chamada de *Analytics.*

Figura 30 – Ferramenta de *Analytics* do Watson *Assistant*.



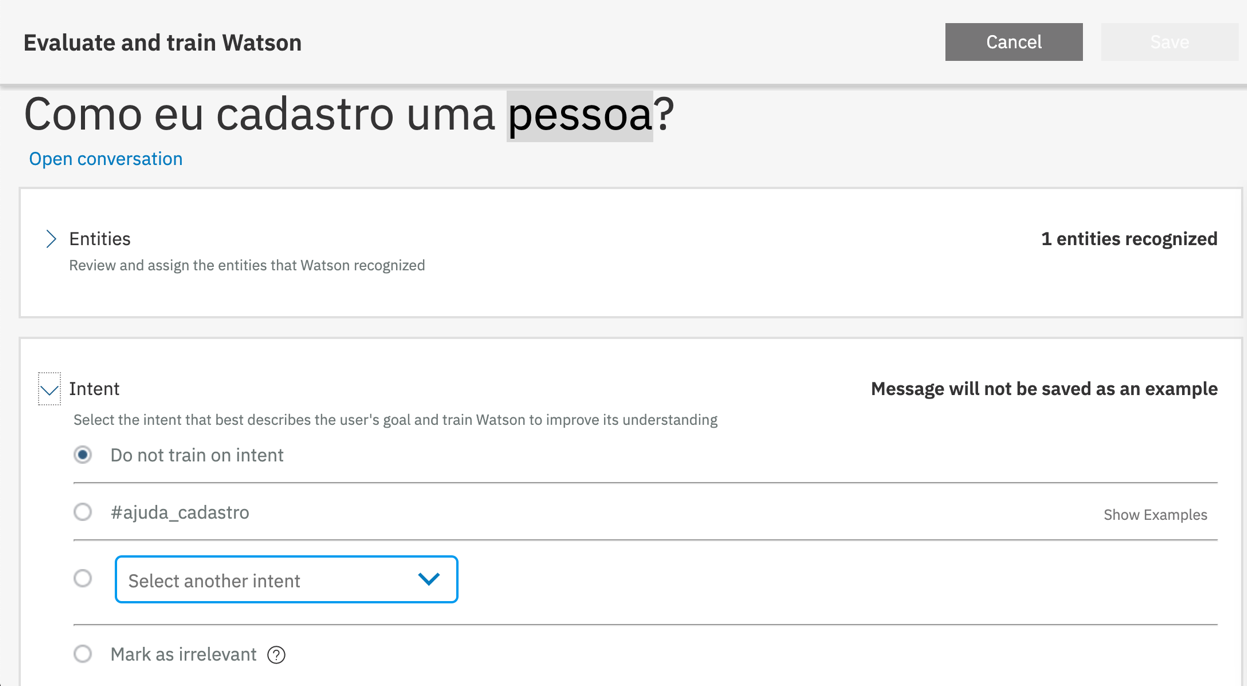
Fonte: (IBM Watson Assistant, 2019).

Na imagem é possível observar que existem dois painéis, um na esquerda com as sentenças dos usuários que foram registradas e um na direita com os detalhes da conversa. As mensagens na esquerda aparecem ordenadas de acordo como elas ocorreram, e é possível selecionar a conversa desejada para visualizá-la no painel da direita, dentro do contexto do diálogo em que ela ocorreu. Nessa ferramenta é possível verificar todo o diálogo que o usuário levou com o *chatbot,* as intenções e entidades que o assistente identificou e verificar também a resposta que o atendente forneceu para o cliente em cada caso. Além dessas informações que dizem respeito ao diálogo e conteúdo também existem informações mais analíticas sobre a conversa, como número de turnos, tempo de duração e data que ocorreu.

No decorrer do desenvolvimento do *chatbot* deste trabalho foi necessário analisar as conversas que o atendente teve com os usuários e foi utilizado esse recurso. Em alguns casos, ao verificar nos registros de log, é possível verificar que o *chatbot* identificou de maneira imprecisa algumas intenções dos usuários. Para estes casos a ferramenta disponibiliza a outra forma de treinamento citada anteriormente, que é a partir do histórico das conversas. Todas as sentenças que estão listadas na Figura 30 e tiveram uma intenção atribuída podem ser editadas para corrigir alguma relação errada que o *Assistant* possa ter efetuado, e consequentemente treinar o atendente para relacionar corretamente na próxima interação que tiver com um usuário sobre o mesmo assunto.

Na Figura 31 é possível visualizar a interface de detalhes de uma sentença, acessada através do histórico das conversas. Nesta página é possível treinar o assistente corrigindo alguma intenção ou entidade que o mesmo tenha mapeado incorretamente. No exemplo da Figura 31, o assistente mapeou a frase “Como eu cadastro uma pessoa?” para a intenção de “#ajuda\_cadastro”, caso essa intenção estivesse incorreta para a sentença, nesta página de detalhes seria possível atribuir outra intenção. Assim como na tela do *try it out* também é exibida uma lista com as intenções para o usuário selecionar a correta e existe a opção de marcar a frase como irrelevante, não mapeando para nenhuma intenção.

Figura 31 – Treinamento através do *Analytics*.



Fonte: (IBM Watson Assistant, 2019).

Portanto, por meio desta ferramenta de *analytics* do Watson *Assistant* foi possível treinar e aprimorar o *chatbot* deste trabalho. O registro das conversas em um histórico que pode ser visualizado e utilizado para treinar o *chatbot* é uma ferramenta muito poderosa e ajudou em diversas correções que foram necessárias. Para os casos em que o serviço vier a ser utilizado com um alto fluxo de conversas, é altamente recomendado que seja utilizada essa ferramenta, porém sem a limitação dos 7 dias em que são apagados os registros do históricos. Essa limitação da versão *free* pode ser superada com a contratação do serviço em um dos planos pagos, ou com a implementação de um mecanismo de histórico de conversas desenvolvido separadamente.

# Avaliação e resultados

O conteúdo utilizado como base de conhecimento para o *chatbot* deste trabalho, foi criado pela equipe responsável pelo atendimento dos clientes deste sistema, ou foi extraído de material previamente criado por esta equipe. Com isso, para efetuar os testes com o *chatbot* foi definido que a equipe de suporte faria três rodadas de conversa com o atendente a fim de mapear possíveis problemas ou pontos de melhoria. Essa escolha foi definida devido a esta equipe ter um alto nível de conhecimento sobre o sistema que o *chatbot* se propõe a atuar.

A equipe tem um grande conhecimento sobre o sistema de emissão de notas da empresa, porém não tem ciência sobre como o *chatbot* foi estruturado para este trabalho. Todo o trabalho que é feito na estrutura do Watson e no desenvolvimento do orquestrador é restrito a quem desenvolveu o trabalho. Com isso, não existe a possibilidade deste profissional conversar diretamente os tópicos que foram mapeados, o que tornaria a conversa mais genuína e sem possíveis vícios. Somado ao fato de que por esse usuário ter amplo domínio sobre o sistema, existe a possibilidade de o mesmo perguntar sobre tópicos que ainda não estão mapeados, aumentando as chances de aperfeiçoar o atendente.

Para a homologação do *chatbot* foi criado então um fluxo de avaliação para ser efetuado juntamente com a equipe do suporte durante o período de testes do atendente. Este fluxo está representado na Figura 32, contendo todas as etapas que foram idealizadas para o processo de melhoria do *chatbot*. Essas etapas iniciam no desenvolvimento, que foi retratado no capítulo 4 deste trabalho. Ao término do desenvolvimento, uma versão do *chatbot* é disponibilizada em um ambiente de testes, que é acessado pela equipe do suporte para homologação. Antes de iniciar um diálogo com o *chatbot*, foi solicitado o preenchimento do formulário [APÊNDICE A – Formulário de testes do chatbot](#_APÊNDICE_A_–)*,* para obter informações sobre qual questionamento será feito ao sistema. Posteriormente ocorre a conversa entre o *chatbot* e o testador, que ao final da conversa pode preencher o formulário, descrito no [APÊNDICE B – Formulário de avaliação do chatbot](#_APÊNDICE_B_–), indicando qual foi a sua experiência e satisfação com a conversa.

Após o preenchimento deste segundo formulário a interação da equipe do suporte com o *chatbot* acaba. A próxima etapa, que precede a do desenvolvimento, consiste na análise dos dados gerados com essas interações dos usuários com o *chatbot,* buscando identificar problemas em algum dos componentes utilizados no *chatbot.* Ao final do fluxo se inicia novamente a etapa de desenvolvimento, com os registros de todas as conversas efetuadas, e com o objetivo então de corrigir as falhas encontradas e melhorar os pontos que forem indicados como negativos.Deste modo é possível desenvolver o atendente e submetê-lo a testes para buscar possíveis falhas, com profissionais que possuem domínio sobre o assunto para indicar respostas erradas.

Figura 32 – Diagrama do fluxo de melhoria do *chatbot*.



Fonte: do autor.

Para ser possível o registro e avaliação das conversas, foram criados dois formulários para serem preenchidos pelos usuários que interagissem com o *chatbot.* Os formulários foram chamados de “Formulário de testes do *chatbot*” e “Formulário de avaliação do *chatbot*”. Os dois formulários foram criados utilizando a ferramenta gratuita Formulários Google, que permite com que sejam criadas e analisadas pesquisas gratuitamente (FORMULARIOS GOOGLE, 2019). O formulário chamado de “Formulário de testes do *chatbot*” foi criado com o objetivo de coletar as informações do testador, antes do mesmo interagir com o *chatbot,* para ser possível entender qual era o seu objetivo com a conversa. Pois o testador estaria simulando um cliente com dúvidas e a informação de qual problema ou assunto é abordado se faz muito importante para correções no atendente.

O segundo formulário, nomeado de “Formulário de avaliação do *chatbot*”, foi criado para ser utilizado após a interação do usuário com o *chatbot*. Esse formulário tem como objetivo coletar as informações de satisfação dos usuários com a conversa, como por exemplo se o *chatbot* foi capaz de solucionar o problema através da conversa e se conversaria novamente com o *chatbot*. As respostas deste formulário utilizam a escala de Likert (LIKERT, 1932), sendo atribuído um nível de gradação de cinco pontos sobre o grau de concordância com as perguntas. Com isso, foram utilizados os valores de 1 a 5 para cada resposta, sendo: (i) Discordo Totalmente, (ii) Discordo, (iii) Não Tenho Certeza, (iv) Concordo e (v) Concordo Totalmente.

O formulário de avaliação tem o preenchimento opcional e foi criado para todos os usuários que interagissem com o atendente, não somente para os testadores, diferente do formulário de testes do *chatbot,* que é apresentado somente aos testadores.

## Problemas encontrados

As primeiras rodadas de testes foram realizadas ainda durante o desenvolvimento da ferramenta. Neste período todas as intenções criadas e os recursos do Watson que foram disponibilizados passaram por testes para avaliar o seu correto funcionamento. Os testes foram realizados em sua grande maioria por meio da ferramenta de *try it out* que fica disponível dentro do ambiente do Watson *Assistant,* conforme explicado na Seção 4.5. Para os recursos que necessitavam do orquestrador, como é o caso do Watson *Discovery,* foi utilizada aplicação criada neste trabalho para os testes. Assim, para os erros encontrados dentro da ferramenta do *Assistant* o treinamento correto já foi efetuado dentro daquela etapa e para os erros mapeados via testes utilizando a aplicação, foi utilizada a ferramenta *Analytics.*

Durante estes testes o primeiro ponto observado foi que a resposta para a intenção de como emitir uma nota, possuía uma resposta muito grande. As respostas grandes não possuem uma exibição adequada dentro do *chat* e reduzem o nível de experiência do usuário. Com isso, antes da primeira versão ser disponibilizada para testes, o nodo que fornecia a resposta direta para como emitir uma nota foi dividido em mais quatro nodos. O resultado desta divisão foi apresentado na Figura 18, aonde o nodo recebeu nodos filhos para melhorar a interação do *chatbot* com os usuários e reduzir a resposta quando essa intenção fosse mapeada. Com este tratamento foi possível obter uma experiência de usuário otimizada quando essa intenção fosse identificada.

A primeira rodada de homologação do *chatbot* ocorreu dentro da empresa, com os usuários do suporte, conforme foi explicado anteriormente, e um usuário de outro setor da empresa, que possui menos conhecimento sobre o sistema. Nesta primeira rodada foram identificados alguns problemas na construção do diálogo dentro do Watson *Assistant,* alguns problemas relacionados com a busca nos documentos, a falta de exemplos em algumas das intenções e a ausência no *chatbot* de diversos assuntos relacionados ao sistema. Estes problemas, assim como as correções aplicadas serão detalhados a seguir.

Um dos problemas relacionados com a construção do diálogo, foi que inicialmente os nodos de emissão de nota fiscal e de cadastro (pessoas e produtos) utilizavam o recurso de *multiple responses,* disponível dentro do Watson *Assistant,* diretamente no nodo principal (pai)*.* Esse recurso permite que a resposta fornecida pelo nodo seja alterada de acordo com a condição definida. O problema que ocorreu foi que a última condição do nodo, que possuía a condição especial *true* e sempre era acionada quando nenhuma das outras condições fosse atendida. Essa lógica está correta, porém os nodos filhos validavam novamente a resposta que deveria ser fornecida para os casos em que a condição não fosse atendida no nodo pai e isso causou um loop infinito dentro dos fluxos em questão quando a sentença do usuário mapeava diretamente para a resposta final, sem a necessidade de validar os nodos filhos.

Um cenário de conversa em que este problema ocorreu foi quando o usuário informou a frase: “Como eu faço para cadastrar uma empresa?”. Essa sentença mapeou diretamente para o nodo com a intenção de “#ajuda\_cadastro” e a condição da entidade ser uma pessoa jurídica também foi mapeada diretamente e com isso a resposta fornecida para o usuário foi a resposta correta de como realizar o cadastro. Porém, após a resposta condicional do nodo pai ser fornecida, o diálogo foi posicionado no primeiro nodo filho ligado a ele, o que fez com que o usuário ficasse preso no fluxo de cadastro, porque os nodos filhos ficavam tentando encontrar a informação sobre qual cadastro o usuário desejava realizar, e a resposta já havia sido fornecida.

Para resolver este problema, a opção de *multiple responses* foi removida do nodo pai e o nodo foi programado para quando identificar a intenção de cadastro, pular diretamente para a validação dos nodos filhos. No último nodo filho foi adicionada a condição especial de *true*, que joga a validação para o primeiro nodo, para realizar a pergunta novamente quando não for possível mapear a entidade correta. Com isso, para a sentença de exemplo acima, o nodo pai identifica a intenção de cadastro e o nodo filho de pessoa jurídica será acionado para fornecer a resposta ao usuário, finalizando o fluxo do diálogo de cadastro. Este comportamento também foi implementado no fluxo de emissão de nota, em que ocorreu o mesmo problema.

Em suma, essa opção de condição dentro das respostas do nodo pai foi removida de todos os nodos existentes neste trabalho. Esse recurso presente no Watson *Assistant* é muito útil para fornecer respostas dinâmicas com base em condições, tornando o diálogo mais realista. Porém, não se adéqua bem nos casos em que deve ser criado o mecanismo de *loop* dentro dos nodos. Essa abordagem da utilização do *loop* se fez necessária para coletar todos os dados antes de fornecer a resposta a uma intenção, como é o caso das intenções de cadastro e forma de contato. Após a correção aplicada esses fluxos foram corrigidos, não apresentando mais problemas.

O problema relacionado a busca em documentos do *Discovery* que ocorreu, foi que o mesmo não foi invocado nenhuma vez. Durante a primeira rodada de homologações realizadas com o *chatbot,* diversos assuntos questionados não foram encontrados. Apesar disso, em nenhum destes casos o Watson *Assistant* indicou para o orquestrador que o serviço de consulta em documentos deveria ser realizado para disponibilizar uma resposta. A abordagem inicial para o *Assistant* indicar ao orquestrador que deve ser realizada a busca em documentos é a intenção “#ajuda\_geral” conforme descrito na seção 4.1. Contudo, o que se observou na primeira interação com os usuários foi de que essa abordagem não foi efetiva.

Analisando os casos foi observado que seriam necessários muitos exemplos distintos para o treinamento da intenção “#ajuda\_geral”. Essa grande quantidade de exemplos faria com o conceito de intenção do serviço fosse utilizado inadequadamente, pois essa intenção seria muito genérica. O problema que poderia ocorrer ao serem fornecidos mais exemplos para essa intenção é que ela seria acionada em mais situações, mesmo não sendo a intenção do usuário. Desse modo, foi necessário utilizar outro mecanismo para indicar ao orquestrador que o *Discovery* deveria ser chamado. O fluxo do *chatbot* continuou o mesmo, com o *Assistant* indicando ao orquestrador quando a consulta à base de documentos deve ser efetuada, porém com outro tratamento.

Sempre que o *Assistant* não identifica a intenção do usuário ele passa por todos os nodos do diálogo, fornecendo a resposta do último nodo, que possui a condição especial *anyting\_else*. Esse nodo sempre retorna a resposta que não foi possível entender o que o usuário deseja e se o mesmo pode reformular a frase. O que acontece é que o usuário tenta repetidas vezes a mesma pergunta, escrita de maneiras diferentes, mas o chatbot continua não identificando porque não tem a pergunta mapeada. Para isso, foi implementada uma solução baseada em Tammy (2018), em que é adicionada uma variável de contexto com um contador no nodo final do diálogo. No caso de Tammy (2018) o contador foi adicionado para o atendente não ficar repetindo que não entendeu, mas sim encaminhar para uma pessoa.

No caso deste trabalho, a implementação é semelhante, porém o contador foi criado para atualizar a variável de contexto com a ação. Assim, o contador é incrementado sempre que o nodo final do diálogo é acessado. Quando o contador chega ao valor dois, significa que a resposta do *chatbot* de que não entendeu já foi fornecida duas vezes, nesse caso, a variável de contexto para invocar o Discovery é gravada. Com essa modificação o atendente deste trabalho não fica respondendo repetidas vezes que não entendeu, mas sim consulta a base de documentos em busca de uma resposta. O nodo de *welcome* sempre inicializa essa variável com zero, como pode ser observado na Figura 21.

Na Tabela 6 foram relacionados todos os assuntos que o *chatbot* não foi capaz de responder durante essa rodada de homologação. Esses assuntos são todos relacionados ao sistema de emissão de notas da empresa, porém não haviam sido identificados como dúvida dos usuários até então. A interação do atendente com os usuários que prestam suporte do produto ocasiona dúvidas mais complexas e que muitas vezes não estão mapeadas, já a interação com usuários de outras equipes geram dúvidas mais simples, mas que também são importantes de estarem no sistema.

Tabela 6 – Conteúdos ausentes no *Chatbot* identificados na primeira homologação

|  |  |
| --- | --- |
| Prazo para cancelamento de uma nota fiscal | Como solicitar uma visita ou treinamento presencial |
| Como consultar uma nota fiscal existente | Como consultar um manual sobre o sistema |
| O que é uma nota fiscal | Como cadastrar os impostos dentro de uma nota |
| Cadastro de CFOP (Operação fiscal) | Como funciona a substituição tributária dentro do sistema |
| Como se cadastram transportadoras | Como cadastrar os vencimentos da NFE |
| Como alterar o cadastro de um cliente | Tipos de nota que o sistema emite |

Fonte: do autor.

Para adicionar estes assuntos ao *chatbot,* algumas novas intenções foram criadas dentro do Watson *Assistant* para contemplar uma parte do conteúdo*.* A outra parte do conteúdo ausente foi adicionado ao documento de dúvidas do Watson *Discovery*. Desta maneira, a base de conhecimento do *chatbot* foi enriquecida, pois esses conteúdos foram adicionados. Essa homologação, efetuada junto ao *chatbot*, foi importante para identificar essas falhas descritas e para adicionar mais conteúdo para o atendente ser capaz solucionar dúvidas de mais assuntos sobre o sistema.

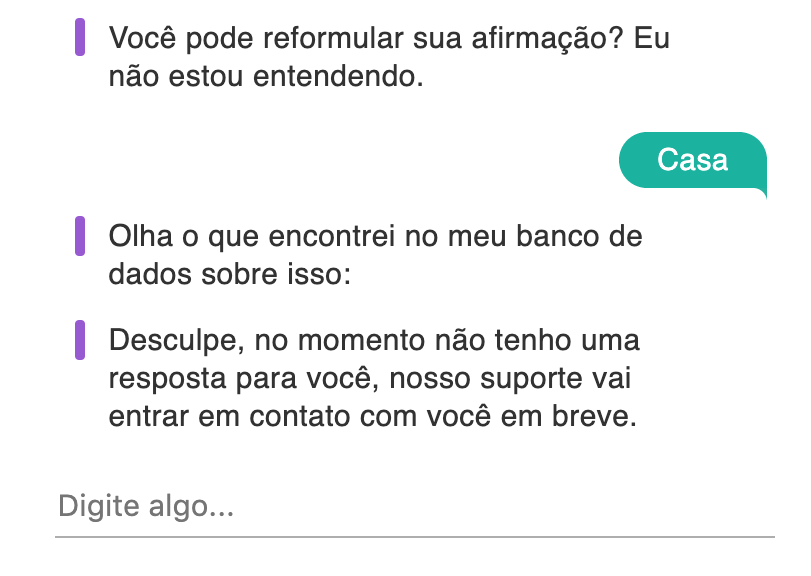
Um comportamento que foi observado durante essa fase de testes, identificado através do histórico das conversas, foi o de que algumas conversas em que o usuário enviava somente uma palavra ao *chatbot*. Por exemplo as palavras “alíquotas”, “impostos” e “vencimentos”. Nessas interações em que o usuário enviou somente uma palavra o *chatbot* não identificou a intenção do usuário e pediu para o mesmo reformular a pergunta. Para esses casos foi considerado que a ação do *chatbot* está correta, porque como o usuário não realizou uma pergunta mais completa, fica muito difícil identificar o que o mesmo deseja de maneira automática. Então nestas situações não foi realizada nenhuma interferência à lógica do *chatbot.*

Um resumo que pode ser feito da primeira rodada de interação com os usuários do *chatbot* é que grande parte dos problemas encontrados foi devido a falta de informações fornecidas ao atendente. A outra parte dos problemas foi devido à algumas implementações erradas no fluxo do diálogo, e algumas alterações que foram necessárias na lógica. Após essa primeira rodada de testes, as informações foram compiladas e o atendente foi ajustado, conforme descrito. Após os ajustes a homologação do *chatbot* seguiu para mais uma rodada de testes com os mesmos usuários da primeira rodada, com a adição de mais uma pessoa que ainda não havia participado.

Na segunda rodada de testes as consultas ao *Discovery* começaram a ocorrer com mais frequência. Devido a isso, foram verificados mais dois pontos de melhoria em relação a utilização deste serviço. Uma delas foi a necessidade de adicionar um tratamento quando a consulta às passagens de texto do *Discovery* não encontrasse nenhum resultado. Quando o *Assistant* indicava ao orquestrador que deveria ser realizada uma consulta aos documentos, era enviada uma mensagem de texto para o usuário, informando que uma consulta seria realizada. Na sequência desta mensagem o orquestrador concatena o melhor resultado encontrado. Porém, quando o orquestrador não encontrava nenhum resultado o usuário recebia a mensagem da consulta e nenhuma outra.

Para solucionar este caso, foi alterado o orquestrador para concatenar um texto de que nenhum resultado foi encontrado no banco de dados para a pergunta do usuário, como pode ser observado na Figura 33. Assim, quando alguma pergunta do usuário não for encontrada no *Assistant* ou no *Discovery,* a mensagem acima será retornada. O outro caso identificado envolvendo o *Discovery* foi o da formatação das mensagens. O retorno do serviço do Discovery é processado pelo orquestrador e a melhor resposta retornada é enviada. Porém, esta mensagem é um trecho do arquivo em que foi encontrada e não está formatada para um *chat*.

Figura 33 – Pergunta não encontrada no *Discovery*.

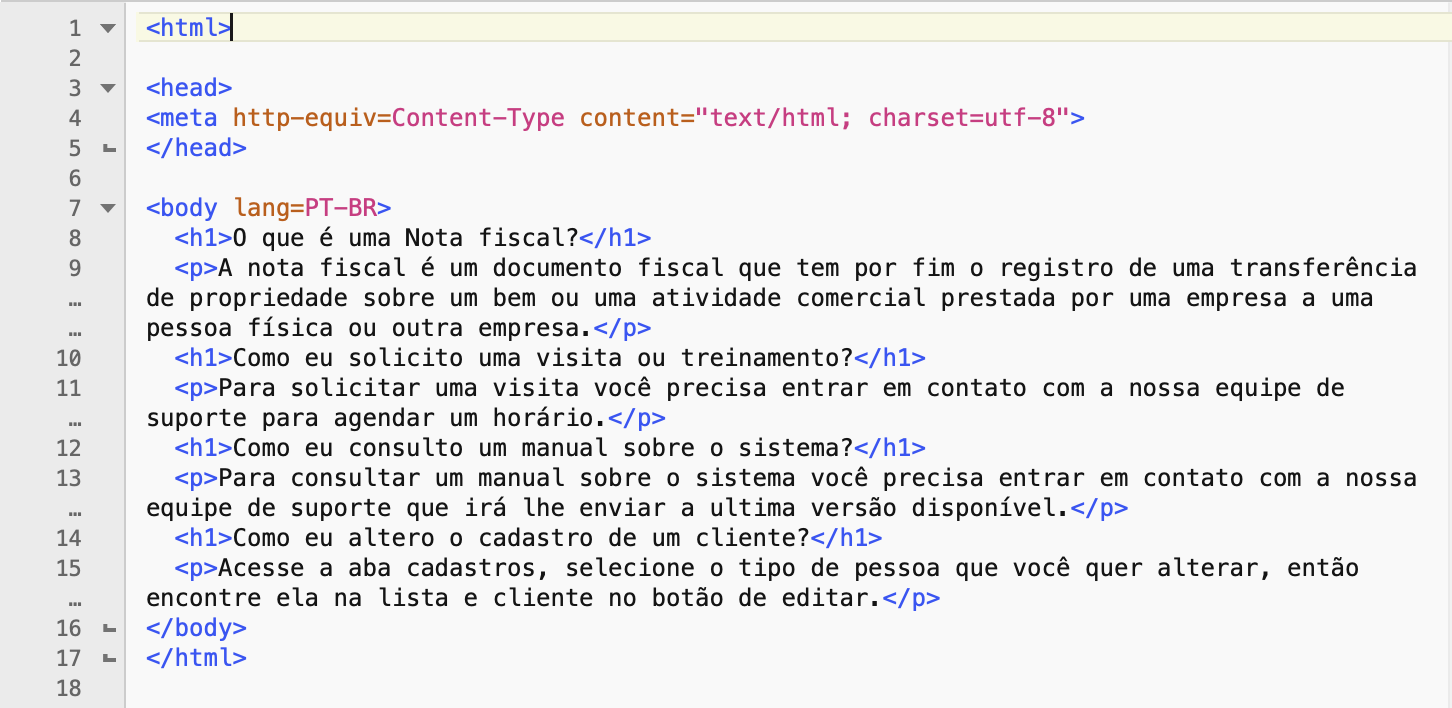


Fonte: do autor.

Para os arquivos no formato *pdf* ou *docx* que foram carregados no framework não existe nenhuma opção de formatação. Contudo, foi criado um novo arquivo no formato *html*, com as perguntas organizadas em forma de FAQ, com perguntas e respostas bem definidas. Esse arquivo foi criado para adicionar os conteúdos que foram surgindo durante a homologação do *chatbot,* e o formato *html* foi escolhido por ser possível editar o retorno antes de enviar para o usuário. Quando o Discovery envia um retorno ele indica em que formato o texto da passagem que ele enviou. Com isso, foi possível adicionar um tratamento no orquestrador para o retorno em *html*. A Figura 34 apresenta uma versão deste arquivo.

No arquivo é possível observar que todas as perguntas foram formatadas com uma *tag* “<h1>” e todas as respostas foram incluídas com a *tag* “<p>”. Dessa forma, o orquestrador processa a passagem de retorno quando a mesma estiver em *html*, retirando fora a linha que possui a pergunta, deixando somente o texto da resposta. A utilização desta implementação faz com que seja retornado para o usuário somente a resposta para a pergunta, sem a pergunta junto. A Figura 24 apresentou uma resposta para o usuário que foi retornada do *Discovery*, que está presente no arquivo da Figura 34. Como pode ser observado, a resposta está formatada corretamente.

Figura 34 – Arquivo em *html* de perguntas e respostas.



Fonte: do autor.

A criação deste tratamento dentro do orquestrador juntamente com a definição de um padrão de arquivo para importação no *Discovery,* criam um modelo de treinamento para o serviço. Utilizando essa abordagem é possível fornecer respostas formatadas para os usuários, melhorando assim a experiência do usuário ao utilizar o serviço. Também é possível incrementar o arquivo quando forem identificadas mais perguntas sem a necessidade de realizar alterações em nenhum outro componente do *chatbot*.

Além destes tratamentos adicionados ao *Discovery*,também foram identificadas mais intenções do usuário que não estão mapeadas em nenhum local. Estas intenções estão listadas na Tabela 7. Na tabela estão concentrados os assuntos que não foram identificados pelo *chatbot* quando o mesmo foi questionado sobre eles na segunda rodada de homologação. Através da tabela é possível observar que o número de intenções não identificadas foi reduzido significativamente. Na segunda rodada de testes alguns usuários também repetiram as perguntas da rodada anterior para testar se o *chatbot* havia sido corrigido. Esses assuntos obtidos ao final da segunda rodada de testes não foram incorporados ao sistema, devido a falta de tempo deste projeto e ao conteúdo ainda não estar documentado em nenhum local dentro da empresa.

Tabela 7 – Conteúdos ausentes no *Chatbot* identificados na segunda homologação

|  |  |
| --- | --- |
| Cadastro de NCM do produto | Modalidade de frete na emissão da nota fiscal |
| Cadastro de contatos da empresa Jurídica | Como cadastrar as informações de cobrança da nota fiscal |
| Cadastro dos endereços das pessoas | Emissão do relatório gerencial de notas emitidas |

Fonte: do autor.

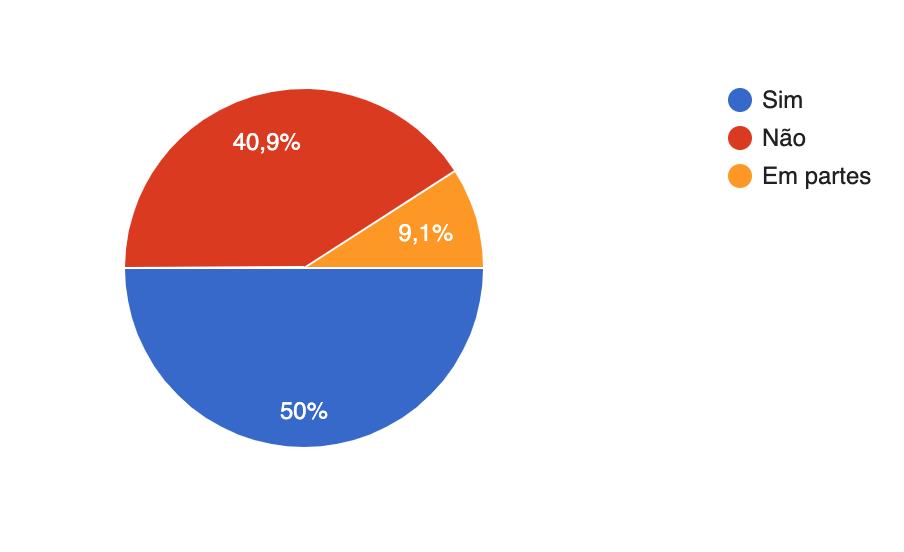
Ao final da segunda rodada de refinamento do *chatbot* é possível constatar que os erros de lógica do diálogo foram mitigados. Foi necessário somente alguns ajustes no *Discovery* para retornar com uma melhor formatação as respostas. Nenhuma intenção dentro do *Assistant* foi identificada de maneira incorreta. Assim sendo é razoável constatar que a lógica criada para o chatbot deste trabalho atua corretamente dentro da ideia em que foi concebido. O maior problema são as perguntas em que o ainda não existe material gerado sobre o conteúdo, devido a própria empresa ainda não possuir esses manuais ou alguma outra forma de documentação.

## Resultados

Um total de 22 pessoas interagiram com o *chatbot* durante o período de testes e responderam o formulário de testes. As duas primeiras perguntas eram referentes a interação do usuário com o atendente*,* e foram utilizadas para o treinamento do *chatbot* nos casos em que ocorria algum erro ou falta de informação, sendo elas: (i) “Descreva com suas palavras qual pergunta será realizada ao Chatbot? Ou o problema do usuário que será simulado.” e (ii) “Na sua análise essa pergunta é sobre qual assunto? Em qual categoria ela se enquadraria?”. Com essas duas perguntas foi possível entender qual o objetivo do testador para ser possível reproduzir em testes posteriormente.

Para a questão 3 do formulário “O chatbot respondeu corretamente a pergunta?” obteve-se um total de 9 respostas não, totalizando 40,9%, 2 respostas “Em partes” com total de 9,1% e 11 respostas sim, totalizando 50% das respostas. A representação gráfica dos resultados desta pergunta pode ser verificada na Figura 35.

Figura 35 – Gráfico da questão 3 do formulário de testes do chatbot.

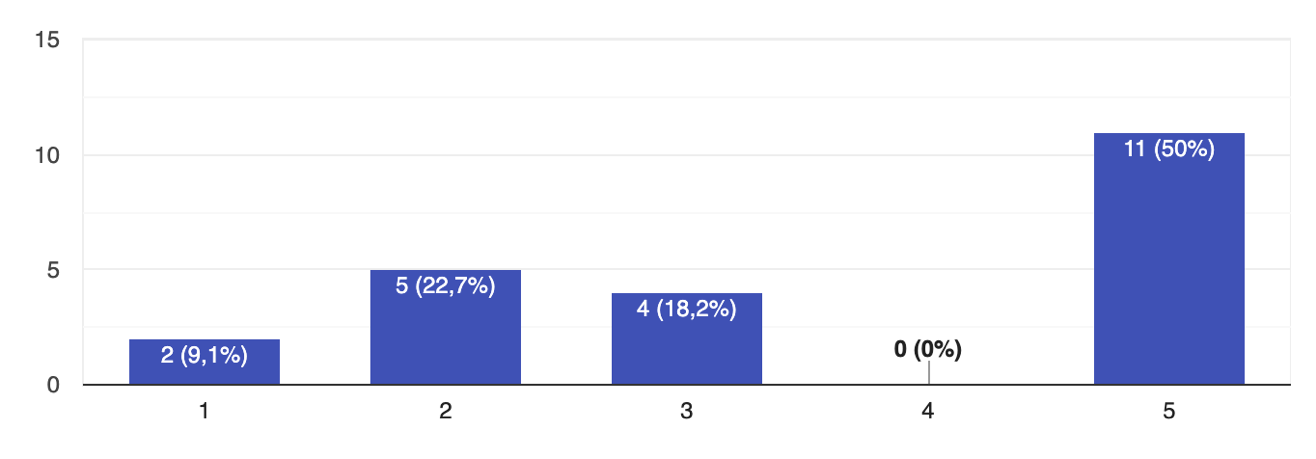


Fonte: do autor.

A questão 4 do formulário, “Se o chatbot errou, qual seria a resposta correta?”, foi utilizada para treinamento do *chatbot* quando a resposta fornecida não estivesse correta. Essa questão foi respondida somente quando o *chatbot* não fornecia uma resposta precisa.

A última questão do formulário de testes do atendente, possui o seguinte texto: “Em uma escala de 1 a 5, qual a nota que você atribui a performance do chatbot?”. Na escala, a nota 1 significa uma péssima performance, enquanto a nota 5 representa uma ótima performance. Para essa questão todos os usuários forneceram uma resposta. A Figura 36 apresenta a representação gráfica das notas fornecidas. Um percentual de 9,1% dos avaliadores atribuiu a nota 1, 22,7% atribuiu a nota 2, 18,2% atribuíram a nota 3, nenhum avaliador utilizou a nota 4 e 50% dos avaliadores forneceu a nota 5.

Figura 36 – Notas atribuídas ao chatbot no formulário de testes.



Fonte: do autor.

O formulário de avaliação do *chatbot,* por sua vez, foi completado por 23 usuários que interagiram com o atendente. Durante o período de testes dentro da empresa alguns usuários foram selecionados para interagir com o *chatbot* e responder o questionário. Na pergunta 1 os usuários tinham a opção de resposta dicotômica, e para todas as demais perguntas foi utilizada a escala de Likert para coletar as respostas.

No tocante da pergunta 1: “Você já tinha conversado com um chatbot antes?”, um total de 82,6% dos usuários responderam que já haviam conversado com um *chatbot* antes, enquanto 17,4% das pessoas responderam contrariamente. Abaixo serão apresentados os resultados para as demais perguntas.

**Pergunta 2**: “Você aprendeu alguma coisa com essa conversa?”

Um total de 60,9% dos usuários concordaram que aprenderam alguma coisa com a conversa, enquanto 4,3% concordaram totalmente com a pergunta. Um percentual de 17,4% dos usuários respondeu não ter certeza se aprendeu algo. Os usuários que discordaram totalizam 8,7% e os que discordaram totalmente 8,7%. Deste modo, concluímos que 65,2% dos usuários obtiveram algum aprendizado conversando com o *chatbot*.

**Pergunta 3**: “O Chatbot conseguiu entender as suas perguntas e respostas?”

O percentual de 43,5% dos usuários indica a concordância de que o atendente foi capaz de entender as suas perguntas e respostas, 13% concordaram totalmente com essa questão, 4,3% não tiveram certeza, 21,7% discordaram com essa pergunta e 17,4% discordaram totalmente. Portanto, é possível concluir que 56,5% dos usuários concordam que o *chatbot* foi capaz de compreender as suas perguntas e respostas, enquanto 39,1% discordam.

**Pergunta 4:** “A sua dúvida ou problema foi resolvido após a conversa?”

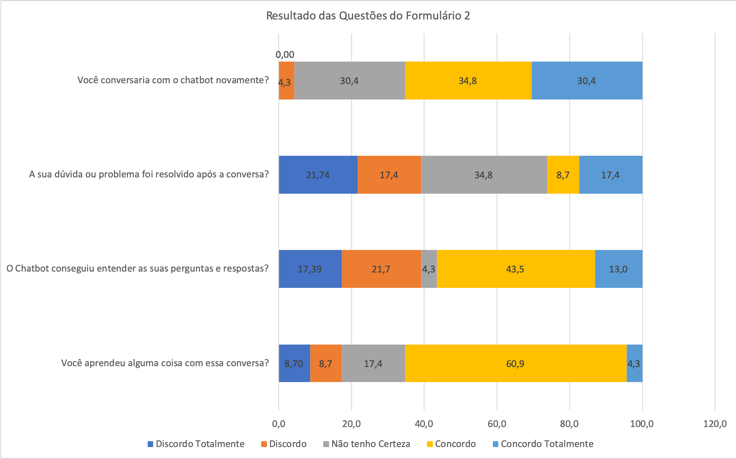
Do total dos usuários, 17,4% discordam que o atendente tenha resolvido o seu problema, 21,7% discordam totalmente que o problema foi resolvido, 34,8% não tiveram certeza se o problema foi resolvido, 8,7% concordaram que o *chatbot* resolveu o seu problema e 17,4% concordaram totalmente que seu problema foi resolvido. Assim sendo, 39,1% dos usuários discordaram que o seu problema foi resolvido, enquanto 26,1% consideram que o seu problema foi resolvido e 34,8% não tem certeza. Concluímos que o alto número de usuários que não tem certeza é devido a respostas com pouco conteúdo ou com falta de continuidade do diálogo sobre a questão.

**Pergunta 5**: “Você conversaria com o chatbot novamente?”

A parcela de 34,8% dos usuários concorda que conversaria com o atendente novamente, 30,4% concordam totalmente com essa pergunta, 30,4% não tem certeza se conversariam com o *chatbot* novamente, e apenas 4,3% discordam. Nenhum dos usuários discordou totalmente desta pergunta. Consequentemente, podemos concluir que 65,2% dos usuários concordam que conversariam com o *chatbot* em uma nova ocasião e apenas 4,3% não conversariam novamente. Consideramos que esse resultado seja favorável devido ao fato do *chatbot* ainda carecer de diversos assuntos, pois indica que os usuários estão dispostos a utilizar a ferramenta mesmo com algumas limitações.

Na Figura 37 é apresentada uma representação gráfica dos resultados do formulário de avaliação do *chatbot.* Na Figura 37, a pergunta 1 do formulário de avaliação foi omitida, devido a pergunta não utilizar a escala de Likert, que é utilizada nas demais questões.

Figura 37 – Resultados do formulário de avaliação.



Fonte: do autor.

# CONCLUSÃO

Pode-se afirmar que os chatbot estão presentes atualmente em diversas áreas e tem aplicação em diversos tipos de situação e problema. As tecnologias e as formas de implementação evoluem a cada dia, na medida em que avanços nas áreas de inteligência artificial, *machine learning* e processamento de linguagem natural evoluem. Ao mesmo tempo em que novas técnicas e algoritmos são criadas, ampliando a capacidade dos desenvolvedores de criar chatbots, os frameworks mais completos conseguem entregar soluções mais satisfatórias comercialmente para as empresas que não podem investir tempo em pesquisa e desenvolvimento de algoritmos ou em treinamento de funcionários.

O Watson da IBM e toda a tecnologia que foi necessária para a sua concepção mostram que o mesmo é uma solução completa para análise de linguagem natural e para a criação de *chatbots*, que podem ser integrados aos dados das empresas e disponibilizados mais rapidamente para os usuários. A arquitetura proposta para este trabalho cumpriu o seu propósito, através do desenvolvimento de um *chatbot* capaz de atender os usuários que possuem dúvidas sobre o sistema de emissão de notas da empresa.

O desenvolvimento deste trabalho, apresentou a maneira como o conteúdo de suporte sobre o sistema foi coletado, organizado e mapeado para dentro dos serviços da IBM. Mostrou o modo como os serviços do Watson *Assistant*, Watson *Discovery* e Watson *Natural Language Understanding* foram estruturados para serem utilizados em conjunto. Em seguida, foi detalhado o desenvolvimento da interface de conversação com o usuário e do *software* responsável pela integração e comunicação de todos os serviços. Do mesmo modo, foram apresentadas todas as formas de treinamento utilizadas para aperfeiçoar o atendente. Por fim, foram descritos os principais problemas encontrados durante a implementação, assim como as soluções aplicadas.

Para avaliar o desempenho do *chatbot* criado, foram realizados testes de conversação com a equipe responsável pelo suporte do sistema e com alguns usuários internos da empresa, com a aplicação de dois questionários. Um com objetivo de coletar dados e melhorar o atendente e outro para avaliar o desempenho. Com base nas respostas coletadas é possível concluir que o *chatbot* apresentou resultados satisfatórios. Por meio dos resultados, concluímos que 65,2% dos usuários aprenderam alguma coisa utilizando o sistema, e em relação as respostas, o *chatbot* respondeu corretamente 50% das perguntas.

Um ponto observado foi o percentual de 34,8% de usuários que não tem certeza que o seu problema foi resolvido. Possivelmente este percentual é devido as respostas não estarem corretas ou a falta de um aprofundamento dentro dos conteúdos na concepção do assistente. Os que discordam que o seu problema foi resolvido totalizaram 39,1%, enquanto 26,1% concordaram que o seu problema foi resolvido. Estes números mostram, que grande parte dos usuários consideram que as respostas não estavam incorretas, porém não estavam completas o suficiente para que as dúvidas fossem consideradas respondidas.

É importante ressaltar que a qualidade e a quantidade das respostas do *chatbot* está diretamente relacionada ao conteúdo disponível. Grande parte das perguntas que o *chatbot* não soube responder surgiram durante a etapa de testes, não estando documentado em nenhum local da empresa estes conteúdos. Neste caso, seria necessário um trabalho no sentido de gerar documentação sobre o sistema para ser possível utilizar para treinamento do *chatbot.* Atualmente, a empresa possui pouca documentação sobre o sistema foco de suporte deste trabalho, com algumas iniciativas neste sentido iniciando, motivadas pela criação do *chatbot*. Uma sugestão de trabalhos futuros é ampliar a base de conhecimento do atendente, pois o volume de informações utilizadas no *chatbot* esta diretamente relacionado com o resultado final.

Referências Bibliográficas

ALMEIDA JR, Oberdan Alves de. Beck: Um Chatbot Baseado na Terapia Cognitivo-Comportamental para Apoiar Adolescentes com Depressão. p. 166, 2017.

ARAUJO, J. P. d. Agentes Conversacionais no Ensino-Aprendizagem de Línguas. 2010.

AZRAQ, Ahmed; AZIZ, Hala; NAPPE, Nicolas; BRAVO, Cesar Rodriguez; SRI, Lak. Building Cognitive Applications with IBM Watson Services: Volume 2 Conversation. p. 248, 2017.

BADA, E. M.; MENEZES, C. S. d. Uma proposta para extração de perguntas e respostas de textos. 2012.

BERNARDINI, Andréia Ana; SÔNEGO, Arildo Antônio; POZZEBON, Eliane. CHATBOTS: UMA ANÁLISE BIBLIOMÉTRICA DO ESTADO DA ARTE DA LITERATURA. ARTEFACTUM, p. 14, 2018.

BLIP. Disponível em: < <https://blip.ai/construcao/>>. Acesso em 06 setembro de 2018.

CUMMINS, Holly. Chatbot Best Practices. Disponível em: <https://www.ibm.com/cloud/blog/chatbot-best-practices>. Acesso em 21 maio de 2019.

DALE, Robert. The return of the chatbots. Natural Language Engineering, v. 22, n. 05, p. 811–817, 2016. (Cambridge University Press).

DAS, Biplab Ch. Question Answering. p. 62, 2014.

FERRUCCI, David; BROWN, Eric; CHU-CARROLL, Jennifer; FAN, James; GONDEK, David; KALYANPUR, Aditya A.; LALLY, Adam; MURDOCK, J. William; NYBERG, Eric; PRAGER, John; SCHLAEFER, Nico; WELTY, Chris. Building Watson: An Overview of the DeepQA Project. AI Magazine, v. 31, n. 3, p. 59, 2010.

FERRUCCI, D. A. Introduction to “This is Watson”. IBM Journal of Research and Development, v. 56, n. 3.4, p. 1:1-1:15, 2012.

FERRUCCI, David; LALLY, Adam. UIMA: an architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering, v. 10, n. 3–4, p. 327–348, 2004.

FERRUCCI, David; LEVAS, Anthony; BAGCHI, Sugato; GONDEK, David; MUELLER, Erric T. Watson: Beyond Jeopardy! Artificial Intelligence, v. 199–200, p. 93–105, 2013.

FORMULÁRIOS GOOGLE. Disponível em: < https://www.google.com/intl/pt-BR/forms/about/>. Acesso em 26 maio de 2019.

GLIOZZO, Dr Alfio; ACKERSON, Chris; BHATTACHARYA, Rajib; et al. Building Cognitive Applications with IBM Watson Services: Volume 1 Getting Started. p. 130, 2017.

GLIOZZO, Alfio; BIRAN, Or; PATWARDHAN, Siddharth; et al. Semantic Technologies in IBM Watson. p. 8, 2013.

HIGH, Rob. The Era of Cognitive Systems: An Inside Look at IBM Watson and How it Works. p. 16, 2012.

HUANG, Jizhou; ZHOU, Ming; YANG, Dan. Extracting Chatbot Knowledge from Online Discussion Forums. IJCAI, p. 423–428, 2007.

IBM. Disponível em: < https://www.ibm.com/us-en/?lnk=m>. Acesso em 08 setembro de 2018.

IBM Watson Assistant. Disponível em: < https://assistant-us-south.watsonplatform.net/ >. Acesso em 07 maio de 2019.

JSON. Introdução ao JSON. Disponível em: < https://www.json.org/json-pt.html>. Acesso em 22 maio de 2019.

JEOPARDY!. In: Wikipédia: a enciclopédia livre. Disponível em: < https://en.wikipedia.org/w/index.php?title=Jeopardy!&oldid=869129683> Acesso em: 16 nov 2018.

LIKERT, R. A technique for the measurement of attitudes. Archives of psychology, 1932.

MAEDA, A. C.; MORAES, S. M. W. Chatbot baseado em Deep Learning: um Estudo para Língua Portuguesa. 2017.

MARIETTO, M. et al. Artificial Intelligence Markup Language: A brief tutorial. CoRR,2013.

MATHUR, A. Program your chatbot to handle long-tail questions with Watson Conversation and Watson Discovery. developer.ibm.com, 2017. Disponível em: < https://developer.ibm.com/dwblog/2017/chatbot-long-tail-questions-watson-conversation-discovery/>. Acesso em: 17 abr. 2017.

NODE.JS. Disponível em: < https://nodejs.org/pt-br/about/>. Acesso em 08 abril de 2019.

NODE.JS NPM. Disponível em: < https://www.w3schools.com/nodejs/nodejs\_npm.asp>. Acesso em 22 maio de 2019.

OLIVEIRA, Tiago Luiz Martiniano de Oliveira; MARQUES, Daniela. Desenvolvimento de Chatbot para auxílio ao ensino à distância usando metodologia Lean. 2018.

PRIMO, Alex; COELHO, Luciano Roth. Comunicação e inteligência artificial: interagindo com a robô de conversação Cybelle. In: MOTTA, L. G. M. et al. (Eds.). Estratégias e culturas da comunicação ed.Brasília. Brasília: Editora Universidade de Brasília, 2002. p. 83-106.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de.​ Metodologia do trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico​. 2. ed. Novo Hamburgo: Universidade Feevale, 2013. 276 p.

SANTARÉM SEGUNDO, José Eduardo. Web Semântica: introdução a recuperação de dados usando SPARQL. In: ENCONTRO NACIONAL DE PESQUISA EM CIÊNCIA DA INFORMAÇÃO, 15., 2014, Belo Horizonte. Anais… Belo Horizonte: UFMG, 2014.

SOUZA, LS d; MORAES, Silvia Maria Wanderley. Construção automática de uma base AIML para chatbot: um estudo baseado na extração de informações a partir de FAQs. Anais do XII ENIAC, p. 137-141, 2015.

SOUZA, Roger Florzino de. DESENVOLVIMENTO DE CHATBOT APLICADO AO ATENDIMENTO DE CLIENTES EM E-BUSINESS. p. 78, 2018.

TAMMY. Using Advanced Dialog Features in Watson Conversation, 2018. Disponível em: <https://medium.com/ibm-watson/using-advanced-dialog-features-in-watson-conversation-5c0cea89e9b3>. Acesso em 26 maio de 2019.

TURING, Alan M. Computing machinery and intelligence. Mind, v. 59, n. 236, p. 433-460, 1950.

VERGARA, Sebastian; EL-KHOULY, Mohamed; TANTAWI, Mariam El; SHIREESH, Marla; SRI, Lak. Building Cognitive Applications with IBM Watson Services: Volume 7 Natural Language Understanding. p. 112, 2017.

VOORHEES, Ellen M; DANG, Hoa Trang. Overview of the TREC 2005 Question Answering Track. p. 12, 2005.

WATSON ASSISTANT. Disponível em: <https://console.bluemix.net/docs/services/conversation/index.html#sobre>. Acesso em 29 agosto de 2018.

WATSON DISCOVERY, Sobre. Disponível em: < https://console.bluemix.net/docs/services/discovery/index.html#sobre>. Acesso em 12 novembro de 2018.

WATSON DISCOVERY, About. Disponível em: < https://cloud.ibm.com/docs/services/discovery?topic=discovery-about>. Acesso em 29 abril de 2019.

WATSON NATURAL LANGUAGE UNDERSTANDING. Disponível em: < https://console.bluemix.net/docs/services/natural-language-understanding/index.html#about >. Acesso em 18 novembro de 2018.

WATSON TONE ANALYZER. Disponível em: < https://console.bluemix.net/docs/services/tone-analyzer/index.html#about >. Acesso em 14 novembro de 2018.

# APÊNDICE A – FORMULÁRIO DE TESTES DO CHATBOT

Questionário aplicado com os usuários de teste antes da interação com o *chatbot.*

1. Descreva com suas palavras qual pergunta será feita ao Chatbot? Ou o problema do usuário que será simulado.
2. Na sua análise essa pergunta é sobre qual assunto? Em qual categoria ela se enquadraria?
3. O chatbot respondeu corretamente a pergunta?
   1. Sim
   2. Não
   3. Em partes
4. Se o chatbot errou, qual seria a resposta correta?
5. Em uma escala de 1 a 5, qual a nota você atribui a performance do chatbot?
   1. 1 (Péssima)
   2. 2
   3. 3
   4. 4
   5. 5 (Ótima)

# APÊNDICE B – FORMULÁRIO DE AVALIAÇÃO DO CHATBOT

Questionário aplicado com os usuários, depois da interação com o *chatbot* para avaliar o desempenho. Com exceção da primeira questão, todos as outras utilizam a escala de Likert (LIKERT, 1932) para as respostas possíveis do usuário.

1. Você já tinha conversado com um chatbot antes?
   1. Sim
   2. Não
2. Você aprendeu alguma coisa com essa conversa?
3. O Chatbot conseguiu entender as suas perguntas e respostas?
4. A sua dúvida ou problema foi resolvido após a conversa?
5. Você conversaria com o chatbot novamente?
6. Comentário Livre.

1. https://www.apple.com/br/siri/ [↑](#footnote-ref-2)
2. https://www.microsoft.com/pt-br/windows/cortana [↑](#footnote-ref-3)
3. https://developer.amazon.com/alexa [↑](#footnote-ref-4)
4. https://www.luis.ai/home [↑](#footnote-ref-5)
5. https://dialogflow.com [↑](#footnote-ref-6)
6. https://trec.nist.gov/ [↑](#footnote-ref-7)
7. http://uima.apache.org [↑](#footnote-ref-8)
8. https://hadoop.apache.org [↑](#footnote-ref-9)
9. https://www.lemurproject.org/indri/ [↑](#footnote-ref-10)
10. http://lucene.apache.org [↑](#footnote-ref-11)
11. https://www.ibm.com/watson/ [↑](#footnote-ref-12)
12. https://www.w3schools.com/html/default.asp [↑](#footnote-ref-13)
13. https://www.w3schools.com/css/ [↑](#footnote-ref-14)
14. https://www.w3schools.com/js/default.asp [↑](#footnote-ref-15)
15. https://nodejs.org/en/ [↑](#footnote-ref-16)
16. https://github.com/watson-developer-cloud/node-sdk [↑](#footnote-ref-17)
17. https://expressjs.com/pt-br/ [↑](#footnote-ref-18)
18. https://www.npmjs.com/package/body-parser [↑](#footnote-ref-19)
19. https://www.npmjs.com/package/watson-developer-cloud [↑](#footnote-ref-20)