IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Álvaro da Silva Fonseca

novembro | 2019

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

SALESFORCE E INTEGRAÇÃO COM SISTEMAS EXTERNOS

ÁLVARO DA SILVA FONSECA

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

NOVEMBRO/2019

Ficha de identificação

Aluno

Nome: Álvaro Da Silva Fonseca

N.º de aluno: 1012577

Curso: Engenharia Informática

Ano Letivo: 2018/2019

Instituição de Ensino

Escola: Escola Superior de Tecnologia e Gestão – Instituto Politécnico da Guarda

Morada: Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Telefone: 271 220 120

Orientador de Estágio: José Quitério Figueiredo

Instituição do Estágio

Nome: LOBA

Morada: Av. Rainha Dona Amélia, 6300-749 Guarda

Telefone: 271 213 008

Supervisor do Estágio: Samuel Alves

Grau Académico: Mestre

Data de início do estágio: 16 de julho de 2019

Data de fim do estágio: 10 de setembro de 2019

Lista de siglas e acrónimos

IBM (*International Business Machines*)

WEB (*World Wide Web*)

PALOP (Países Africanos de Língua Oficial Portuguesa)

API (*Application Programming Interface*)

IOT (*Internet of Things*)

CRM (*Customer relationship management*)

HTML (*HyperText Markup Language*)

CSS (*Cascading Style Sheets*)

SDK (*Software Development Kit*)

JSON (*JavaScript Object Notation*)

Agradecimentos

No decorrer do desenvolvimento deste projeto recebi muitos apoios, dos quais foram essenciais para a concretização do mesmo.

Primeiramente quero agradecer à minha família, nomeadamente a minha mãe e a minha irmã não só pelo apoio durante o estágio, mas também ao longo do meu percurso académico. Apoio esse que me ajudou a manter o foco no meu objetivo de concluir o curso com sucesso.

Agradeço também a todos os meus amigos e irmãos da Igreja Adventista da guarda, em especial ao Vanderley Quaresma e ao José Baia pelo companheirismo e pelos momentos de alegria.

Quero, ainda agradecer aos profissionais da LOBA por estarem sempre disponíveis para ajudar e também pela simpatia. Ainda quero agradecer ao meu supervisor de estágio, Samuel Alves, pelas ideias dadas ao longo do projeto.

Por fim, agradeço ao meu orientador de estágio, José Quitério, pelos seus conselhos e também pela sua disponibilidade.

Resumo

Este relatório descreve o trabalho desenvolvido, em contexto de estágio, no âmbito da unidade curricular Projeto de Informática do curso de Engenharia Informática do Instituto Politécnico da Guarda.

O projeto tem como objetivo o desenvolvimento de um chatbot. O chatbot tem como função realizar o apoio ao cliente em uma página de eventos contruída em Salesforce. O chatbot foi construído utilizando a ferramenta de criação de chatbot da IBM Cloud, a IBM Assistant, e posteriormente integrado ao Salesforce. O chatbot desenvolvido permite aos clientes pesquisarem por eventos existentes num determinado mês, cidade ou data.

Palavras-chave: ChatBot, Salesforce, IBM Assitant.

Abstract

This report describes the work carried out, in internship context, within of the Curricular Unit Informatics Project in the Degree in Computer Engineering of the Guarda Polytechnic Institute.

The project aims to develop a chatbot. Chatbot is designed to provide customer support on an event page built in Salesforce. Chatbot was built using IBM Cloud's chatbot authoring tool, IBM Assistant, and later integrated with Salesforce. The developed chatbot allows customers to search for existing events in a particular month, city or date.

Key Words: ChatBot, Salesforce, IBM Assitant.

Índice Geral

1. Introdução 1
   1.1. Caracterização sumária da instituição 1
   1.2. Objetivos 1
   1.3. Etapas do Estágio 2
   1.4. Estrutura do documento 2
2. Estado da Arte 3
   2.1. Chatbot 3
   2.2. Análise e utilização de chatbots 4
   2.3. Ferramentas para criar chatbots 4
      2.3.1. Dialogflow 6
      2.3.2. IBM Watson Assistant 12
      2.3.3. Outras ferramentas 15
      2.3.3.1. Amazon Lex 15
      2.3.3.2. Salesforce Einstein 16
      2.3.4. Tabela de comparação 16
3. Metodologia e Análise de Requisitos 18
   3.1. Metodologia usada no desenvolvimento do projeto 18
   3.2. Aprendizagem de salesforce 18
   3.3. Tabela de atores e objetivos 19
   3.4. Diagrama de casos de uso 19
4. Tecnologias 20
   4.1. Salesforce 20
   4.3. Apex 22
   4.4. IBM salesforce SDK 22
5. Implementação 23
   5.2. Visualforce Pages 27
   5.3. Classes Apex – Controladores 29
   6. Verificações e validações 32
   7. Conclusões 36
   Referências Bibliográficas 37

Bibliografia 37

   Anexos 39
      Tabela/Objeto “Evento__c” 40
      Anexo A2 41
      Página Visualforce “ProcurarEventosChat.vfp” 41

Anexo A3 46
Controlador “SECONTROLLER.apxc” 46
Anexo A4 57
Métodos “updateEnt_evento” e “updateEnt_cidade” 57
Anexo A5 60
Intents criadas no IBM Watson Assistant 60
Anexo A6 69
Condições definidas para as diferentes formas de pesquisa que o utilizador pode utilizar. 69
Anexo A7 84
Janela do botão “?” 84

Índice de figuras

Figura 1- DialogFlow agente 6
Figura 2- DialogFlow Intent 7
Figura 3- DialogFlow entity 8
Figura 4- DialogFlow actions 8
Figura 5- DialogFlow Prompts 9
Figura 6- DialogFlow Resposta 9
Figura 7- DialogFlow demonstração I 10
Figura 8- DialogFlow demostração II 11
Figura 9- IBM Watson Assistant intent 12
Figura 10- IBM Watson Assistant entity 13
Figura 11- IBM Watson Assistant actions e resposta 14
Figura 12- IBM Watson Assistant demonstração 15
Figura 13- Diagrama de casos de uso 19
Figura 14- Intent #PedirEventoMes e exemplos 24
Figura 15- Entity @Meses 25
Figura 16- Condição para a pesquisa dos eventos por mês I 26
Figura 17- Condição para a pesquisa dos eventos por mês II 27
Figura 18- Interface do chatbot 28
Figura 19- Chatbot na página dos eventos 32
Figura 20- Chatbot com Interface em inglês 33
Figura 21- Resposta a entrada "ola" 33
Figura 22- Chatbot: Todos os eventos disponíveis 34
Figura 23- Chatbot: eventos para um cidade e mês 34
Figura 24- Chatbot: detalhes de um evento 35
Figura 25- Chatbot: Entrada inválida do utilizador 35
Figura 26- Intent "welcome" 60
Figura 27- Intent "PedirEventosPorCidades" 61
Figura 28- Intent “PedirEventosDisponiveis” 62
Figura 29- Intent “PedirEventosDisponiveis” 63
Figura 30- Intent “PedirEventosDisponiveis” 63
Figura 31- Intent “PedirEventosAno” 64
Figura 32- Intent “PedirEventos” 65
Figura 33- Intent “PedirEventos” 66
Figura 34- Intent “DetalhesEventos” 67
Figura 35- Intent “Agradecimento” 68
Figura 36- Condições: Pesquisa pelo ano I 69
Figura 37- Condições: Pesquisa pelo ano II 70
Figura 38- Condições: Pesquisa pelo dia e mês I 71
Figura 39- Condições: Pesquisa pelo dia e mês II 72
Figura 40- Condições: Pesquisa pelo dia e mês II 72
Figura 41- Condições: Pesquisa pelo mês I 73
Figura 42- Condições: Pesquisa pelo mês II 74
Figura 43- Condições: Pesquisa pelo mês III 74
Figura 44- Condições: Pesquisa pela cidade I 75
Figura 45- Condições: Pesquisa pela cidade II 75

Figura 46- Condições: Pesquisa pela cidade III 76
Figura 47- Condições: Pesquisa pelo mês e ano 77
Figura 48- Condições: Pesquisa pela cidade e mês I 78
Figura 49- Condições: Pesquisa pela cidade e mês II 78
Figura 50- Condições: Pesquisa pela cidade e mês III 79
Figura 51- Condições: Pesquisa pela cidade e ano 80
Figura 52- Condições: Pesquisa pela cidade, mês e ano 81
Figura 53- Condições: Agradecimentos 82
Figura 54- Condições para quando não se cumpre nenhuma condição 83
Figura 55- Janela de ajuda ao utilizador 84

Índice de Tabelas

Tabela 1- Comparação entre as ferramentas de criação de chatbot 16
Tabela 2- Atores e os respetivos objetivos no sistema 19
Tabela 3- Serviços da plataforma Salesforce 20
Tabela 4- Intent #PedirEventosMes e os seus exemplos 24
Tabela 5- Tabela/Objeto "Evento__c" 40

1. Introdução

O presente documento apresenta todo o trabalho realizado durante dois meses, na empresa LOBA. O trabalho consistiu em desenvolver um chatbot de apoio ao cliente para uma página de eventos em Salesforce. O chatbot desenvolvido tem como função mostrar os eventos disponíveis na página consoante as entradas dos clientes.

1.1. Caracterização sumária da instituição

A antiga Dom Digital - Novas Tecnologias de Informação Lda, que agora faz parte do grupo LOBA, é uma empresa que presta serviços tendo como base a infraestrutura da Internet. Foi fundada em janeiro de 1997 na cidade da Guarda. Foca a sua atividade nas áreas de desenvolvimento web e mobile e marketing digital, prestando serviços e criando soluções com resultados em Portugal, Espanha e PALOP (Países Africanos de Língua Oficial Portuguesa). Tornou-se a primeira parceira da Salesforce em Portugal, tendo grande parte dos seus projetos desenvolvidos com base em nessa plataforma [1].

1.2. Objetivos

Os objetivos definidos pela LOBA para este projeto são os seguintes:

- Capacidade de análise critica das situações
- Dominar os conceitos de Salesforce
- Ligações com API e Salesforce
- Elaboração de relatórios/documentos com qualidade do ponto de vista estrutural de redação e de apresentação
- Desenvolvimento do chatbot.

1.3. **Etapas do Estágio**

Inicialmente, foram realizadas pesquisas acerca das várias ferramentas existentes para a criação de chatbots de forma a avaliar as vantagens e desvantagens das mesmas. De seguida, definiu-se os diálogos e as diferentes formas de pesquisa que os clientes poderiam utilizar para procurar pelos eventos. Por último, utilizou-se ferramentas de desenvolvimento da plataforma Salesforce, tais como Apex e o Visualforce. Estas ferramentas permitiram implementar as funcionalidades do chatbot que posteriormente foi integrado a uma página de eventos.

1.4. **Estrutura do documento**

No capítulo 1, Introdução, apresenta a caracterização sumaria da Instituição de acolhimento, um resumo do projeto e os objetivos. No capítulo 2, Estado de Arte, é descrita a pesquisa efetuada sobre as ferramentas de criação de chatbot e também é feito um resumo do que é um chatbot. De seguida, no capítulo 3, fica um resumo da metodologia utilizada no desenvolvimento do projeto, da aprendizagem feita para a realização do projeto e também pode ver-se a tabela de atores e o diagrama de casos de uso. No capítulo 4, mostra as tecnologias essenciais para o desenvolvimento do projeto e uma breve descrição das mesmas. Posteriormente, no capítulo 5, descreve a implementação da solução proposta e mostra como foi criado o chatbot. No penúltimo capítulo é feita a verificação e validação, mostra imagens do projeto possibilitando a validação das funcionalidades implementadas. Por fim, no capítulo 7, é feita uma conclusão do trabalho realizado. No final do documento encontram-se as referências bibliográficas e os anexos citados ao longo do documento.

2. Estado da Arte

Neste capítulo é feita uma explicação sobre o que é um chatbot e também é descrita a pesquisa realizada acerca das diferentes ferramentas para criação de chatbots.

2.1. **Chatbot**

Com o mercado da internet em grande expansão e cada vez mais diversificado, a competitividade entre as empresas para prestar o melhor e mais rápido serviço a uma grande quantidade de clientes, que estão sempre em crescimento e que cada vez mais dão mais valor a qualidade do serviço, torna-se uma missão quase impossível. A solução para essas entidades passa pela utilização de chatbots. Os chatbots permitem a estas empresas a diminuição do tempo de resposta aos clientes, seja numa compra, na resolução de um problema ou num pedido de informação, possibilitando as equipas de suporte ganharem tempo para resolução de problemas mais complexos.

Chatbot são robôs desenvolvidos para se comunicarem com os utilizadores e fazem uso da inteligência artificial para tornar essa comunicação mais próxima da realidade. A utilização dos chatbots nos dias que correm permite que o apoio ao cliente aconteça de forma automática diminuindo a intervenção de humanos. Esse apoio passa por responder a pedidos de informação por parte dos utilizadores ou até mesmo resolver um determinado problema.

A capacidade de os robôs entenderem a intenção dos utilizadores é essencial para construir chatbots funcionais. O uso das tecnologias como deep learning e machine learning tornou-se fundamental para que os robôs tenham a capacidade de interpretar a intenção dos utilizadores de forma a fornecer soluções mais exatas para as diferentes situações [2].

2.2. **Análise e utilização de chatbots**

Durante a análise feita aos diferentes serviços de comércio eletrónico é de se destacar que muitos deles não possuem um chatbot e aqueles que têm não o apresentam na página inicial do site.

Durante o nosso dia a dia entramos com frequência em diversos sites, seja para obter informação ou realizar uma compra, e muitas das vezes percorremos diversos menus e páginas até encontrarmos a informação desejada. A solução para este problema seria a implementação de um chatbot na página inicial desses serviços, o que possibilitaria aos utilizadores um acesso mais rápido a informação.

No momento atual em que vivemos em que o acesso a informação é feito de várias formas, o uso do chatbot vem trazer uma nova forma de aceder a essa mesma informação.

2.3. **Ferramentas para criar chatbots**

Durante a fase de pesquisa pude concluir que a maioria das ferramentas analisadas fazem uso da tecnologia de Processamento de Linguagem Natural. Essa tecnologia possibilita ao sistema de chatbot captar o que lhe é escrito, compreender o significado e dar uma resposta numa linguagem que os utilizadores entendem. Reparei que na maioria das ferramentas analisadas, a criação de chatbots é feita definindo os seguintes componentes básicos:

- **Agente**

Um agente é um módulo de processamento de linguagem natural, que é criado e utilizado na aplicação ou serviço que possuímos. Os agentes compreendem as nuances da linguagem humana e transformam as entradas do utilizador em dados estruturados que as nossas aplicações e serviços conseguem entender. Um agente pode ser treinado para cenários esperados que envolvem perguntas e solicitações dos utilizadores [3].

- **Intent**

Para cada agente é definido várias intents. As intents representam a intenção do utilizador e para cada intent, são definidos vários exemplos de possíveis entradas que o utilizador pode usar para acionar a intent [4].

- **Entities**

Entities são mecanismos usados para identificar e extrair dados úteis de entradas de linguagem natural. As intents permitem que o agente processe a motivação subentendida em uma entrada específica do cliente. No entanto, as entities são usadas para captar informações específicas mencionadas pelos utilizadores, incluindo endereços, nomes de produtos ou quantidades com unidades e outros dados.

Todos os dados importantes que se quiser extrair da entrada do utilizador precisam de ter uma entity correspondente. Por exemplo, se o utilizador diz "Eu quero uma laranja", "laranja" pode ser um valor de parâmetro para uma entity designada fruta [5].

- **Ações**

Podem ser definidas para cada intent. Quando a entrada de um utilizador corresponde a uma intent, o sistema fornece a possibilidade de efetuar uma determinada ação.

- **Resposta**

As respostas podem ser definidas como respostas de texto, fala ou visuais. Podem fornecer respostas ao utilizador, pedir por mais informações ou encerrar uma conversa [6].

2.3.1. **Dialogflow**

É uma plataforma adquirida pela Google e permite criar interfaces conversacionais baseados em voz ou texto (como chatbots) para sites, aplicações móveis, plataforma de messaging e dispositivos IoT (Internet of Things). Utiliza machine learning e a tecnologia de processamento de linguagem natural que permite criar interações naturais com os utilizadores. O Dialogflow pode analisar vários tipos de entrada dos clientes, incluindo entradas de texto ou áudio. Também pode responder aos utilizadores de várias maneiras, seja por meio de texto ou com fala sintética [7].

Como forma de demonstrar os passos para contruir um chatbot usando o Dialogflow, as figuras abaixo apresentam um pequeno exemplo de um chatbot para mostra filmes segundo uma determinada categoria. Primeiro criou-se um agente, que é demonstrado na figura 1.

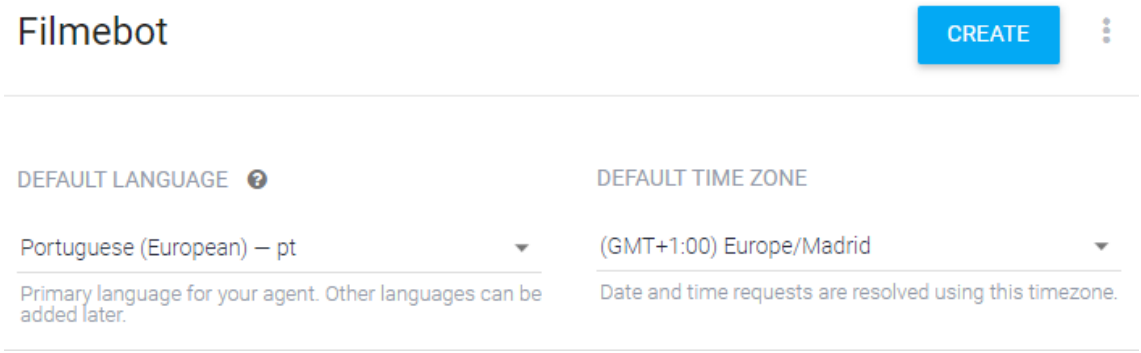

Figura 1- DialogFlow agente

Depois de criar o agente, definiu-se a intent “Pedir Filme” e também foram adicionados exemplos de frases. Os utilizadores ao mencionarem estes exemplos ou frases parecidas, a intent vai ser acionada. Na figura 2, é apresentado a intent mencionada anteriormente.

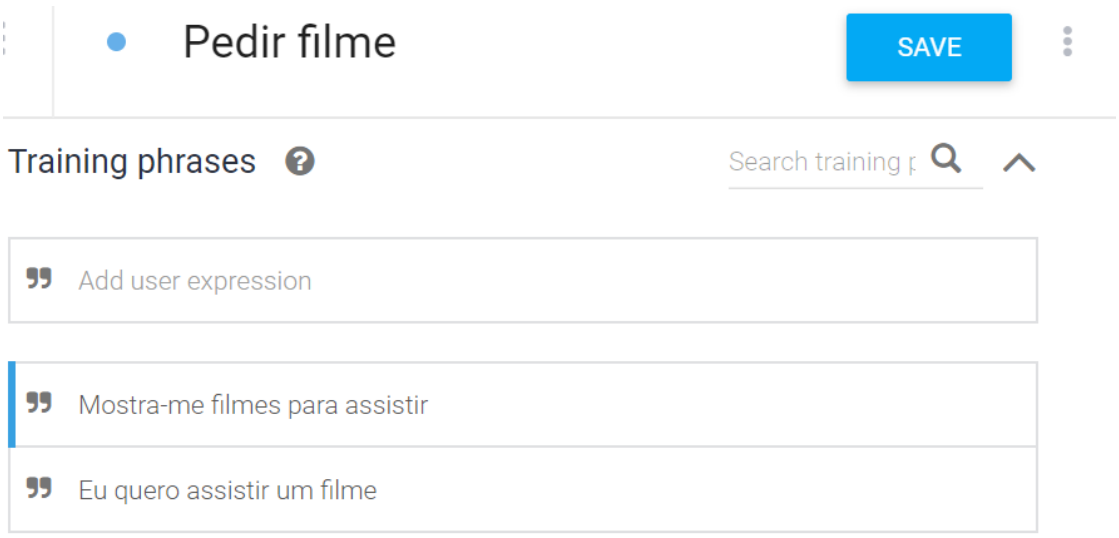

Figura 2- DialogFlow Intent

Como o objetivo do chatbot exemplo é mostrar filmes de uma determinada categoria, então é importante extrair esse tipo de informação a partir das entradas dos utilizadores. A figura 3, mostra a entity “CategoriaFilme” e os seus valores.

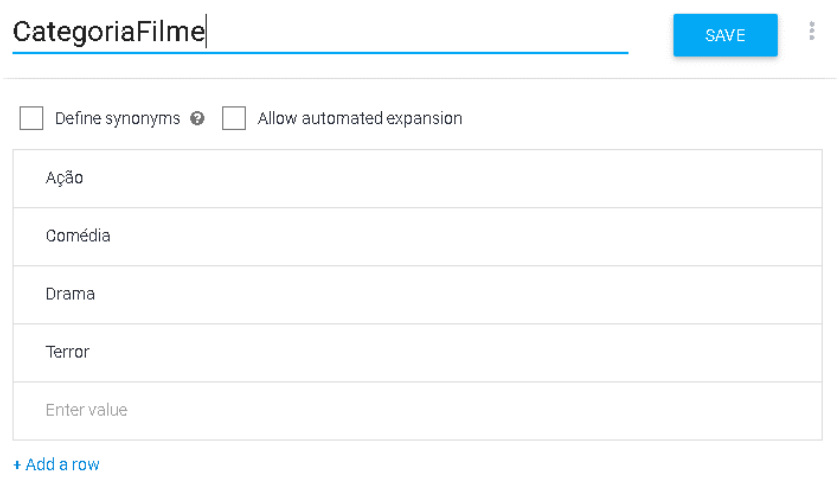

Figura 3- DialogFlow entity

Quando a intent "Pedir Filme" é acionada, uma ação é realizada. Se a entrada do utilizador contém algum dos valores da entity "CategoriaFilme" esse valor fica contido no "Parameter" e pode ser utilizado para formular uma resposta através da utilização do "value" ($CategoriaFilme). Caso a entrada dos utilizadores não possua valores da entity "CategoriaFilme", a resposta para o utilizador passa a ser a mensagem contida no "PROMPTS", como demonstram a figura 4 e 5.

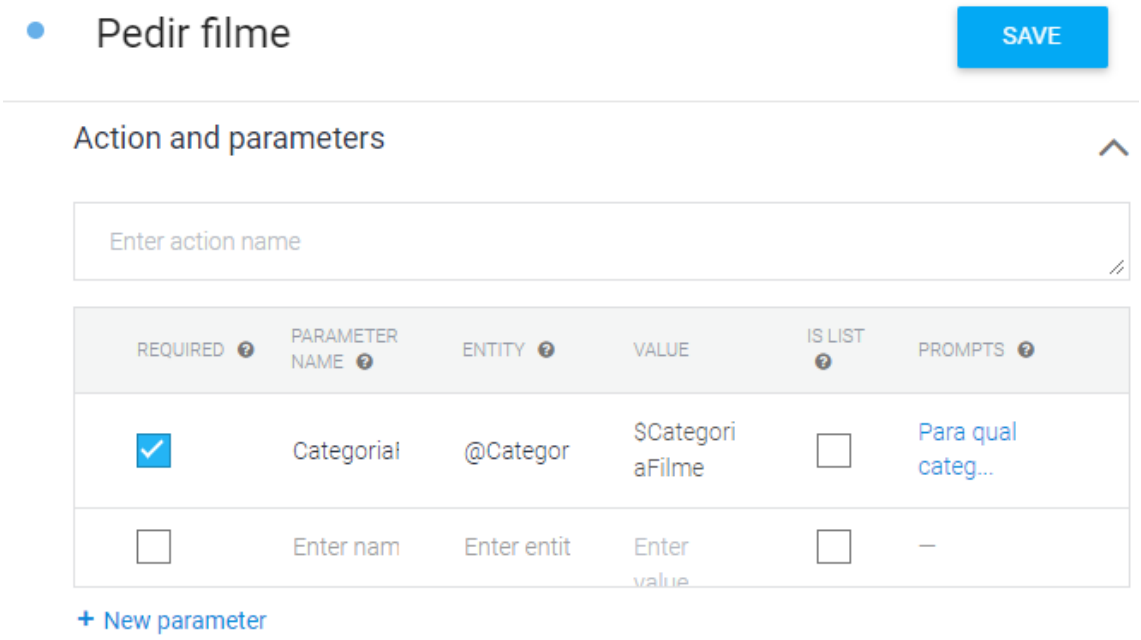

Figura 4- DialogFlow actions

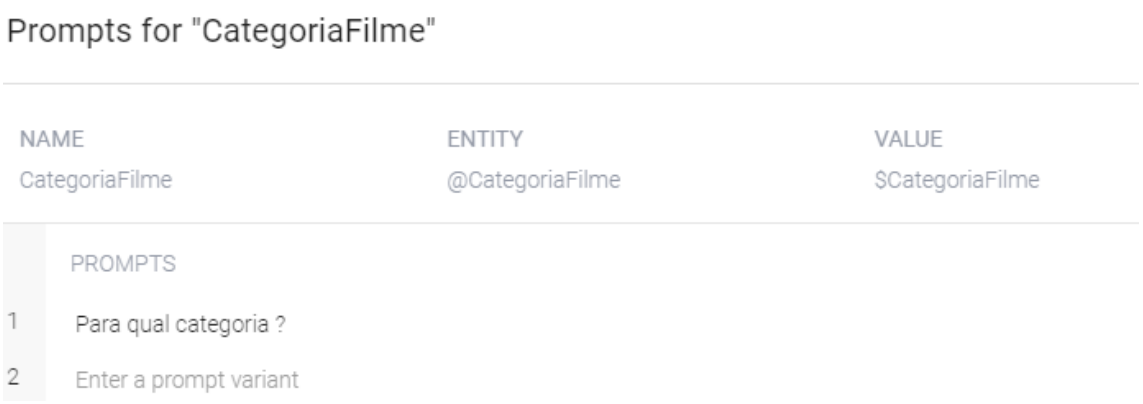

Figura 5- DialogFlow Prompts

A categoria do filme escrita pelo utilizador como mencionado anteriormente, pode ser utilizada para formular uma reposta através do "value" ($CategoriaFilme), como é apresentado na figura 6.

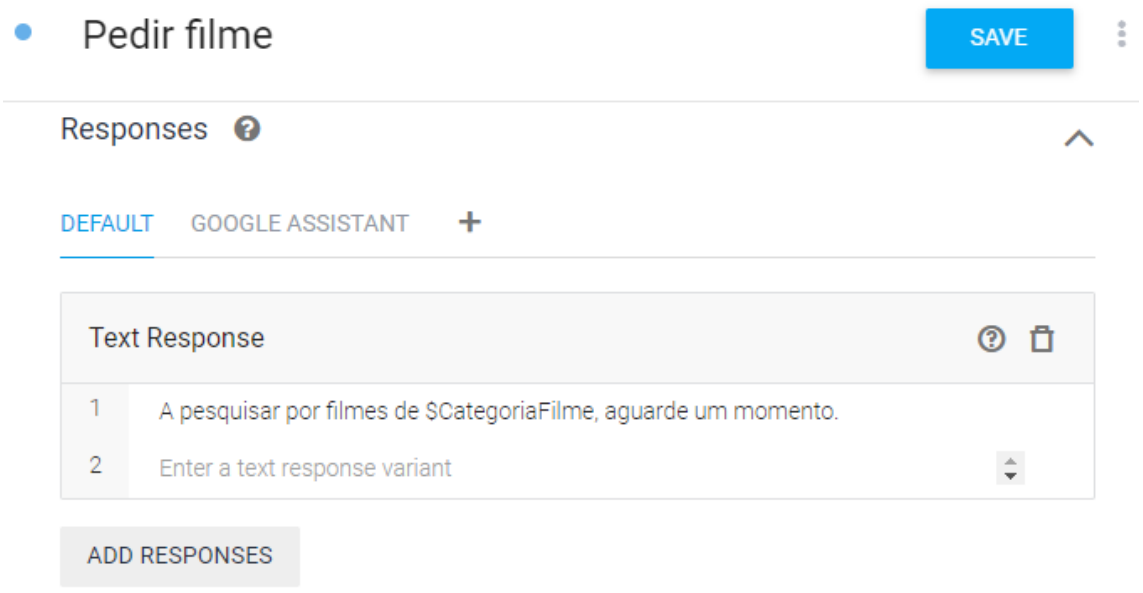

Figura 6- DialogFlow Resposta

Na demonstração abaixo, na figura 7, a entrada ‘quero assistir um filme’, aciona a intent “Pedir Filme” e a ação tomada foi de mostrar a reposta do “Prompt”, dado que o utilizador não mencionou nenhum dos valores da entity “CategoriaFilme”. A resposta do “Prompt” leva o utilizador a especificar a categoria do filme que é utilizada na resposta ‘A pesquisar por filmes de Drama, aguarde um momento’, permitindo um diálogo mais dinâmico e mais real.

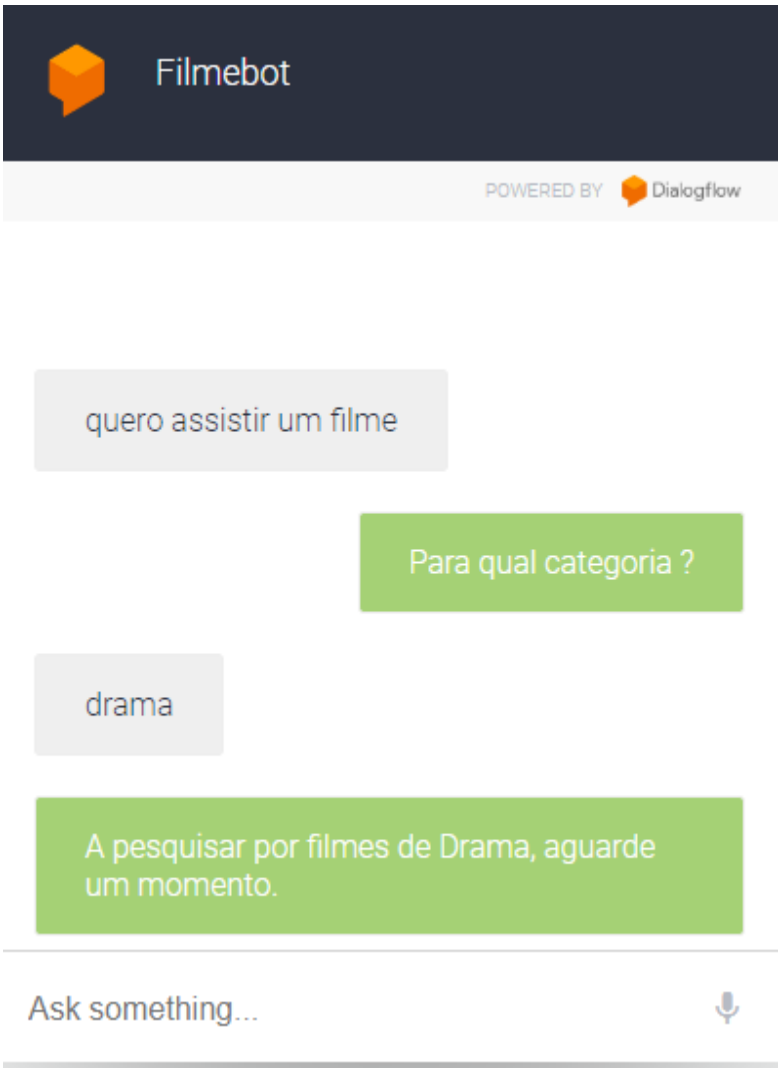

Figura 7- DialogFlow demonstração I

Na figura 8, a entrada do utilizador faz referência a intent “PedirFilme” e a um dos valores da entity “CategoriaFilme”, logo o “Prompt” não é acionado e o utilizador recebe a resposta final contendo a categoria mencionada pelo mesmo.

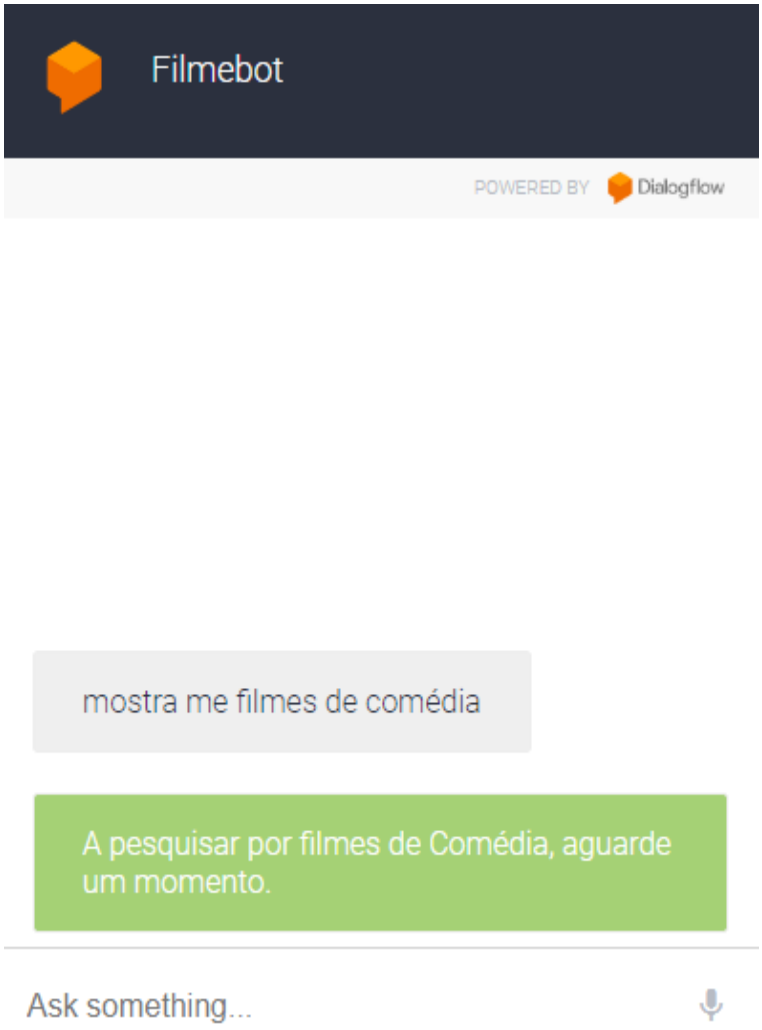

Figura 8- DialogFlow demostração II

No exemplo demostrado acima, o Dialogflow mostrou-se ser uma boa ferramenta para construção de chatbots. O tempo de utilização foi curto e nem todas as funcionalidades foram exploradas por completo, mas a ferramenta possui pontos positivos como, a possibilidade de configurar o agente em várias línguas, ou a funcionalidade de extrair elementos das entradas dos utilizadores e usar esses mesmos valores para formular uma resposta.

2.3.2. IBM Watson Assistant

A plataforma da Cloud da IBM fornece vários serviços, dentre os quais está o IBM Watson Assistant, que é um sistema de perguntas e respostas que fornece uma interação de diálogo entre o sistema de conversação e os utilizadores. O Assistant é alimentado pelo Machine Learning e Processamento de Linguagem Natural. Permite integrações em várias plataformas [8].

As figuras a seguir mostra um exemplo simples da criação de um chatbot utilizando o IBM Assistant. O chatbot será para pesquisa de eventos para uma determinada cidade.

Como a intenção do utilizador será de pesquisar eventos para uma determinada cidade, criou-se uma intent com exemplos que o utilizador pode utilizar para acionar essa mesma intent. A figura 9, mostra a intent criada e os exemplos que os utilizadores podem usar para fazer referência a mesma.

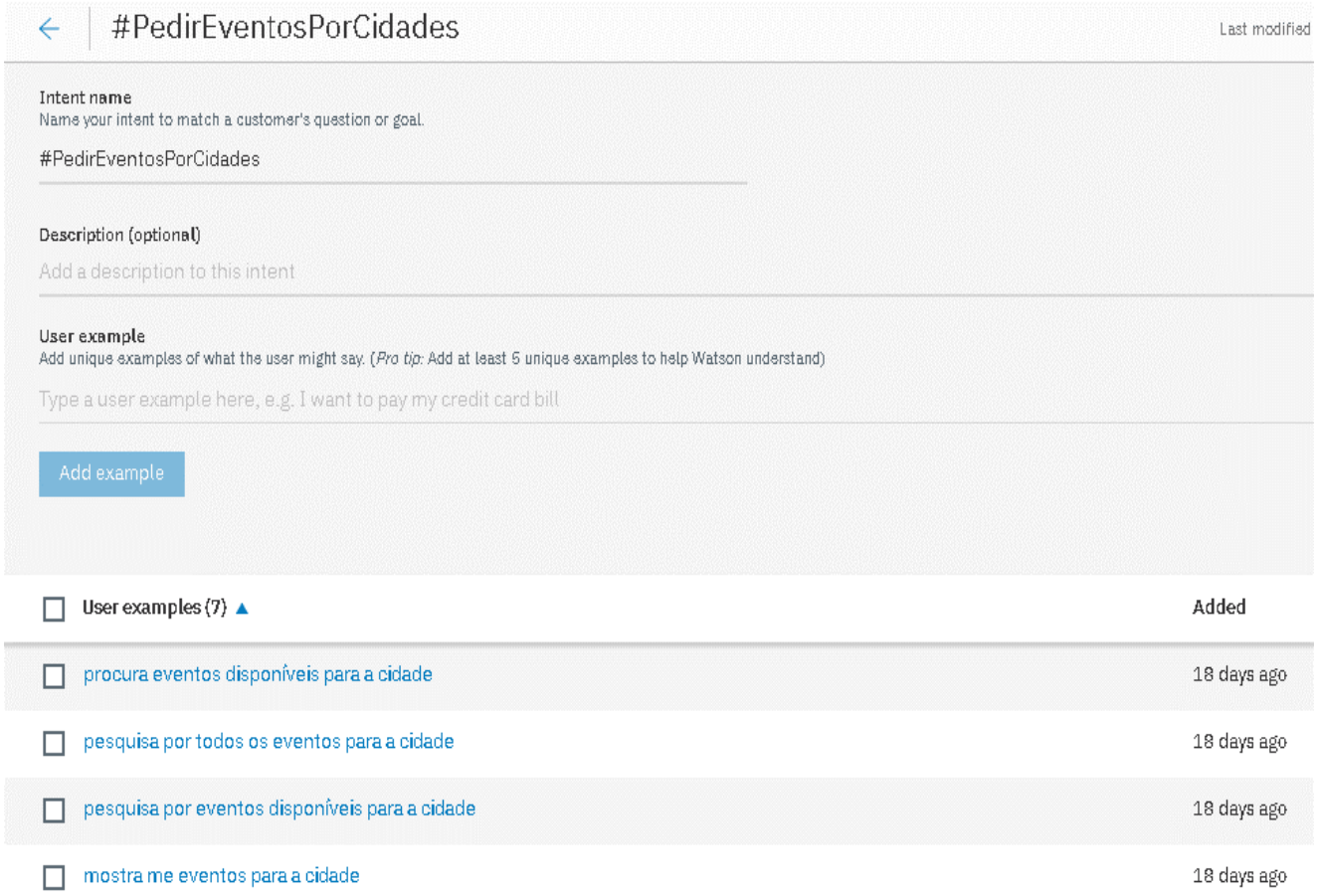

Figura 9- IBM Watson Assistant intent

Como o objetivo do chatbot é para auxiliar na pesquisa de eventos para uma determinada cidade é de extrema importância extrair as cidades referidas pelos utilizadores, logo criou-se uma entity designada por "@cidade", que é apresentada na figura 10.

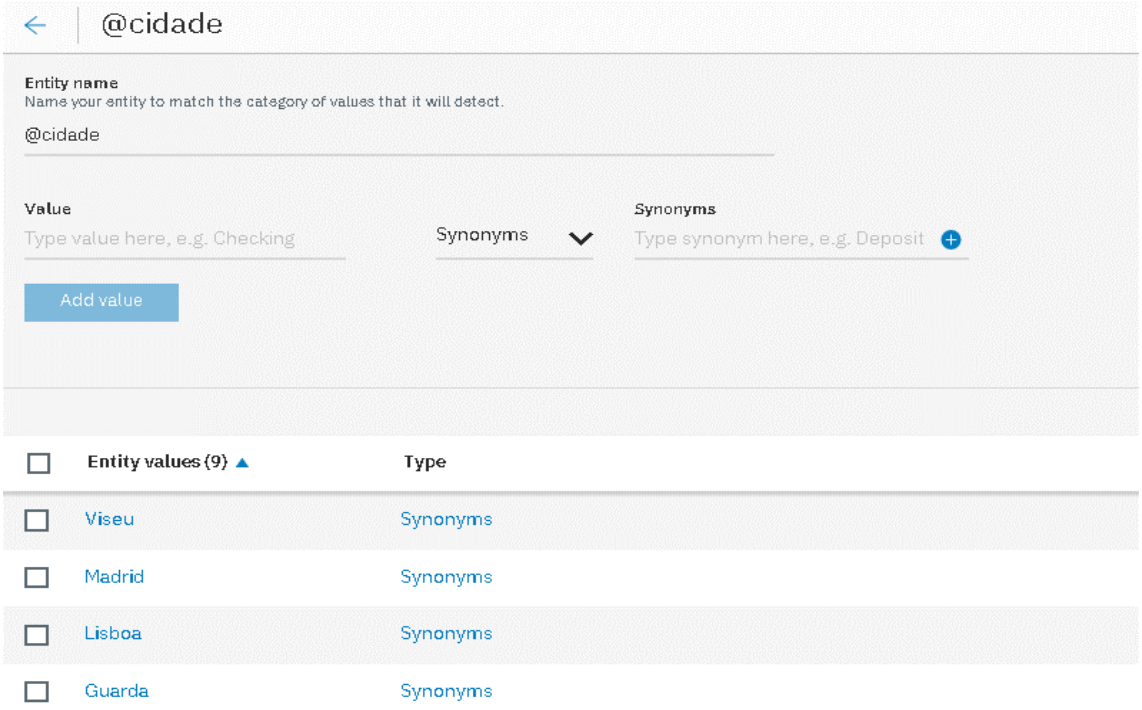

Figura 10- IBM Watson Assistant entity

O objetivo passa por quando o utilizador mencionar um dos exemplos da intent("#PedirEventosporCidade") e da entity("@cidade") é desencadeada uma ação que vai ser uma resposta contendo o nome da cidade referida pelo utilizador. Por outras palavras, quando o sistema identificar que a entrada do utilizador contém um dos exemplos da intent("#PedirEventosporCidade") e da entity("@cidade"), uma resposta do tipo texto é dado ao utilizador. No IBM Assistant é possível definir condições com as entities e as intents, o que permite definir exatamente a forma como queremos que o utilizador realize uma determinada pesquisa. A figura 11, apresenta a condição definida.

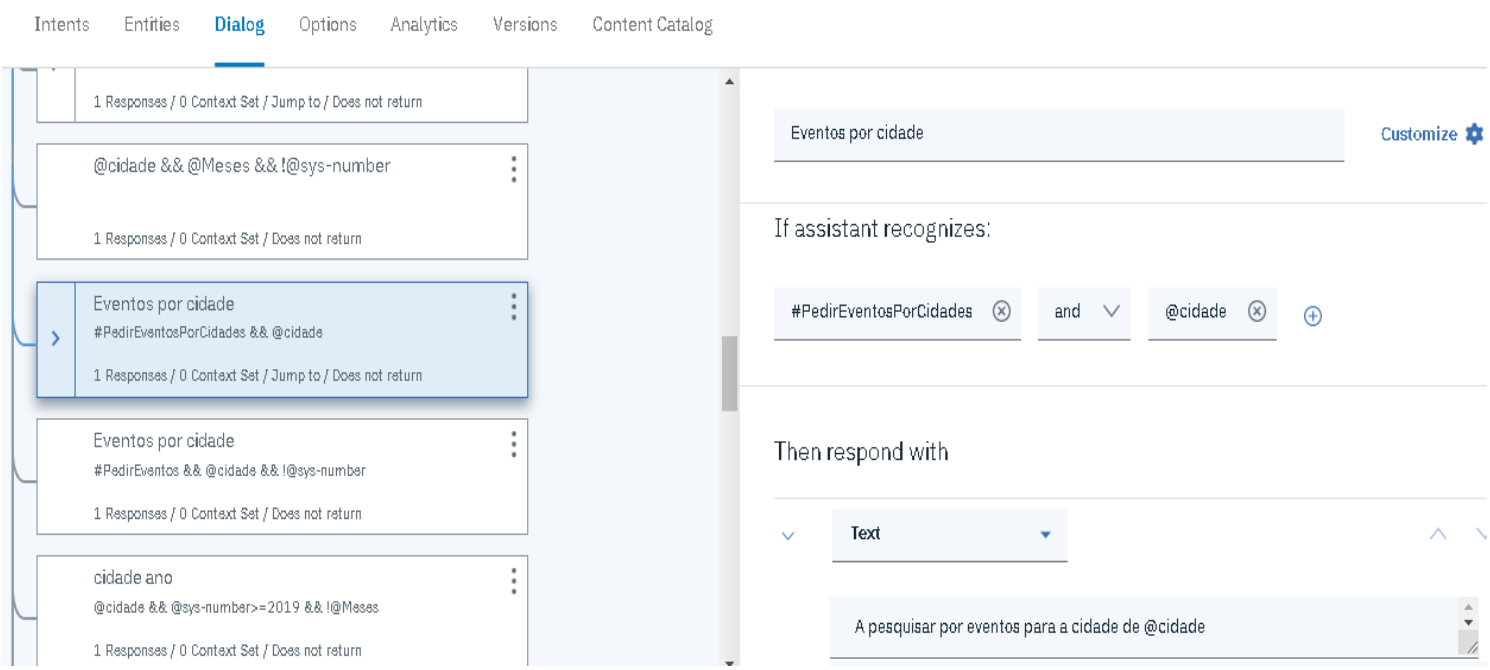

Figura 11- IBM Watson Assistant actions e resposta

Na figura 12, vemos que o sistema identificou que a condição definida anteriormente, #pedirEventosCidade and @cidade, foi cumprida. Isto é, a entrada do utilizador contém uns dos exemplos da intent "#pedirEventosCidade" e da entity "@cidade", logo o utilizador é respondido com a resposta de texto definida anteriormente.

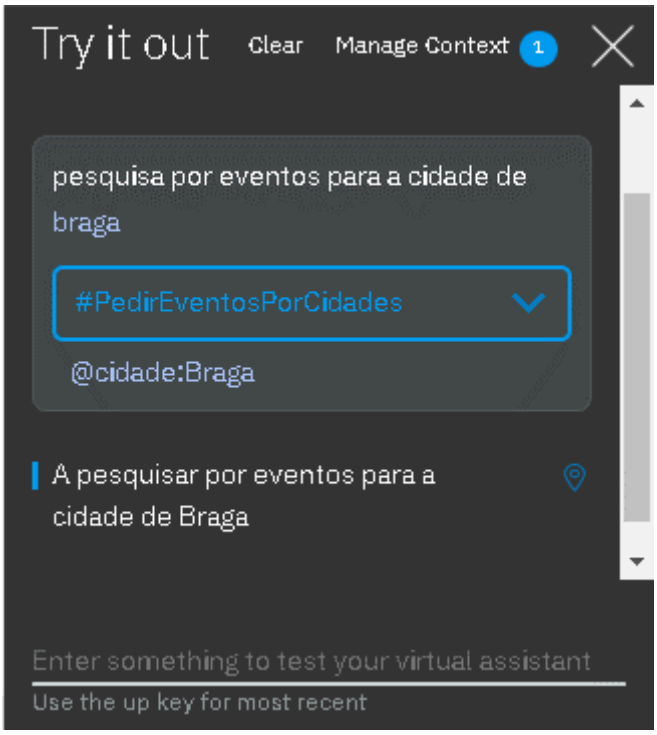

Figura 12- IBM Watson Assistant demonstração

O IBM Watson Assistant, demonstrou ser uma ótima ferramenta de criação de chatbot. Como apresentado anteriormente, esta ferramenta possui funcionalidades importantes para a criação de um chatbot, funcionalidades essas como a possibilidade de definir condições com as intents e as entities, o que permite definir exatamente como a entrada do utilizar aciona uma determinada resposta. Essa ferramenta também possui uma área de testes, o que possibilita verificar as condições definidas.

2.3.3. Outras ferramentas

As ferramentas apresentadas neste seção também formam alvo de pesquisa, mas ao contrário das anteriores não foram testadas devido ao facto da ferramenta IBM Assistant ter sido escolhida para a criação do chatbot dado a suas funcionalidades e também por questões de gestão do tempo.

2.3.3.1. Amazon Lex

O Amazon Lex é um serviço para a criação de interfaces de conversa. O Amazon Lex disponibiliza funcionalidades avançadas de aprendizado profundo de Automatic

Speech Recognition (ASR – Reconhecimento automático de fala), para a conversão de fala em texto, e Natural Language Understanding (NLU – Compreensão de linguagem natural), para o reconhecimento da intenção do texto. Permite a integração de chatbots de voz ou texto em dispositivos móveis, aplicações web e serviços de messaging [9].

2.3.3.2. Salesforce Einstein

É uma plataforma que permite criação de chatbots para interagir ou prestar serviços ao cliente, funcionando como um representante de vendas ou agente de apoio. O Einstein possui a tecnologia chamada processamento de linguagem natural (NLP) que permite ao chatbot fornecer respostas automatizadas mais semelhantes à humana. A plataforma Salesforce já vem integrado com esta funcionalidade o que facilita a sua implementação.

Após pesquisa realizada a estas duas ferramentas, verificou-se que levariam mais tempo para se dominar os conceitos da ferramenta, os passos para criação de um chatbot mostram-se mais complexos em relação as outras ferramentas. Isso deve-se não só aos conceitos, que são diferentes, mas também as particularidades das plataformas que fornecem o serviço.

2.3.4. Tabela de comparação

A tabela 1, apresenta uma comparação de diversos recursos importantes para criação de um chat, entre as ferramentas mencionadas anteriormente.

	IBM Watson Assitant	**Salesforce Einstein**	**Amazon Lex**	**Dialogflow**
Natural Language Processing (NLP)	sim	sim	sim	sim
Integração em serviços/aplicações externas	sim	sim	sim	sim
Área de testes	sim	-	sim	sim
Acesso ao serviço	serviço gratuito	serviço gratuito	serviço pago (gratuito durante 12 meses)	serviço gratuito

Tabela 1- Comparação entre as ferramentas de criação de chatbot

Uma ferramenta para criar chabot necessita ter a tecnologia de processamento de linguagem natural, de forma a interpretar o que lhe é escrito e fornecer as melhores soluções. Como é apresentado na tabela acima, esta tecnologia está presente em todos as ferramentas analisadas. A integração com serviços/aplicações externas dessas ferramentas é importante, dado que muita das vezes os chatbots criados com essas ferramentas são utilizados em outras plataformas. A área de testes é necessária para que a verificação do que foi definido na criação do chatbot. O IBM Assistant e o Dialogflow possuem essa área, mas depois da utilização o da IBM Assistant mostrou-se mais completa, dado que especifica todas as intents e entities que são acionadas. Antes da utilização de um serviço é importante saber se essa utilização traz custos adicionais. Todas as ferramentas analisadas são gratuitas, exceto a Amazon Lex que tem o período de utilização gratuita de doze meses.

3. Metodologia e Análise de Requisitos

3.1. Metodologia usada no desenvolvimento do projeto

Para desenvolver um chatbot é necessário seguir um conjunto de processos, melhorando com isso todo o processo de desenvolvimento. Tendo em vista esse objetivo optou-se pela escolha da metodologia de desenvolvimento ágil. Considerou-se o supervisor de estágio como cliente, dado ao seu envolvimento no projeto.

No desenvolvimento ágil utiliza-se uma abordagem de planeamento incremental e muito iterativa. Cada iteração é um miniprojecto, que normalmente dura de 1 a 4 semanas, e inclui todas as fases para a sua implementação como levantamento de recursos e requisitos, desenvolvimento de código, testes e documentação. Ao final de cada iteração deve haver uma entrega ao cliente, que inclua um conjunto de novas funcionalidades, uma nova versão de software. Após essa entrega há um novo processo de comunicação com o cliente e então são definidas quais deverão ser as novas entregas.

Durante o decorrer do estágio, foram realizadas reuniões a cada 15 dias. Nas reuniões mostrava-se o trabalho desenvolvido (funcionalidades, pesquisas). Esse trabalho era analisado pelo supervisor de estágio e a partir dessa análise eram definidas novas tarefas, novas funcionalidades a serem implementadas ou correções que poderiam ser feitas de forma a melhorar o projeto.

3.2. Aprendizagem de salesforce

De forma a obter conhecimentos acerca da tecnologia salesforce utilizou-se a plataforma Trailhead. O Trailhead é uma plataforma online gratuita que permite aprender salesforce de uma maneira eficiente.

Os conteúdos são separados em módulos, que por sua vez estão separados em unidades e a medida que o utilizador responde as perguntas e completa os exercícios de cada unidade, ganha pontos e emblemas. A plataforma também possui as “trilhas”

que são um conjunto de módulos. Elas funcionam como um caminho de aprendizagem guiado para uma necessidade especifica.

Os exercícios práticos no final de cada unidade permitem não só aumentar o conhecimento sobre o salesforce mas também melhorar as competências e desenvolvimento dentro da plataforma.

3.3. **Tabela de atores e objetivos**

A tabela abaixo pretende demonstrar os atores envolvidos no sistema e os seus respetivos objetivos, para além de permitir entender a função de cada um dentro do projeto.

Atores	**Objetivos**
Clientes	Pedir informações sobre os eventos que estão disponíveis.

Tabela 2- Atores e os respetivos objetivos no sistema

3.4. **Diagrama de casos de uso**

Os diagramas de casos de uso descrevem as principais funcionalidades dos sistemas e as iterações dessas mesmas funcionalidades com os utilizadores. Este diagrama mostra as funcionalidades do sistema do ponto de vista do utilizador.

A figura 13 apresenta o diagrama de casos de uso com as principais funcionalidades.

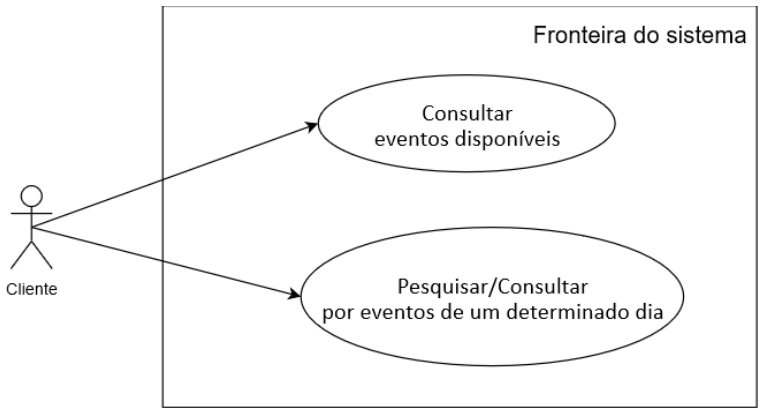

Figura 13- Diagrama de casos de uso

4. Tecnologias

Neste capítulo é feita uma breve descrição das tecnologias utilizadas no desenvolvimento projeto.

4.1. Salesforce

O Salesforce é uma plataforma de CRM (Custumer Relationship management) desenvolvida na cloud, que aproxima as empresas e o cliente. Esta plataforma é composta por vários serviços, como o Sales Cloud, Service Cloud, o Marketing Cloud, Commerce Cloud, e o App Cloud [9].

A tabela abaixo mostra um resumo dos serviços acima identificados.

Serviço	Descrição
Sales Cloud	Possibilita rastrear informações e interações dos clientes, automatiza processos de negócios complexos, mantém todas as informações atualizadas e controla a eficácia das campanhas de marketing.
Service Cloud	É um serviço para atendimento e suporte ao cliente. O serviço possibilita os agentes de apoio ao cliente resolver os problemas do cliente mais rapidamente, fornece aos clientes acesso a respostas para resolver os problemas por conta própria e ajuda a personalizar o serviço.
Marketing Cloud	Permite personalizar o email marketing em grande escala, envolver-se com mensagens móveis, conectar o social ao marketing, vendas e serviços, gerenciar campanhas publicitárias para ajudar na aquisição de clientes e fornecer conteúdo da web personalizado que seja eficiente.
Commerce Cloud	Permite que as empresas gerenciem o comércio digital com soluções integradas para o comércio, ponto de venda e gerenciamento de pedidos. O Commerce Cloud ajuda a lançar novos sites, a criar experiências de clientes, a colocar lojas online e integrar tecnologias de parceiros.
App Cloud	É uma coleção de ferramentas de desenvolvimento que permite criar aplicações que serão executados na plataforma Salesforce. Uma dessas ferramentas é o Force.com, que permite aos administradores e desenvolvedores criarem sites e aplicações com o Apex.

Tabela 3- Serviços da plataforma Salesforce

(**Fonte**: https://searchcustomerexperience.techtarget.com/definition/Salesforcecom)

Na plataforma Salesforce uma aplicação é um conjunto de objetos, campos e outras funcionalidades que dão suporte a um processo de negócios. Os objetos são tabelas da base de dados do Salesforce que armazenam um tipo de informação específico. Existem objetos padrão na plataforma salesforce, como Contas e Contatos, e objetos personalizados. Registos são linhas nas tabelas da base de dados do objeto. Registos são os dados reais associados a um objeto. Campos são colunas nas tabelas da base de dados do objeto. Os objetos padrões e personalizados têm campos [10].

O desenvolvimento e a implementação deste projeto foram realizados nesta plataforma.

4.2. Visualforce

O Visualforce é uma estrutura que usa linguagem de marcação baseada em tags, semelhante ao HTML (HyperText Markup Language), e um conjunto de "controladores padrão" do lado do servidor que tornam as operações básicas da base de dados, como consultas e inserções, muito simples de executar. Permite criar interfaces de utilizador personalizadas e sofisticadas.

Cada tag do Visualforce corresponde a um componente da interface de utilizador, como uma seção de página, uma lista, ou um campo. A linguagem de marcação do Visualforce permite a utilização das componentes do HTML, CSS (Cascading Style Sheets) e JavaScript, dando-lhe uma flexibilidade considerável na forma de implementar a interface de utilizador.

O comportamento dos componentes do Visualforce pode ser controlado pela mesma lógica usada nas páginas padrão do Salesforce, ou por uma classe de controlador escrita no Apex [11].

Esta tecnologia foi utilizada para criar a interface do chatbot no Salesforce.

4.3. Apex

Apex é uma linguagem de programação orientada para objetos que permite os executar instruções de controlo de fluxo e transações nos servidores Salesforce em conjuntos com as chamadas para API.

A sintaxe desta linguagem é parecida com Java. O Apex permite adicionar lógica de negócio à maioria dos eventos do sistema, incluindo cliques de botão, atualizações de registos relacionados, e páginas de Visualforce. O código Apex pode ser inicializado pelos pedidos de serviço Web e por triggers nos objetos.

O Apex proporcionada o acesso direto aos registos que estão na base de dados, e fornece instruções e linguagens como, o Salesforce Object Query Language e o Salesforce Object Search Language para manipular tais registos [12].

Esta tecnologia foi utilizada para criar os controladores e classes do projeto.

4.4. **IBM salesforce SDK**

O IBM Salesforce SDK (Software Development Kit) fornece suporte a todos aqueles que procuram uma maneira mais fácil de integrar os serviços da IBM Watson no ambiente Salesforce.

A utilização dos serviços da cloud da IBM leva ao uso de centenas de linhas de código do Apex. Mas com o uso deste SDK é possível realizar o mesmo com apenas cinco linhas de código. Os dados são enviados e recebidos utilizando modelos de objeto.

O SDK oferece bibliotecas para os dois serviços mais populares do Watson, o Watson Assistant e o Watson Discovery. Está disponível no GitHub e pode ser implantado numa organização do Salesforce.

5. Implementação

Através do estudo e escolha das tecnologias mais eficientes para a implementação do projeto e da análise dos requisitos necessários, foram reunidas as condições favoráveis a uma implementação mais eficaz. No decorrer do desenvolvimento do projeto, tentou-se sempre encontrar formas de facilitar e simplificar não só a implantação das funcionalidades, mas também a utilização por parte do utilizador.

Este projeto foi realizado na perspetiva de completar um outro projeto desenvolvido por outros estagiários e que consistiu em criar um website de eventos. Então, durante a fase de implementação foram utilizados as tabelas/objetos e os seus respetivos dados do projeto mencionado. Utilizou-se o objeto designado de "Evento__c", e que se encontra no anexo A1, na página 40.

5.1. IBM Watson Assistant

Com a ferramenta IBM Watson Assistant criou-se um chatbot na plataforma da IBM Cloud, com todas as intents, entities, ações e respostas necessárias. O chatbot criado identifica quando o utilizador pesquisa pelos eventos mencionando a cidade dos eventos, o mês de realização do evento, o ano, o mês e o ano, a cidade e o mês, a cidade e o ano, a cidade o mês e o ano dos eventos, ou quanto o utilizador procura por todos os eventos disponíveis. O utilizador também pode pedir detalhes de um determinado evento, mencionando o nome do mesmo. Para cada uma dessas situações o utilizador é respondido de uma forma diferente, o que permite saber que tipo de informação está a ser pedida.

5.1.1. Intents e entity

Antes de desenvolver a interface do chatbot definiu-se, no IBM Watson Assistant, as intents, as entity, e posteriormente as ações e as repostas, termos esses que se encontram referenciados na página 9 e 10. As intents criadas encontram-se no anexo A5, na página 60.

As intents definem as intenções do utilizador, e existem várias formas de acionar uma determinada intent. Na tabela 3 e na figura 14 pode-se ver a intent #PedirEventosMes e os vários exemplos que o utilizador pode usar para acionar a mesma.

Intent	**Exemplos**
#PedirEventosMes	• Eventos para o mês de • Mostra me eventos para o mês de • Mostra me todos os eventos para o mês de • Quais eventos tem para o mês de • Que eventos há no mês de

Tabela 4- Intent #PedirEventosMes e os seus exemplos

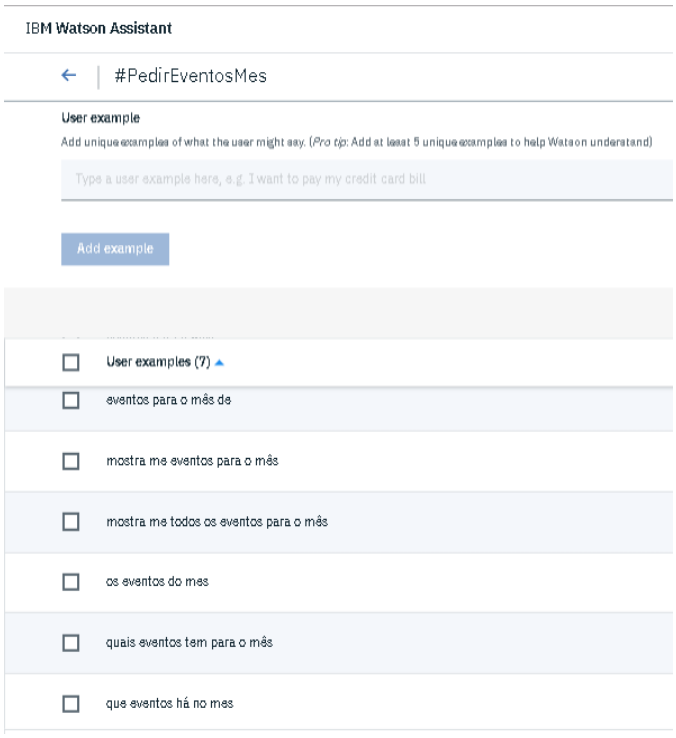

Figura 14- Intent #PedirEventoMes e exemplos

As entities permitem extrair informações importantes que o utilizador digita. o IBM Assistant fornece diversas intities importantes. Intities como, a @sys-date e o @sys-number. A @sys-date permite extrair datas de frases ou palavras como," hoje", "sexta-

feira", ou "12 de dezembro" e identificar quando o utilizador menciona uma determinada data. O @sys-number extrai os números mencionados pelos de utilizadores (por exemplo, 21 ou vinte e um). O valor extraído é representado como sequência numérica.

Dado que uma das formas de pesquisa dos eventos é pelo mês, então é importante identificar quando um determinado mês é mencionado pelo utilizador. Logo, definiu-se a entity @Meses, que podemos ver na figura 15. Também se criou outras entities como a @cidade, que permite identificar as cidades mencionadas pelos utilizadores, e por fim o @eventos, que identifica os eventos do texto dos clientes. Estas entities têm como exemplos todas as cidades e nome dos eventos presentes na base de dados do salesforce, que estão na tabela/objeto "Evento__c".

Figura 15- Entity @Meses

5.1.2. Ações e Respostas

No IBM Assistant foram criados nós, e em cada nó definiu-se as condições que permitem determinar que intents ou entities devem ser mencionadas pelos utilizadores para acionar uma determinada ação ou resposta. Todos as condições definidas para cada forma de pesquisa encontram-se no anexo A6, na página 69.

Definiu-se da seguinte forma a condição para o utilizador acionar a pesquisa por mês. O utilizador menciona uns dos exemplos da intent "#PedirEventoMes" e também da entity "@Meses" e que têm de ser diferentes da entity "@Cidades". Caso a condição se confirme, o sistema da IBM Assistant extrai o exemplo da entity "@Meses" mencionada pelo utilizador para produzir uma resposta mais personalizada para o mesmo. Na figura 16 mostra a condição acima referida.

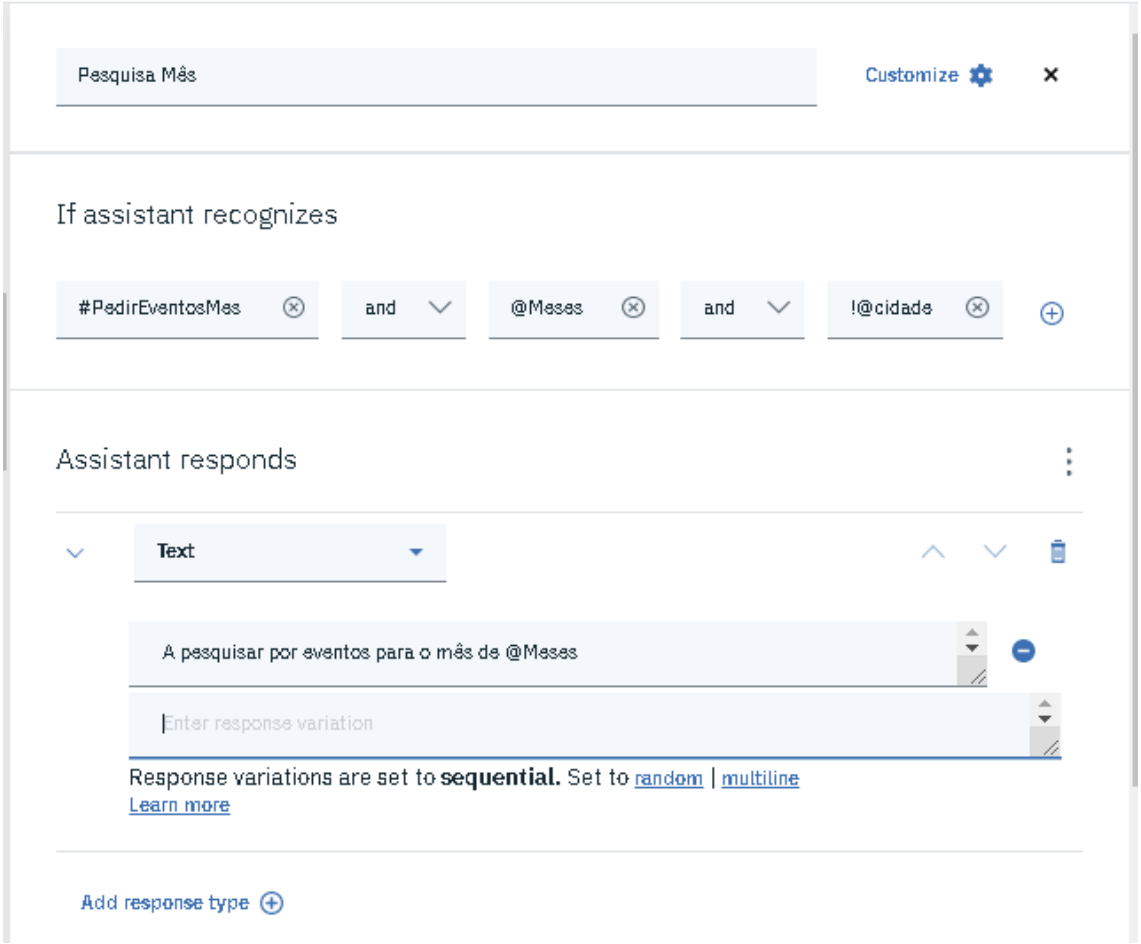

Figura 16- Condição para a pesquisa dos eventos por mês I

Na figura 17 pode-se ver a área de testes da IBM Assistant. Mostra o utilizador a acionar a condição que foi anteriormente mencionada. A entrada (“Mostra me eventos para o mês de novembro”) é identificada como pertencente a intent “#PedirEventoMes “e também são extraídos dados como “novembro”, que pertencem a intity “@Meses”. A entity “@sys-date” também extrai dados importante como a data de início e fim do mês de novembro.

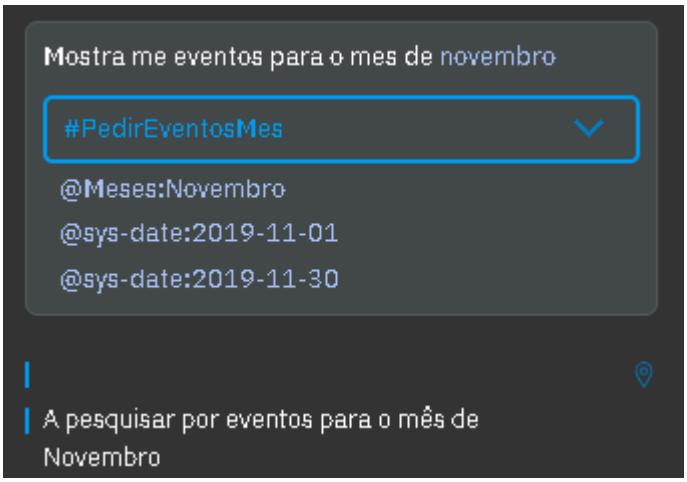

Figura 17- Condição para a pesquisa dos eventos por mês II

5.2. **Visualforce Pages**

Desenvolveu-se a interface do chat utilizando as marcações do Visualforce. Esta página de Visualforce foi associada a um controlador (“SECONTROLLER”) desenvolvido para a mesma. Na figura 18 pode-se ver a interface do chatbot contruído com o Visualforce.

Foi definida uma caixa de texto com o objetivo do utilizador digitar as suas intenções. No lado direito criou-se um botão (“Enviar”) para o utilizador submeter os seus pedidos. Esse Botão quando clicado, atribui o conteúdo da caixa de texto a variável (“PedirEve”) e aciona o método do “SECONTROLLER” designado de “eventos_secontroller”, como mostra o parte do código abaixo.

```
<apex:inputTextarea value="{!PedirEve}"cols="27" rows="2"  html-
placeholder="{!PlaceHolder}",

onkeydown="if(event.keyCode==13){this.blur();call_eventos_meth();}"  />
```

```
<apex:actionFunction name="call_eventos_meth"
action="{!Eventos_SECONTROLLER}"/>

<apex:commandButton action="{!Eventos_SECONTROLLER}"
value="{!TituloBotaoenviar}" />
```

Pode-se verificar que a caixa de texto é representada pela componente "inputTextarea" e o botão pela "commandButton", ambas componentes do Visualforce. O método "eventos_secontroller" é chamado de duas formas, a primeira é quando o botão "Enviar" é pressionado e a segunda verifica-se quando a tecla "Enter" do teclado é pressionada. O atributo "onkeydown" da componente "inputTextarea" permite verificar se a tecla "Enter" foi pressionada ou não, se sim a função "call_eventos_meth" é acionada. Esta função está associada ao método "eventos_secontroller" do controlador.

O código completo da interface do chatbot pode ser consultado no anexo A2, na página 41.

Figura 18- Interface do chatbot

Quando o botão "EN" e o "PT", são pressionados a interface muda para inglês ou português, respetivamente. No código abaixo pode-se verificar que a componente do Visualforce ("commandButton") representa dois botões mencionados e quando são pressionados acionam as ações "!mudarIdiomapt" e "!mudarIdiomaen" do controlador "SECONTROLLER", o código completo desse controlador encontra-se em anexo A3, na página 46.

```
<apex:commandButton value="PT" action="{!mudarIdiomapt}"  "/>
 <apex:commandButton value="EN" action="{!mudarIdiomaen}" />
```

5.3. Classes Apex – Controladores

Um controlador do Visualforce é um conjunto de instruções que especificam o que acontece quando um utilizador interage com os componentes especificados na marcação do Visualforce, como quando um utilizador clica um determinado botão.

Como já foi mencionado a página de Visualforce em que a interface do chatbot foi construído está associada ao controlador "SECONTROLLER". Esse controlador tem a função de enviar o que o utilizador escreve para o chatbot criado na IBM Assistant, receber a resposta, e com a resposta encontrar os eventos disponíveis.

Como já foi mencionado, quando o botão "Enviar" é pressionado, o conteúdo da caixa de texto é atribuído a variável ("PedirEve") e também aciona o método do "SECONTROLLER" designado de "eventos_secontroller". Esse método é responsável por enviar o valor atribuído a variável "PedirEve" para o chatbot criado na IBM Assistant, como é mostrado no código abaixo.

```
1. PedirEve_bk=PedirEve; //apagar o input do cliente
2. PedirEve='';
3. IBMAssistantV1 assistant = new IBMAssistantV1('2018-02-16');
4.   IBMAssistantV1Models.MessageInput input = new
5.IBMAssistantV1Models.MessageInputBuilder().text(Pedi6.rEve_bk).build()
;
7.IBMAssistantV1Models.MessageOptions options = new
8.IBMAssistantV1Models.MessageOptionsBuilder().workspa9.ceId(workspace_i
d).input(input).build();
10.IBMAssistantV1Models.MessageResponse response =
11.assistant.message(options);
12.reply = response.getOutput().getText().get(0);
```

Com o IBM Salesforce SDK, já mencionado atrás, foi possível importar para a plataforma salesforce uma biblioteca com um conjunto de classes que permitem interagir com o chatbot, criado no IBM Assistant, através da linha de código do Apex.

No código acima na linha 3 uma instância do IBM Assistant é criada, no ponto 4 é indicado o texto escrito pelo utilizador que vai ser enviado, o objeto "input" é criado com um construtor onde é especificado a variável "PedirEve_bk" como parâmetro. Variável essa que contém o que o utilizador escreveu. Na linha 8 é especificado o workspaceId do chatbot criado na IBM Assistant. Na linha 10 é feito um pedido ao chatbot criado na IBM Assistant e retorna um objeto no formato JSON (JavaScript Object Notation). O objeto mencionado tem informações muito importantes, como as intents e as intities identificadas pelo chatbot no texto do utilizador, bem com as respostas. Na linha 12, a resposta é extraída do objeto para a variável "reply". Como já referido, na página 27, para cada forma de pesquisa pelos eventos por parte do utilizador, definiu-se uma resposta que é dada pelo chatbot criado na IBM Assistant. Essas respostas são utilizadas para realizar as diferentes procuras pelos eventos pedidos pelos utilizadores na base de dados do salesforce. O código abaixo mostra como é feita a procura pelos eventos, caso o utilizador faça um pedido para pesquisar eventos para uma determinada cidade. A versão completa do código, entra-se no anexo A3, na página 46.

```
.

resp3='A pesquisar por eventos para a cidade de';
.
.
 if(reply.contains(resp3)){
    String local_ev;
    local_ev= response.getEntities().get(0).getValue();
        PedirEventosCidade(local_ev);
 }
.
.
```

No código acima, verifica-se que se a variável "reply" possuir os caracteres da variável "resp3" a condição é verdadeira, isto é, o chatbot criado na IBM Assistant analisa o texto do utilizador e identifica a que intent e entities pertence, e dependendo das condições definidas é dada uma resposta. Todos esses dados, como já mencionado são enviados no formato JSON. Avariável "reply" toma o valor da resposta dada, a

variável “local_ev” toma o valor da entity identificada, que nesse caso será a cidade pedida pelo utilizador. A função “PedirEventosCidade”, procura no objeto “Evento__c”, que contem todos os registos, os eventos com as locais iguais a variável “local_ev”. Todos os eventos encontrados, são colocados numa lista e mostrados para o utilizador na seção com nome de “Eventos”, como pode-se ver na figura 18 da interface do chatbot. Na seção “Eventos”, o utilizador quando clica no nome do evento é redirecionado para a página do evento em específico.

Os exemplos das entities “@cidade” e “@eventos”, são atualizados quando é enviado um primeiro pedido do salesforce para o chatbot criado na IBM Assistant. Quando são adicionados novos eventos na base de dados salesforce é importante verificar se novas cidades foram adicionadas de forma a atualizar os exemplos das entities “@cidade” e “@eventos”. Para isso, criou-se a classe “UpdateEntities”, que tem dois métodos, o “updateEnt_evento” e o “updateEnt_cidade”. O “updateEnt_evento” atualiza a entity “@evento”. Na tabela/objeto “Evento__c” pode haver eventos com o campo cidade com o mesmo valor, por isso criou-se o “updateEnt_cidade”, que permite fazer essa verificação e criar uma lista de cidade dos eventos, mas sem valores repetidos. Essa lista é utilizada para atualizar os exemplos na entity “@cidade” no IBM Assistant. O código completo da classe “UpdateEntities” encontra-se no anexo A4, na página 57.

6. Verificações e validações

Ao longo do projeto foram realizados testes, que permitiram a correção de erros que foram surgindo e também possibilitaram um melhor funcionamento das funcionalidades implementadas.

Como referido anteriormente este projeto, que consistiu em criar um chatbot para uma página de salesforce, foi adicionado a página de eventos criada por outros estagiários.

Na figura 19 encontra-se o chatbot na página mencionada anteriormente.

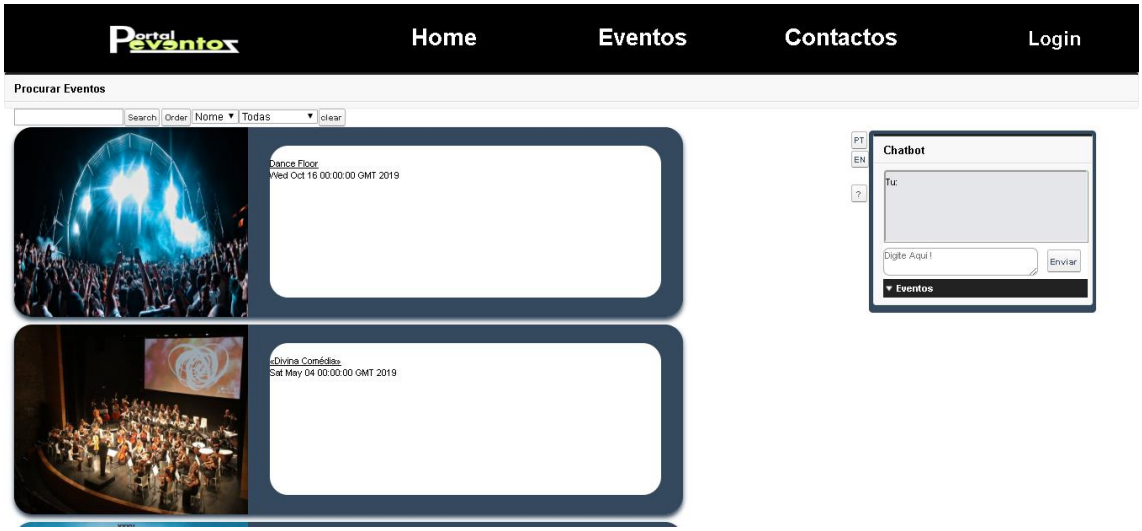

Figura 19- Chatbot na página dos eventos

Na figura 20 encontra-se o chatbot com a interface em inglês.

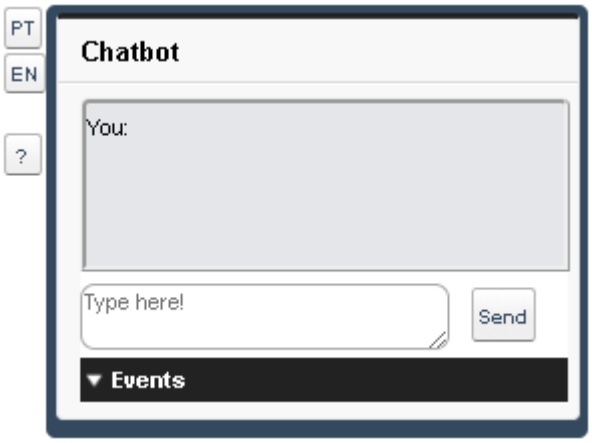

Figura 20- Chatbot com Interface em inglês

Na figura 21 pode-se ver a resposta do chatbot para entrada ‘ola’.

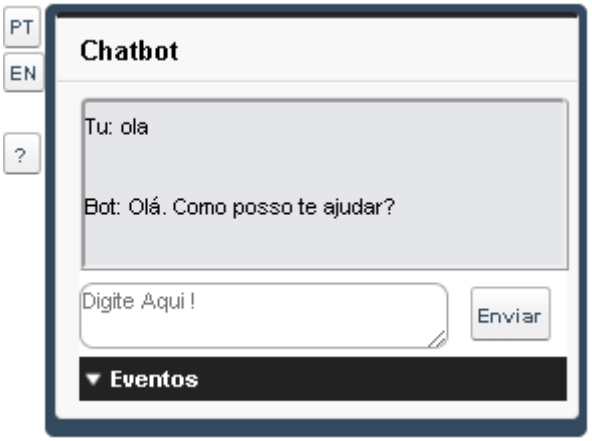

Figura 21- Resposta a entrada "ola"

A figura 22 tem o objetivo de mostrar quando o utilizador solicita todos eventos disponíveis para o chatbot.

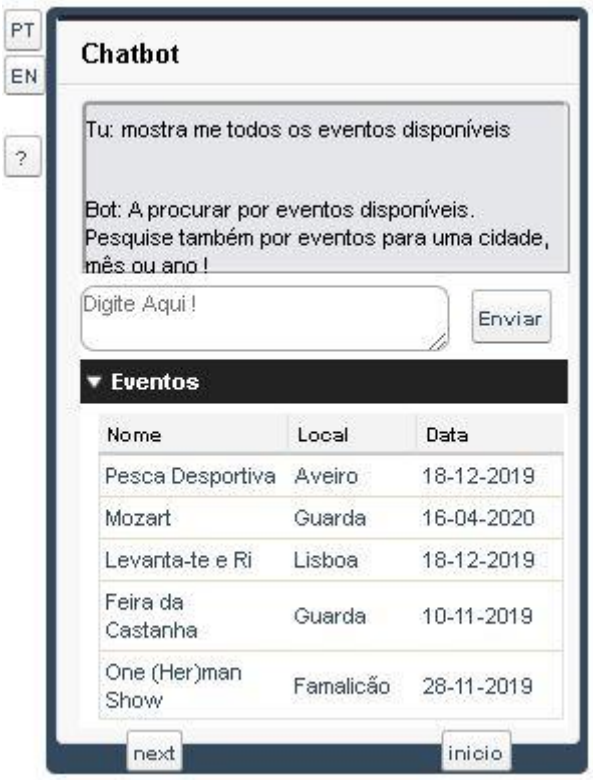

Figura 22- Chatbot: Todos os eventos disponíveis

A figura 23 tem o objetivo de mostrar quando o utilizador solicita eventos para uma determinada cidade e mês.

Figura 23- Chatbot: eventos para um cidade e mês

A figura 24 tem o objetivo de mostrar quando o utilizador pede os detalhes de um determinado evento para o chatbot.

Figura 24- Chatbot: detalhes de um evento

A figura 25 mostra a resposta do chatbot quando o utilizador introduz uma entrada que não faz correspondência a nenhuma das condições definidas para as formas de pesquisa.

Figura 25- Chatbot: Entrada inválida do utilizador

7. Conclusões

Inicialmente existiram algumas dificuldades no desenvolvimento do projeto, dado que foi um primeiro contacto com as tecnologias utilizadas neste projeto. Mas o estudo realizado tanto da plataforma Salesforce como também das ferramentas de criação de chatbot, permitiram facilitar o desenvolvimento do projeto.

O estágio curricular na LOBA, permitiu ganhar mais experiência e também adquirir mais conhecimentos não só de novas tecnologias, mas também dos métodos de trabalho de uma empresa.

Concluiu-se o projeto com sucesso e os objetivos inicialmente definidos, pela empresa, foram alcançados. A duração do estágio foi pequena para o desenvolvimento de um chatbot completo. É de salientar que ainda há trabalho que pode ser feito para a melhoria do projeto desenvolvido. Esse trabalho passaria pela implementação de mais funcionalidades como por exemplo, adicionar mais diálogos ao chatbot, o que permitiria ao utilizador realizar conversas mais dinâmicas. Outra melhoria passaria por aumentar as formas e as variações de pesquisa utilizadas pelo utilizador para procurar pelos eventos, dado que nem todos escrevem da mesma maneira e nem utilizam as mesmas palavras para escrever. As funcionalidades do chatbot para a língua inglesa também necessitam de melhorias.

A experiência adquirida durante o estágio, foi muito gratificante pois permitiu obter uma visão mais ampla do que é trabalhar numa empresa. Também possibilitou a consolidação dos conhecimentos obtidos ao longo do curso.

Referências Bibliográficas

Bibliografia

[1] Dom Digital, “Quem somos,” [Online]. Available: https://www.domdigital.pt/sobrenos/quemsomos.asp. [Acedido em 18 Setembro 2019].

[2] Salesforce, “Chatbot: O que é e como funciona?,” [Online]. Available: https://www.salesforce.com/br/atendimento-ao-cliente/chatbot/. [Acedido em 17 Julho 2019].

[3] Dialogflow, “Agents,” [Online]. Available: https://cloud.google.com/dialogflow/docs/agents-overview. [Acedido em 10 Setembro 2019].

[4] Dialogflow, “Intents,” [Online]. Available: https://cloud.google.com/dialogflow/docs/intents-overview. [Acedido em 10 Setembro 2019].

[5] Dialogflow, “Intities,” [Online]. Available: https://cloud.google.com/dialogflow/docs/entities-overview. [Acedido em 10 Setembro 2019].

[6] Dialogflow, “Responses,” [Online]. Available: https://cloud.google.com/dialogflow/docs/intents-responses. [Acedido em 10 Setembro 2019].

[7] Google, “Documentação do Dialogflow,” [Online]. Available: https://cloud.google.com/dialogflow/docs/. [Acedido em 29 Julho 2019].

[8] IBM, “watson assistant,” [Online]. Available: https://www.ibm.com/cloud/watson-assistant/.

[9] Salesforce, “What is Salesforce,” [Online]. Available: https://www.salesforce.com/products/what-is-salesforce/?d=70130000000i7zF. [Acedido em 31 Outubro 2019].

[10] Trailhead, “Get Started with the Salesforce Platform,” [Online]. Available: https://trailhead.salesforce.com/en/content/learn/modules/starting_force_com/starting_intro. [Acedido em 2 Setembro 2019].

[11] Salesforce, “What is Visualforce?,” [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_intro_what_is_it.htm. [Acedido em 2 Setembro 2019].

[12] Salesforce, “What is Apex?,” [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm#!. [Acedido em 2 Setembro 2019].

[13] Bryan, “NLP vs. NLU: What’s the Difference?,” 5 Outubro 2016. [Online]. Available: https://medium.com/@lola.com/nlp-vs-nlu-whats-the-difference-d91c06780992. [Acedido em 30 Setembro 2019].

[14] IBM, “Getting started with Watson Assistant,” [Online]. Available: https://cloud.ibm.com/docs/services/assistant?topic=assistant-getting-started#getting-started-tutorial. [Acedido em 8 Agosto 2019].

[15] L. Patino, “Embedding Watson Conversation Service into your Salesforce apps has never been this easy,” 6 Novembro 2017. [Online]. Available: https://developer.ibm.com/dwblog/2017/watson-conversation-apex-sdk-salesforce/. [Acedido em 12 Agosto 2019].

[16] L. Gonçalves, “METODOLOGIA AGILE, TUDO O QUE PRECISA DE SABER SOBRE ESTE TEMA,” [Online]. Available: https://luis-goncalves.com/pt-pt/o-que-e-metodologia-agile/. [Acedido em 2 Setembro 2019].

[17] [Online]. Available: https://trailhead.salesforce.com/pt-BR/.

[18] [Online]. Available: https://developer.salesforce.com/.

Anexos

Anexo A1

Tabela/Objeto “Evento__c”

Nome do campo	Data type(Salesforce)
atividade__c	Text Area(255)
breve_descricao__c	Text(200)
Cidade__c	Text(50)
Cidade_Normalizada__c	Text(50)
CoOrganizador__c	Text(80)
CreatedById	Lookup(User)
Data__c	Date
Descricao__c	Text Area(255)
Duracao__c	Time
Estado__c	Picklist
Name	Text(80)
EventoEstado__c	Formula (Text)
geoloc__c	Geolocation
Resticao_de_idades__c	Picklist
LastModifiedById	Lookup(User)
Local__c	Text(120)
Localizacao__c	Lookup(Localizacao)
LotacaoMax__c	Number(7, 0)
Nome__c	Text(150) (Unique Case Insensitive)

Tabela 5- Tabela/Objeto "Evento__c"

Anexo A2

Página Visualforce “ProcurarEventosChat.vfp”

```
<apex:page controller="SearchController" extensions="SECONTROLLER">
    <div style="width:100%">
        <c:Header />
    </div>
<!—IGNORAR Pagina de eventos-->
<apex:pageBlock title="Procurar Eventos" id="pb" >
<apex:form >
<div style="width:100%;">
<div style="width:75%;float:left;">
      <apex:actionFunction action="{!search}" name="actionFunction" />
      <apex:inputText value="{!strKey}"
onkeydown="if(event.keyCode==13){this.blur();actionFunction();}" />
      <apex:commandButton action="{!search}" value="Search" />
      <apex:commandButton action="{!changeOrder}" value="Order" />

      <apex:selectList value="{!strparam}" size="1" required="true"  >
          <apex:selectOptions value="{!items}"/>
            <apex:actionSupport event="onchange" action="{!search}">
      </apex:actionSupport>
      </apex:selectList>

      <apex:selectList value="{!strcat}" size="1" required="true"  >
      <apex:selectOptions value="{!categorias}"/>
      <apex:actionSupport event="onchange" action="{!search}"/>
      </apex:selectList>
      <apex:commandButton action="{!clear}" value="clear" />

      <apex:repeat value="{!eventos}" var="ev" >
            <c:EventosComponent url="{!ev.URLImagem__c}"
            idev="{!ev.id}" date="{!ev.Data__c}" name="{!ev.name}"/>
      </apex:repeat>

      <apex:pageBlockSection columns="4" >
      <apex:pageBlockTable value="{!eventos}" var="ev" >
            <apex:column headerValue="Nome">
                  apex:outputLink value="DetalhesEvento?id={!ev.id}"
                  id="theValue" >{!ev.name}
            </apex:outputLink><br/>
            </apex:column>
            <apex:column value="{!ev.Preco__c}"/>
            <apex:column value="{!ev.Data__c}"/>
            </apex:pageBlockTable>
      </apex:pageBlockSection>

      <apex:panelGrid columns="7">
      <apex:commandButton status="fetchStatus" reRender="pb"
value="|<" action="{!first}" disabled="{!!hasPrevious}" title="First
Page"/>
      <apex:commandButton status="fetchStatus" reRender="pb" value="<"
action="{!previous}" disabled="{!!hasPrevious}" title="Previous
Page"/>
      <apex:commandButton status="fetchStatus" reRender="pb" value=">"
action="{!next}" disabled="{!!hasNext}" title="Next Page"/>
      <apex:commandButton status="fetchStatus" reRender="pb"
value=">|" action="{!last}" disabled="{!!hasNext}" title="Last Page"/>
```

```
<apex:outputText >{!(pageNumber * size)+1-size}-{!IF((pageNumber *
size)>noOfRecords, noOfRecords,(pageNumber * size))} of {!noOfRecords}
</apex:outputText>
<apex:commandButton status="fetchStatus" reRender="pb" value="Refresh"
action="{!refresh}" title="Refresh Page"/>

<apex:outputPanel style="color:#4AA02C;font-weight:bold">
      <apex:actionStatus id="fetchStatus" startText="Fetching..."
stopText=""/>
</apex:outputPanel>
</apex:panelGrid>
</div>
<!-- IGNORAR FIM Pagina de eventos:-->


 <!--Chatbot: Criado para o projeto definido neste documento INICIO-->

<div style="width:25%; float:right;background-color:; margin:30px 0px
0px 0px; ">
<div style="width:325px;" id='todo_chat'>
<div style="height:auto;width:290px; float:right; background-
color:#34495E;border-radius: 5px 5px 5px 5px;padding:5px 5px 0px
5px;">

<apex:pageBlock title="Chatbot">
<div style="background-color:white;height:auto;width:270px; border-
radius: 5px 5px 0px 0px;">

<div style="background-color:rgba(1, 12, 50, 0.1);
height:90px;width:265px;border-radius: 5px 5px 0px 0px; border-
style:ridge; ">

<div style="background-color:; height:40px;border-radius: 5px 5px 0px
0px;padding-top:5px;">
<apex:outputText value="{!TuandYou}: {!PedirEve_bk}" style="">
</apex:outputText>
</div>

<div style="background-color:; height:40px;border-radius: 0px 0px 5px
5px; padding-top:5px;">
<apex:outputText value="{!reply}"  >
</apex:outputText>
</div>
</div>

<div style="background-color:; height:48px;width:270px; border-radius:
0px 0px 0px 0px;">


<div style="float:left; width:65%;padding:6px 0px 0px 0px;">  <!--
Caixa de texto -->

<apex:inputTextarea value="{!PedirEve}" cols="27" rows="2"  html-
placeholder="{!PlaceHolder}" style="border-radius:8px;"
onkeydown="if(event.keyCode==13){this.blur();call_eventos_meth();}"
/>

<apex:actionFunction name="call_eventos_meth"
action="{!Eventos_SECONTROLLER}" />
</div>
```

```
<div style="float:right; width:20%;padding:8px 0px 0px 2px;">
<!--Botão Enviar -->
<apex:commandButton action="{!Eventos_SECONTROLLER}"
value="{!TituloBotaoenviar}"  style="height:30px; color:#34495E; "/>
</div>
</div>

<div style="height:auto;">

<apex:pageBlockSection title="{!TituloSessaoeventos}" columns="1" >
<!--Seção Eventos -->

<apex:repeat value="{!detalheList}" var="det_" >
      <apex:outputText value="{!det_.Local__c}"  rendered="{!
      mostrarDetalhe}" ></apex:outputText>

      <apex:outputText value="{!det_.Data__c}" rendered="{!
      mostrarDetalhe}" ></apex:outputText>

      <apex:outputText value="{!det_.Preco__c}" rendered="{!
      mostrarDetalhe}" ></apex:outputText>

      <apex:outputText value="{!det_.Resticao_de_idades__c}"
      rendered="{! mostrarDetalhe}" ></apex:outputText>

      <apex:outputText value="{!det_.TipoEvento__c}"
      rendered="{!mostrarDetalhe}" ></apex:outputText>

<apex:image url="{!det_.URLImagem__c}" width="250" height="40" alt=""
rendered="{! mostrarDetalhe}"/>

<apex:outputLink value="https://apitest-developer-
edition.eu19.force.com/DetalhesEvento?id={!det_.Id}" target="_blank"
style="text-decoration: true; " rendered="{! mostrarDetalhe}">
<p align="center" style="background-color:rgba(1, 12, 50, 0.1);
width:250px;">{!VerMais}</p>
</apex:outputLink>
</apex:repeat>

<apex:pageBlockTable value="{!Lista}" var="ev"
rendered="{!MostrarEvento}" width="100%">
      <apex:column headerValue="Nome" width="98px" >
      <!— quando o nome do evento é pressionado a um redireccionamento
para a página espicifica do evento->

    <apex:outputLink value="https://apitest-developer-
edition.eu19.force.com/DetalhesEvento?id={!ev.Id}" target="_blank"
style="text-decoration: none;color:#34495E;">{!ev.Name}
      </apex:outputLink>
</apex:column>

<apex:column value="{!ev.Local__c}" style=" color:#34495E;"/>
<apex:column value="{!ev.Data__c}" style=" color:#34495E;"/>

</apex:pageBlockTable>
</apex:pageBlockSection>

<div style="width:280px;">

<div style="float:left; width:80px;padding:0px 0px 0px 25px">
```

```
<apex:commandButton action="{!ProximaPagina}" value="next"
style="height:;  color:#34495E;" rendered="{!MostrarBtnSeguinte}" />
</div>

<div style="float:right;width:80px;">
<apex:commandButton action="{!inicio}" value="inicio"  style="height:;
color:#34495E;" rendered="{!MostrarBtnInicio}"/>
</div>

</div>
</div>

</div>
</apex:pageBlock>
</div>

<div id="idioma" style="width:15px; float:left;">

<div id="idiomaPT">  <!--Botão PT -->

<apex:commandButton value="PT" action="{!mudarIdiomapt}" id="buttonPT"
style=" color:#34495E;"/>

</div>

<div id="idiomaEN"><!--Botão EN -->

 <apex:commandButton value="EN" action="{!mudarIdiomaen}"
id="buttonEN" style=" color:#34495E;"/>
 </div>

 <div id="Ajuda" style="padding:20px 0px 0px 0px">  <!--Botão ? -->

 <apex:commandButton value=" ? " onclick="openWin()" id="buttonAjuda"
style=" color:#34495E;" />

</div>
</div>
</div>
</div>
</div>
</apex:form>
</apex:pageBlock>

<script>
    var text_ = " Este Chatbot permite pesquisar por eventos
    disponíveis na pagina "
    var myWindow;
<!—Janela que abri ao clicar no Botão ? -->
    function openWin() {
      myWindow=window.open("", "", "width=650, height=300");
      myWindow.document.write("<p style='color:#34495E;font-
family:courier;font-size:100%;text-align:center;'>Este Chatbot permite
pesquisar por eventos disponíveis na página Portal Eventos</p>");
      myWindow.document.write("<p>As pesquisas podem ser feitas
utilizando:</p>");
      myWindow.document.write("<ul><li>O dia e mês do eventos
(Exemplo: Pesquisa por eventos para o dia 21 de Setembro)</li> <li>O
mês (Exemplo: Pesquisa por eventos disponiveis para o mês de
Dezembro)</li> <li>Mês e ano (Exemplo: Procura eventos em Abril de
2020)</li> <li>Ano (Exemplo: Procura por eventos para o ano de
```

```
2019)</li> <li>Cidade (Exemplo: Mostra me os eventos disponiveis para
a cidade da Guarda)</li> <li>Cidade e mês (Exemplo: Mostra me os
eventos do mês de Dezembro na cidade Guarda)</li> <li>Cidade,mês e ano
(Exemplo: Procura por eventos em outubro de 2019 na cidade de
braga)</li></ul>");
      myWindow.moveTo(500, 100);
      myWindow.focus();
    }
</script>

<!--Chatbot: Criado para o projeto definido neste documento FIM-->

</apex:page>
```

Anexo A3

Controlador “SECONTROLLER.apxc”

```
public with sharing class SECONTROLLER {

    public SECONTROLLER(SearchController controller){}

    UpdateEntities uptest = new UpdateEntities();
    List<Evento__c> evlist = new List<Evento__c>();
    List<Evento__c> evlist_out = new List<Evento__c>();
    List<Evento__c> detalhelist = new List<Evento__c>();
    List<String> M ;
    List<Evento__c> ev_procurado {get;set;}
    public String data {get;set;}     //toma o valor da data do evento
a pesquisar
    public String PedirEve {get;set;} //toma o valor do input do
utilizador
    public String reply {get;set;}   //toma o valor da resposta do
chatbot
    public String replydata {get;set;} //toma o valor da data contida
na resposta do chatbot
    public String PedirEve_bk {get;set;}
    public String eventoEscolhido {get; set;}
    Integer n_i=0,n_f=5,n;
    Boolean incrementar=true;
    Boolean MostrarBtnSeguinte= false;
    Boolean MostrarBtnInicio= false;
    Boolean ingles= false;
    Boolean portugues = true;
    String workspace_id;
    String TituloBotaoenviar = 'Enviar';
    String PlaceHolder = 'Digite Aqui !';
    String TuandYou = 'Tu';
    String TituloSessaoeventos = 'Eventos';
    String VerMais = 'Ver mais !';

    String EntityName;
    Boolean MostrarDetalhe = false;
    Boolean MostrarEvento = false;
    Boolean carregarEntities =true;


    public void Eventos_SECONTROLLER(){
         MostrarEvento=false;
         MostrarBtnSeguinte=false;
         MostrarBtnInicio=false;
        incrementar=true;
        n_i=0;n_f=5;evlist_out.clear();
        if(portugues){
            M = new List
<String>{'Janeiro','Fevereiro','Março','Abril','Maio','Junho','Julho',
'Agosto','Setembro','Outubro','Novembro','Dezembro'};

             workspace_id='519eb057-c090-4e05-acc8-0dbdd4cd327f';

            TituloSessaoeventos = 'Eventos';
        }else{
```

```
             M = new List
<String>{'Jenuary','February','March','April','May','June','July','Aug
ust','September','October','November','December'};

            workspace_id='908b99d4-6470-4bcb-8221-30af5d1c4fb3';


            TituloSessaoeventos = 'Events';
        }


        if(carregarEntities) {
            //atualizada a entity @eventos
            uptest.updateEnt_evento(workspace_id,'eventos');

            //atualizada a entity @cidade

            uptest.updateEnt_cidade(workspace_id,'cidade');

            carregarEntities=false;
        }

        PedirEve_bk=PedirEve;
        PedirEve='';//apagar o input do cliente o botão enviar é
pressionado


        IBMAssistantV1 assistant = new IBMAssistantV1('2018-02-16');

        IBMAssistantV1Models.MessageInput input = new
IBMAssistantV1Models.MessageInputBuilder()
        .text(PedirEve_bk).build();

        IBMAssistantV1Models.MessageOptions options
        = new IBMAssistantV1Models.MessageOptionsBuilder()
        .workspaceId(workspace_id)
        .input(input)
        .build();

        IBMAssistantV1Models.MessageResponse response =
assistant.message(options);

        reply = response.getOutput().getText().get(0);
        String resp1,resp2,resp3,resp4,resp5,resp6,resp7,resp8,resp9;

        if(portugues){
            resp1='A pesquisar por eventos para o dia'; // pesquisa
dia
            resp2='A procurar por eventos disponíveis. Pesquise também
por eventos para uma cidade, mês ou ano !'; //pesquisa todos os
eventos disponíveis
            resp3='A pesquisar por eventos para a cidade de';//
pesquisa cidade
            resp4='A pesquisar por eventos para o mês de';// pesquisa
mes
            resp5='A pesquisar por eventos para o ano';// pesquisa ano
            resp6='A pesquisar mais informações sobre o evento'; //
pesquisa detalhes evento
            resp7='A olhar por eventos na cidade de';//pesquisa cidade
mes
            resp8='Procurando eventos em';// pesquisa cidade ano
```

```
            resp9='pesquisa de eventos cidade mês ano';  // pesquisa
cidade mes ano

        }else{
            resp1='Looking for available  events for'; // pesquisa dia
            resp2='Searching for available  events'; //pesquisa todos
os eventos disponíveis
            resp3='Searching events for the';// pesquisa cidade
            resp4='Looking for events in';// pesquisa mes
            resp5='Searching for events for the year';// pesquisa ano
            resp6='Looking for more details about';// pesquisa
detalhes evento
            resp7='available events are being searched for';//pesquisa
cidade mes
            resp8='Looking events in';// pesquisa cidade ano
            resp9='events city month year'; // pesquisa cidade mes ano
        }

        if(!reply.contains(resp1)){

            MostrarDetalhe=false;
            reply='Bot:  '+reply;
            evlist.clear();//limpa a lista dos eventos mostrados

            if(reply.contains(resp2)){
                PedirEventosDesponiveis();
            }

            if(reply.contains(resp3)){
                String local_ev;

                local_ev= response.getEntities().get(0).getValue();
                    PedirEventosCidade(local_ev);
            }

            if(reply.contains(resp7)){
                  String data1,data2,Mes,local_ev;

if(M.contains(response.getEntities().get(1).getValue())){
                    local_ev=response.getEntities().get(0).getValue();
                    Mes=response.getEntities().get(1).getValue();
                    data1=response.getEntities().get(2).getValue();
                    data2=response.getEntities().get(3).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
                }else
if(M.contains(response.getEntities().get(3).getValue())){
                    local_ev=
response.getEntities().get(0).getValue();
                    Mes=response.getEntities().get(3).getValue();
                    data1=response.getEntities().get(1).getValue();
                    data2=response.getEntities().get(2).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
                }else
if(M.contains(response.getEntities().get(0).getValue())){
                    local_ev=
response.getEntities().get(3).getValue();
                    Mes=response.getEntities().get(0).getValue();
                    data1=response.getEntities().get(1).getValue();
                    data2=response.getEntities().get(2).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
```

```
                }else
if(M.contains(response.getEntities().get(2).getValue())){
                    local_ev=
response.getEntities().get(3).getValue();
                    Mes=response.getEntities().get(2).getValue();
                    data1=response.getEntities().get(0).getValue();
                    data2=response.getEntities().get(1).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
                }
            }

            if(reply.contains(resp8)){
                system.debug('dentro resp8');
                 String data1,data2,Ano,local_ev;

if(uptest.lista_cidades().contains(response.getEntities().get(0).getVa
lue())){
                     system.debug('dentro resp8 contains1');
                    local_ev=response.getEntities().get(0).getValue();
                    Ano=response.getEntities().get(3).getValue();
                    data1=response.getEntities().get(1).getValue();
                    data2=response.getEntities().get(2).getValue();
                    PedirEventosCidadeAno(local_ev,data1,data2,Ano);
                }else
if(uptest.lista_cidades().contains(response.getEntities().get(3).getVa
lue())){
                    local_ev=
response.getEntities().get(3).getValue();
                    Ano=response.getEntities().get(2).getValue();
                    data1=response.getEntities().get(0).getValue();
                    data2=response.getEntities().get(1).getValue();
                    PedirEventosCidadeAno(local_ev,data1,data2,Ano);
                }

            }

            if(reply.contains(resp9)){
                 system.debug('passei aqui ;resp9;');
                  String data1,data2,Mes,local_ev;

if(M.contains(response.getEntities().get(1).getValue())){
                    system.debug('passei aqui ;m.contains;');
                    local_ev=response.getEntities().get(0).getValue();
                    Mes=response.getEntities().get(1).getValue();
                    data1=response.getEntities().get(2).getValue();
                    data2=response.getEntities().get(3).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
                }else
if(M.contains(response.getEntities().get(3).getValue())){
                    local_ev=
response.getEntities().get(0).getValue();
                    Mes=response.getEntities().get(3).getValue();
                    data1=response.getEntities().get(1).getValue();
                    data2=response.getEntities().get(2).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
                }else
if(M.contains(response.getEntities().get(2).getValue())){
                    local_ev=
response.getEntities().get(4).getValue();
                    Mes=response.getEntities().get(2).getValue();
                    data1=response.getEntities().get(0).getValue();
```

```
                    data2=response.getEntities().get(1).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
                }else
if(M.contains(response.getEntities().get(0).getValue())){
                    local_ev=
response.getEntities().get(4).getValue();
                    Mes=response.getEntities().get(0).getValue();
                    data1=response.getEntities().get(1).getValue();
                    data2=response.getEntities().get(2).getValue();
                    PedirEventosCidadeMes(local_ev,data1,data2,Mes);
                }
            }

            if(reply.contains(resp4)){

                 String data1,data2,Mes;

if(M.contains(response.getEntities().get(0).getValue())){
                    Mes=response.getEntities().get(0).getValue();
                    data1=response.getEntities().get(1).getValue();
                    data2=response.getEntities().get(2).getValue();
                }else{
                    Mes=response.getEntities().get(2).getValue();
                    data1=response.getEntities().get(0).getValue();
                    data2=response.getEntities().get(1).getValue();
                }

                 PedirEventoMes(data1,data2,Mes);
            }

            if(reply.contains(resp6)){
                MostrarDetalhe=true;//mostra os detalhes
                MostrarEvento=false;
                MostrarBtnSeguinte=false;
                eventoEscolhido =
response.getEntities().get(0).getValue();

               List<Evento__c> list_eventos = [SELECT Name, Local__c,
Data__c,EventoEstado__c,Preco__c,Duracao__c,Resticao_de_idades__c,Tipo
Evento__c,URLImagem__c
                                                FROM Evento__c where
EventoEstado__c !='DECORRIDO' and
                                                EventoEstado__c
!='CONCLUIDO' and EventoEstado__c !='CANCELADO' and Name=
:eventoEscolhido];

                TituloSessaoeventos=eventoEscolhido;
                detalhelist=list_eventos;

            }

            if(reply.contains(resp5)){
                 String data1,data2,ano;
                data1=response.getEntities().get(0).getValue();
                 data2=response.getEntities().get(1).getValue();
                ano=response.getEntities().get(2).getValue();
                 Date date1= Date.valueOf(data1);
                 Date date2= Date.valueOf(data2);
                 List<Evento__c> list_eventos = [SELECT Name,
Local__c,  Data__c,EventoEstado__c,Preco__c
```

```
                                                FROM Evento__c where
EventoEstado__c !='DECORRIDO' and
                                                EventoEstado__c
!='CONCLUIDO' and EventoEstado__c !='CANCELADO' and Data__c > :date1
and Data__c < :date2 ];

                evlist=list_eventos;

                if(evlist.isEmpty()){
                    MostrarEvento=false;
                    if(portugues){
                        reply='Bot: Não existem eventos desponiveis
para o ano '+ano;
                    }else{
                        reply='Bot: There are no events for '+ano;
                    }
                }else{ MostrarEvento=true;
                     }

            }


        }else{

            MostrarDetalhe=false;
            replydata = response.getEntities().get(1).getValue();
            PedirEventoData(replydata);

        }
    }


    public void mudarIdiomapt(){
        if(ingles){
            MostrarBtnInicio=false;
            MostrarBtnSeguinte=false;
             MostrarEvento=false;
            ingles=false;
            portugues=true;
           TituloBotaoenviar='Enviar';
            PlaceHolder='Digite Aqui !';
            TuandYou='Tu';
            TituloSessaoeventos='Eventos';
            reply='';//limpa a conversa quando o idioma é alterado
            PedirEve_bk='';
            PedirEve='';
            evlist.clear(); //limpa o eventos mostrados na seção
Eventos
            detalhelist.clear();
            carregarEntities=true;
            VerMais = 'Ver mais !';

        }else{
            portugues=true;
        }

    }

    public void PedirEventosDesponiveis() {
        MostrarEvento=true;
```

```
        List<Evento__c> list_eventos = [SELECT Name, Local__c,
Data__c,EventoEstado__c,Preco__c FROM Evento__c where EventoEstado__c
!='DECORRIDO' and EventoEstado__c !='CONCLUIDO' and EventoEstado__c
!='CANCELADO'];
        evlist = list_eventos;
    }

    public void PedirEventosCidade(String local_ev){


                if(local_ev=='Lisbon'){
                    local_ev='Lisboa';
                }

                List<Evento__c> list_eventos = [SELECT Name, Local__c,
Data__c,EventoEstado__c,Preco__c
                                                FROM Evento__c where
EventoEstado__c !='DECORRIDO' and
                                                EventoEstado__c
!='CONCLUIDO' and EventoEstado__c !='CANCELADO'];
                local_ev.toLowerCase();
                for(Evento__c evs: list_eventos){

                    if(evs.Local__c.toLowerCase()==local_ev){

                         evlist.add(evs);
                    }

                }

                if(evlist.isEmpty()){
                    //reply='';
                    MostrarEvento=false;
                    if(portugues){
                        reply='Bot: Não existem eventos desponiveis
para '+local_ev.toUpperCase();
                    }else{
                        reply='Bot: There are no events for
'+local_ev.toUpperCase();
                    }
                }else{ MostrarEvento=true;
                     }

    }

    public void PedirEventoData(String data){

        evlist.clear();//limpa a lista dos eventos mostrados

        reply='Bot:  '+reply;

        List<Evento__c> list_eventos = [SELECT Name, Local__c,
Data__c,Preco__c FROM Evento__c where EventoEstado__c !='DECORRIDO'
and EventoEstado__c !='CONCLUIDO' and EventoEstado__c !='CANCELADO'];

        data=replydata;

        Date dataVar;
        String data_string;
```

```
        //para cada valor da lista list_ev é feita uma comparação de
datas
        for(Evento__c evs :list_eventos){

            dataVar=evs.Data__c;
            data_string=String.valueOf(dataVar);
            //os registos que tiverem a data com o mesmo valor da data
passado por parametro são adicionados na lista ev_list
            if(data_string==data){
                evlist.add(evs);

            }
        }

        //se a lista de eventos estiver vazia o reply é alterado
            if(evlist.isEmpty()){
                reply='';
                MostrarEvento=false;
                if(portugues){
                    reply='Bot: Não existem eventos desponiveis para a
data '+data;
                }else{
                    reply='Bot: There are no events for '+data;
                }
            }else{MostrarEvento=true;//MostrarBtnInicio=true;
                 }

}

    public void PedirEventoMes(String data1,String data2,String Mes){

        Date date1= Date.valueOf(data1);
        Date date2= Date.valueOf(data2);
        List<Evento__c> list_eventos = [SELECT Name, Local__c,
Data__c,EventoEstado__c,Preco__c
        FROM Evento__c where EventoEstado__c !='DECORRIDO' and
        EventoEstado__c !='CONCLUIDO'and EventoEstado__c !='CANCELADO'
and Data__c > :date1 and Data__c < :date2 ];
        evlist=list_eventos;


        if(evlist.isEmpty()){
        //reply='';
        MostrarEvento=false;
        if(portugues){
            reply='Bot: Não existem eventos desponiveis para o mês de
'+Mes;
        }else{
            reply='Bot: There are no events for '+Mes;
        }
        }else{ MostrarEvento=true;
             }
    }
    public void PedirEventosCidadeMes(String local_ev,String
data1,String data2,String Mes){
        system.debug('passei aqui ;cidadeMes;');
        Date date1= Date.valueOf(data1);
        Date date2= Date.valueOf(data2);
        List<Evento__c> list_eventos = [SELECT Name, Local__c,
Data__c,EventoEstado__c,Preco__c
        FROM Evento__c where EventoEstado__c !='DECORRIDO' and
```

```
        EventoEstado__c !='CONCLUIDO'and EventoEstado__c !='CANCELADO'
and Local__c =:local_ev and Data__c > :date1 and Data__c < :date2 ];
        evlist=list_eventos;

        if(evlist.isEmpty()){

            MostrarEvento=false;
            if(portugues){
                reply='Bot: Não existem eventos para a cidade de
'+local_ev+' para o mês de '+Mes;
            }else{
                reply='Bot: There are no events for '+local_ev+' city
in '+Mes;
            }
        }else{
            if(portugues){
                reply='Bot: A pesquisar eventos para a cidade de
'+local_ev+' para o mês de '+Mes;
            }else{
                reply='Bot: Looking for events for '+local_ev+' city
in '+Mes;
            }
            MostrarEvento=true;
        }
    }

    public void PedirEventosCidadeAno(String local_ev,String
data1,String data2,String Ano){

        Date date1= Date.valueOf(data1);
        Date date2= Date.valueOf(data2);
        List<Evento__c> list_eventos = [SELECT Name, Local__c,
Data__c,EventoEstado__c,Preco__c
        FROM Evento__c where EventoEstado__c !='DECORRIDO' and
        EventoEstado__c !='CONCLUIDO'and EventoEstado__c !='CANCELADO'
and Local__c =:local_ev and Data__c > :date1 and Data__c < :date2 ];
        evlist=list_eventos;



        if(evlist.isEmpty()){

        MostrarEvento=false;
        if(portugues){
            reply='Bot: Não existem eventos para a cidade de
'+local_ev+' em '+Ano;
        }else{
            reply='Bot: There are no events for '+local_ev+' city in
'+Ano;
        }
        }else{

            MostrarEvento=true;
        }
    }
     public void mudarIdiomaen(){
        if(portugues){
            MostrarBtnInicio=false;
            MostrarBtnSeguinte=false;
            MostrarEvento=false;
            portugues=false;
```

```
            ingles=true;
            TituloBotaoenviar='Send';
            PlaceHolder='Type here!';
            TuandYou='You';
            TituloSessaoeventos='Events';
            reply='';//limpa conversa quando o idioma é alterado
            PedirEve_bk='';
            PedirEve='';
            evlist.clear();//limpa os eventos mostrados a seção de
eventos
            detalhelist.clear();
             carregarEntities=true;
            VerMais = 'See more !';
        }else{
            ingles=true;
        }
    }

    public String getTituloBotaoenviar(){
        return TituloBotaoenviar;
    }

    public String getPlaceHolder(){
        return PlaceHolder;
    }

    public String getTuandYou(){
        return TuandYou;
    }

    public String getTituloSessaoeventos(){
        return TituloSessaoeventos;
    }

    public String getVerMais(){
        return VerMais;
    }

   public List<Evento__c> getListaEv(){
       return evlist;
    }

     public List<Evento__c> getDetalheList(){
       return detalhelist;
    }

    public  Boolean getMostrarDetalhe(){
        return MostrarDetalhe;

    }

    public  Boolean getMostrarEvento(){
        return MostrarEvento;

    }
    public Boolean getMostrarBtnSeguinte(){
        return MostrarBtnSeguinte;
    }

    public Boolean getMostrarBtnInicio(){
        return MostrarBtnInicio;
```

```
    }

    public void inicio(){
        n_i=0;n_f=5;
        MostrarBtnSeguinte=true;
        incrementar=true;
        ProximaPagina();

    }
        public void ProximaPagina(){

            evlist_out.clear();
        if(!getListaEv().isEmpty()){
            if(getListaEv().size()<=5){
                MostrarBtnInicio=false;
            }else{MostrarBtnInicio=true;}
            for(Integer i=n_i;i<n_f;i++){
                if(i>=getListaEv().size()){
                    MostrarBtnSeguinte=false;
                    incrementar=false;
                    break;
                }else{
                    evlist_out.add(getListaEv()[i]);
                }
            }
            if(incrementar){
                n_i=n_f;
                n_f=n_i+5;
            }
        }

        }
    public List<Evento__c> getLista(){
        if(!evlist_out.isEmpty()){
            return evlist_out;
        }else{
            MostrarBtnSeguinte=true;
           ProximaPagina();
        }
        return evlist_out;
    }

}
```

Anexo A4

Métodos “updateEnt_evento” e “updateEnt_cidade”

```
public class UpdateEntities {
    public void updateEnt_evento(String workspace_id, String
EntityName){


        List<Evento__c> lista_ev = [SELECT Name,TipoEvento__c,
Local__c,  Data__c,EventoEstado__c
                                                FROM Evento__c where
EventoEstado__c !='DECORRIDO' and
                                                EventoEstado__c
!='CONCLUIDO' ]; // lista lista_ev contem todos os eventos

        IBMAssistantV1 assistant = new IBMAssistantV1('2018-02-16');

        List<IBMAssistantV1Models.CreateValue> lista = new
List<IBMAssistantV1Models.CreateValue>();
        IBMAssistantV1Models.CreateValue createValue ;



        for(Evento__c item: lista_ev){
            createValue = new
IBMAssistantV1Models.CreateValueBuilder().value(item.Name).build();
            lista.add(createValue);
        }


        //update
        IBMAssistantV1Models.UpdateEntityOptions options1 = new
IBMAssistantV1Models.UpdateEntityOptionsBuilder()

.workspaceId(workspace_id).entity(EntityName).newValues(lista).build()
;

        IBMAssistantV1Models.Entity entity =
assistant.UpdateEntity(options1); // executa o update a entity
@eventos

    }


    List<Evento__c> lista_ev = [SELECT Name,TipoEvento__c, Local__c,
Data__c,EventoEstado__c
                                                FROM Evento__c where
EventoEstado__c !='DECORRIDO' and
                                                EventoEstado__c
!='CONCLUIDO' ];

    List<Evento__c> lista_ev_sem_repeticao = new List<Evento__c>();

    public void updateEnt_cidade(String workspace_id, String
EntityName){
        lista_ev_sem_repeticao.add(lista_ev[0]);

    Object local;
```

```
        for(Integer i = 0; i< lista_ev.size();i++){

            local=lista_ev[i].Local__c;
            verificarCidadeDuplicada(local,lista_ev[i]);// para cada
local da lista lista_ev é verificado se já existe na lista
lista_ev_sem_repeticao

        }

        IBMAssistantV1 assistant = new IBMAssistantV1('2018-02-16');

        List<IBMAssistantV1Models.CreateValue> lista = new
List<IBMAssistantV1Models.CreateValue>();
        IBMAssistantV1Models.CreateValue createValue ;



       for(Evento__c item: lista_ev_sem_repeticao){

            createValue = new
IBMAssistantV1Models.CreateValueBuilder().value(item.Local__c).build()
;
            lista.add(createValue);
        }

        createValue = new
IBMAssistantV1Models.CreateValueBuilder().value('Lisbon').build();
        lista.add(createValue);


        //update
        IBMAssistantV1Models.UpdateEntityOptions options1 = new
IBMAssistantV1Models.UpdateEntityOptionsBuilder()

.workspaceId(workspace_id).entity(EntityName).newValues(lista).build()
;

        IBMAssistantV1Models.Entity entity =
assistant.UpdateEntity(options1);
        System.debug(entity);

        //System.debug(lista_ev);
    }


    public void verificarCidadeDuplicada(Object local,Evento__c
evento){
        Integer stop = 0;

        for(Integer j = 0; j<lista_ev_sem_repeticao.size();j++){
            if(!lista_ev_sem_repeticao[j].Local__c.equals(local)){
                    stop=stop+1;
                }
        }

        if(stop==lista_ev_sem_repeticao.size()){
                lista_ev_sem_repeticao.add(evento);
            }

    }
```

```
    public List<String> lista_cidades(){
        List<String> cidades = new List<String>();

        for(Evento__c c:lista_ev_sem_repeticao){
            cidades.add(c.Local__c);
        }
        return cidades;
    }

}
```

Anexo A5

Intents criadas no IBM Watson Assistant

Na figura 25 apresenta os exemplos que os utilizadores podem usar para acionar a intent "welcome".

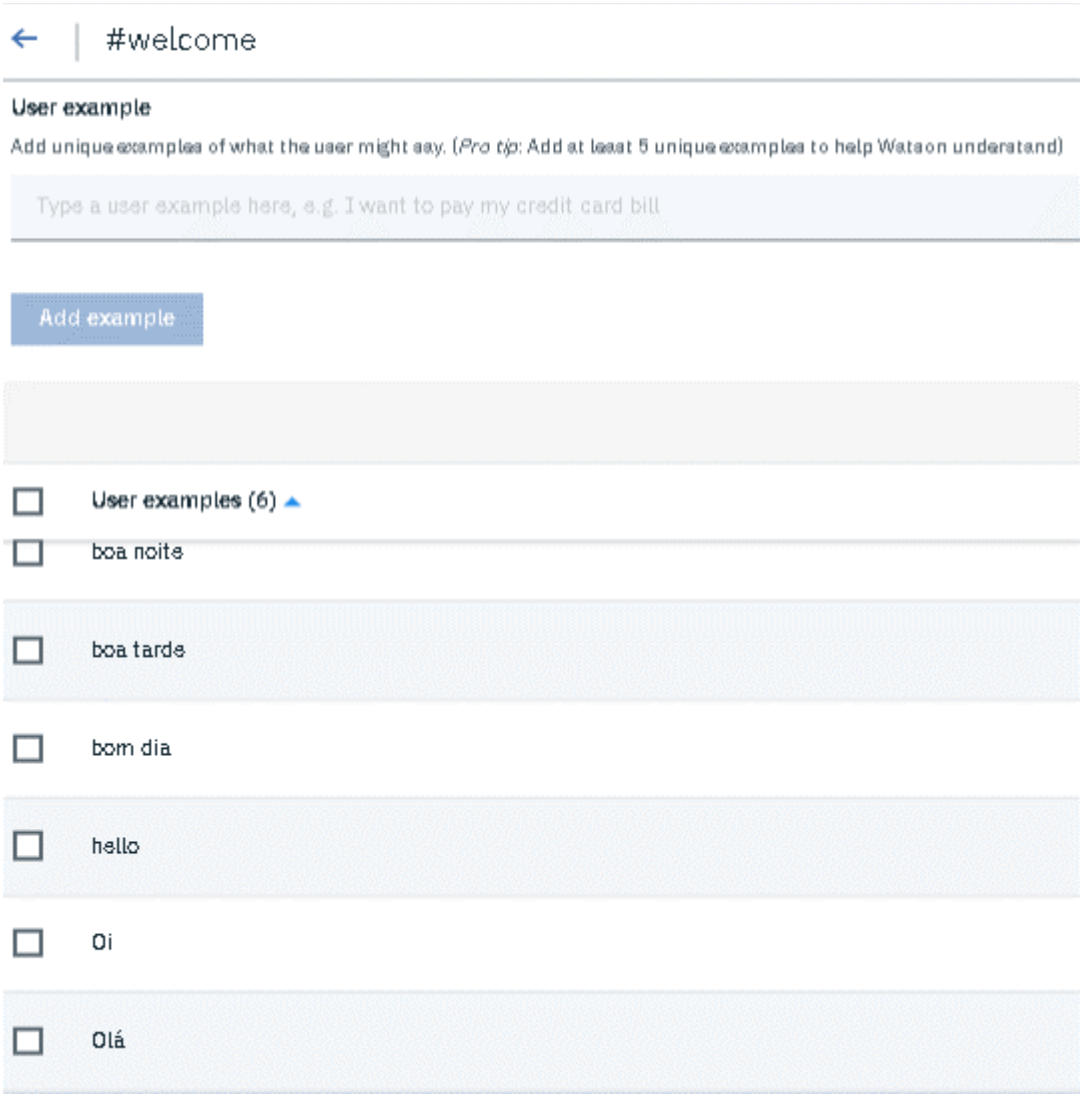

Figura 26- Intent "welcome"

Na figura 26 apresenta os exemplos que os utilizadores podem usar para acionar a intent "PedirEventosPorCidades".

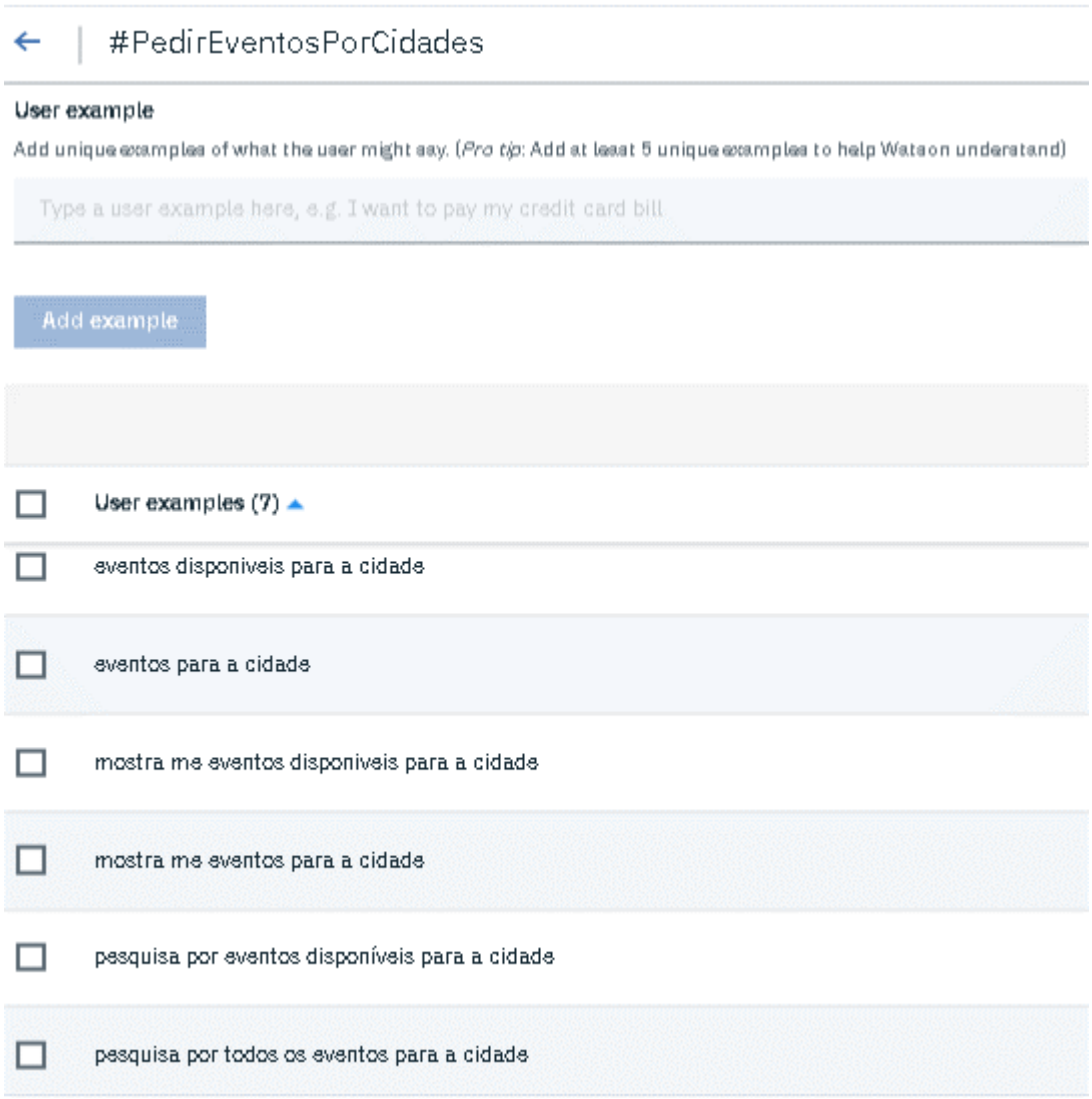

Figura 27- Intent "PedirEventosPorCidades"

Nas figuras 27, 28, e 29 apresentam os exemplos que os utilizadores podem usar para acionar a intent “PedirEventosDisponiveis”.

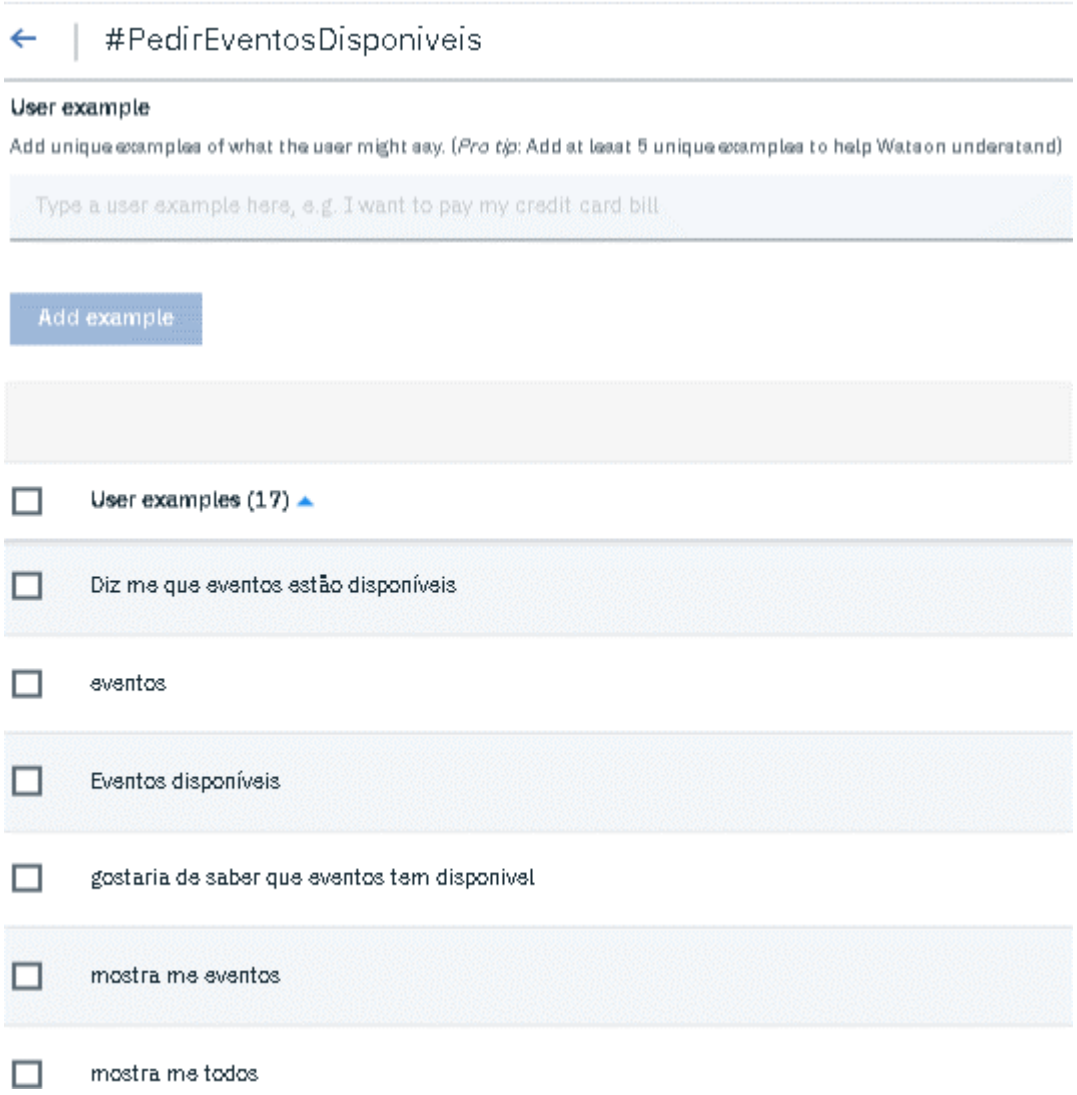

Figura 28- Intent “PedirEventosDisponiveis”

Figura 29- Intent "PedirEventosDisponiveis"

Figura 30- Intent "PedirEventosDisponiveis"

Na figura 30 apresenta os exemplos que os utilizadores podem usar para acionar a intent "PedirEventosAno".

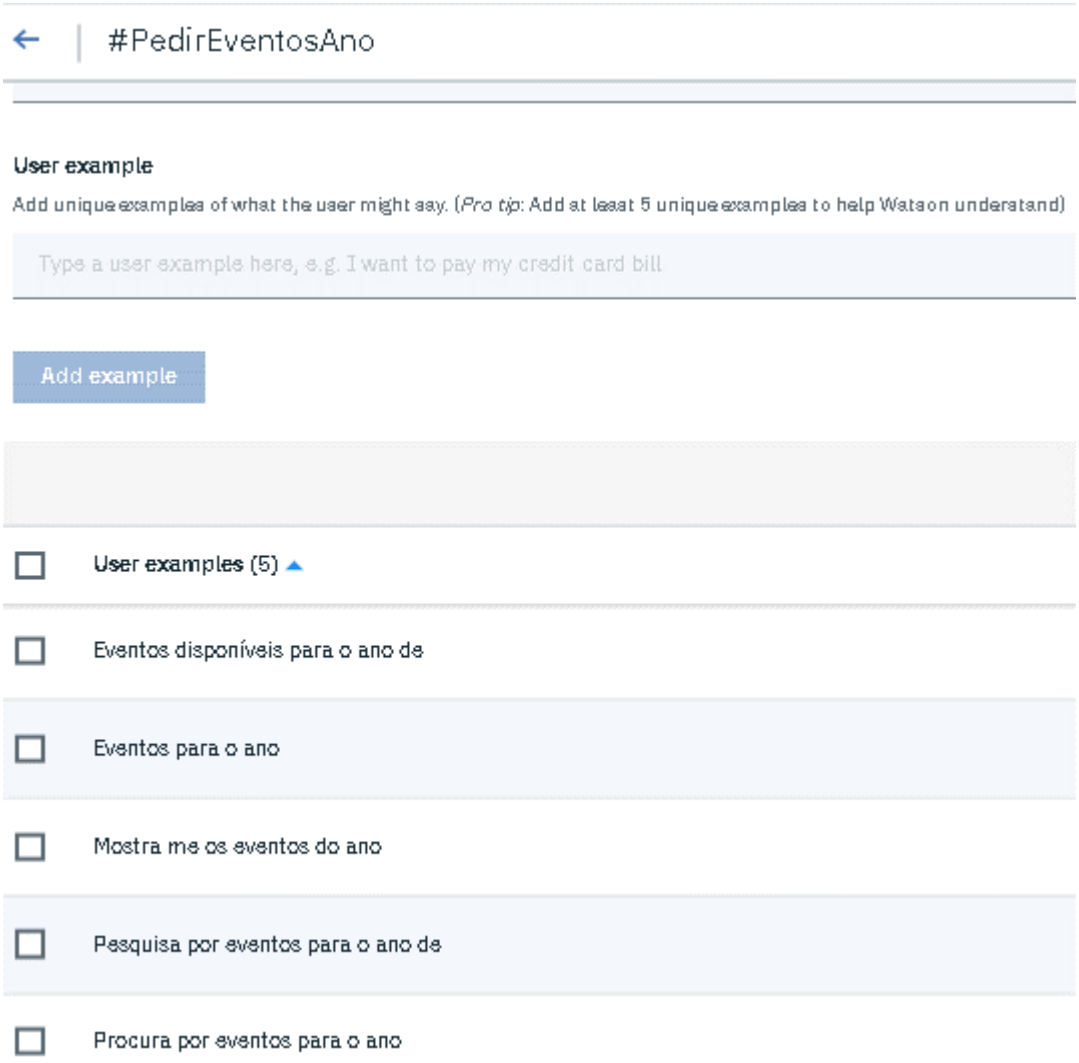

Figura 31- Intent "PedirEventosAno"

Na figuram 31 e 32 apresentam os exemplos que os utilizadores podem usar para acionar a intent “PedirEventos”.

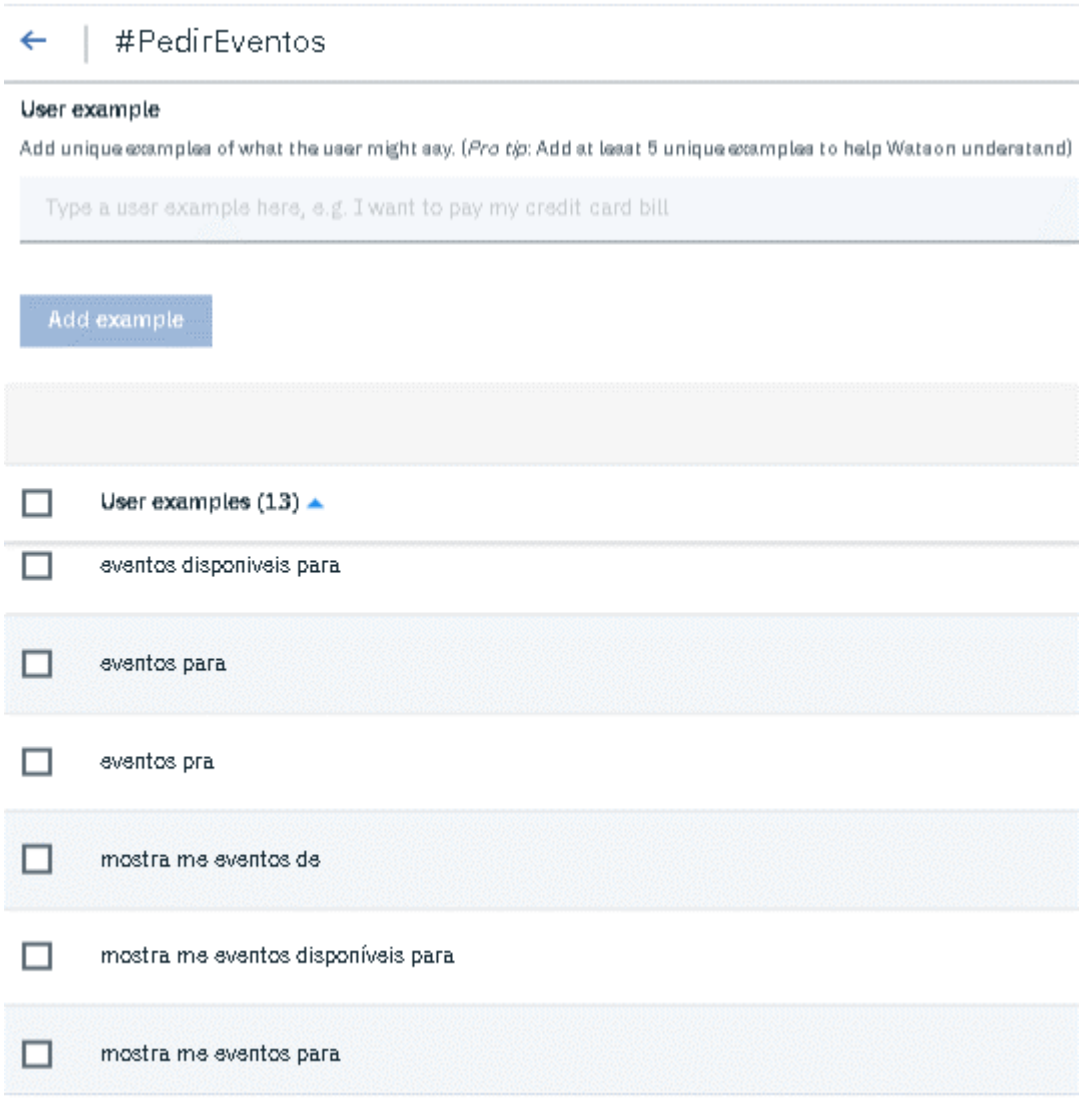

Figura 32- Intent “PedirEventos”

Figura 33- Intent "PedirEventos"

Na figura 33 apresenta os exemplos que os utilizadores podem usar para acionar a intent "DetalhesEventos".

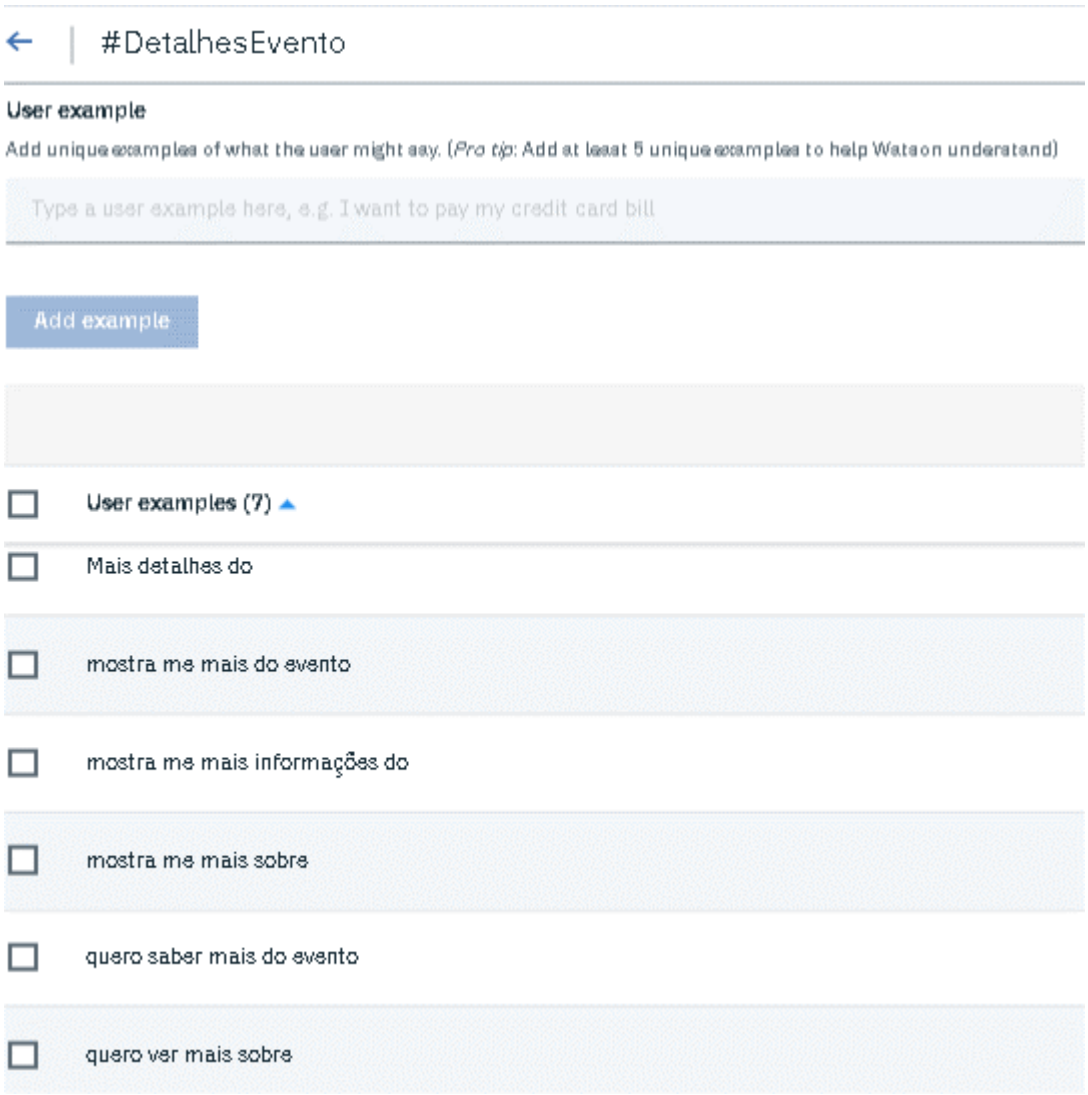

Figura 34- Intent "DetalhesEventos"

Na figura 34 apresenta os exemplos que os utilizadores podem usar para acionar a intent "Agradecimento".

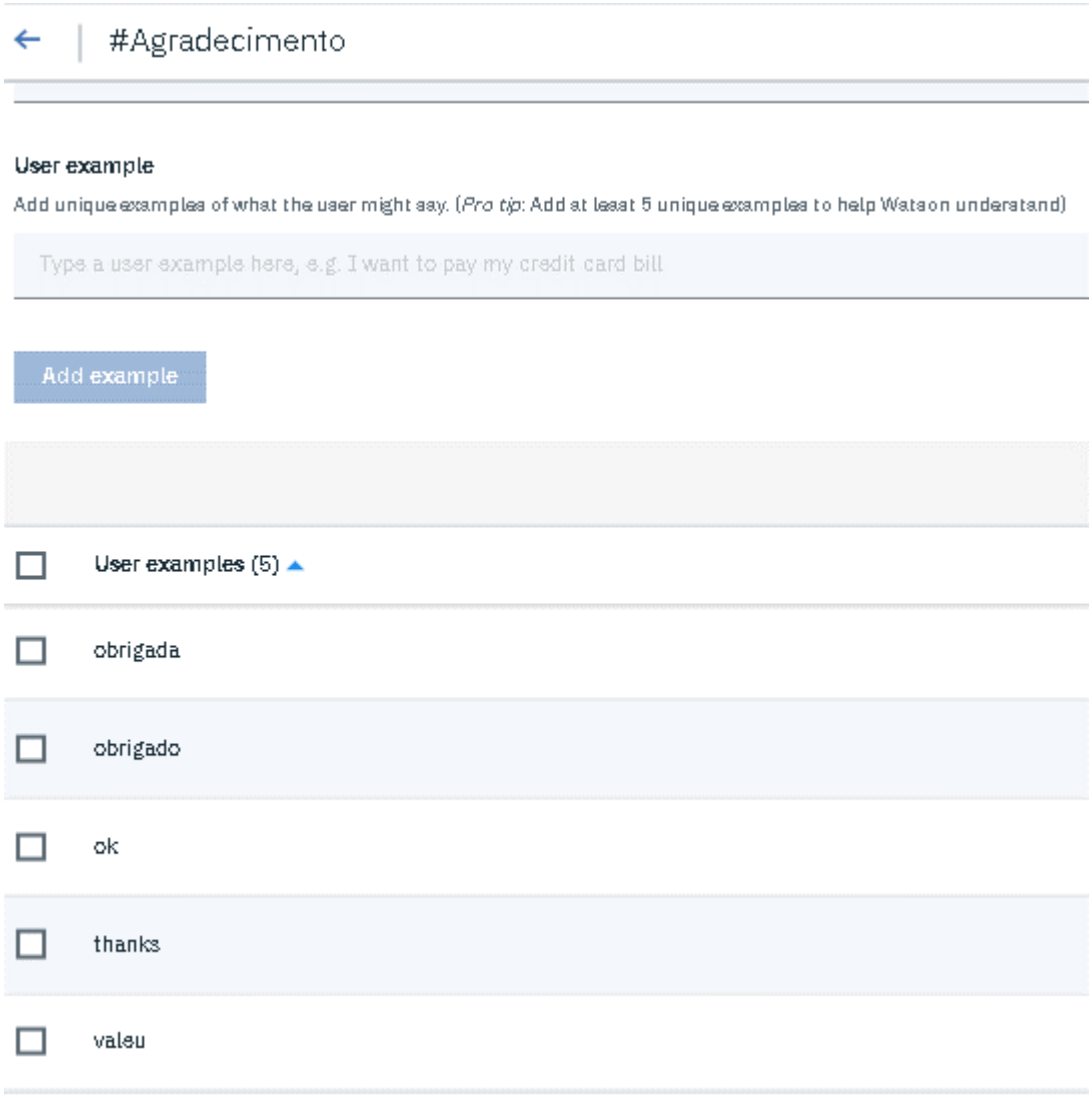

Figura 35- Intent "Agradecimento".

Anexo A6

Condições definidas para as diferentes formas de pesquisa que o utilizador pode utilizar.

Na figura 35 e 36 apresentam as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pelo ano.

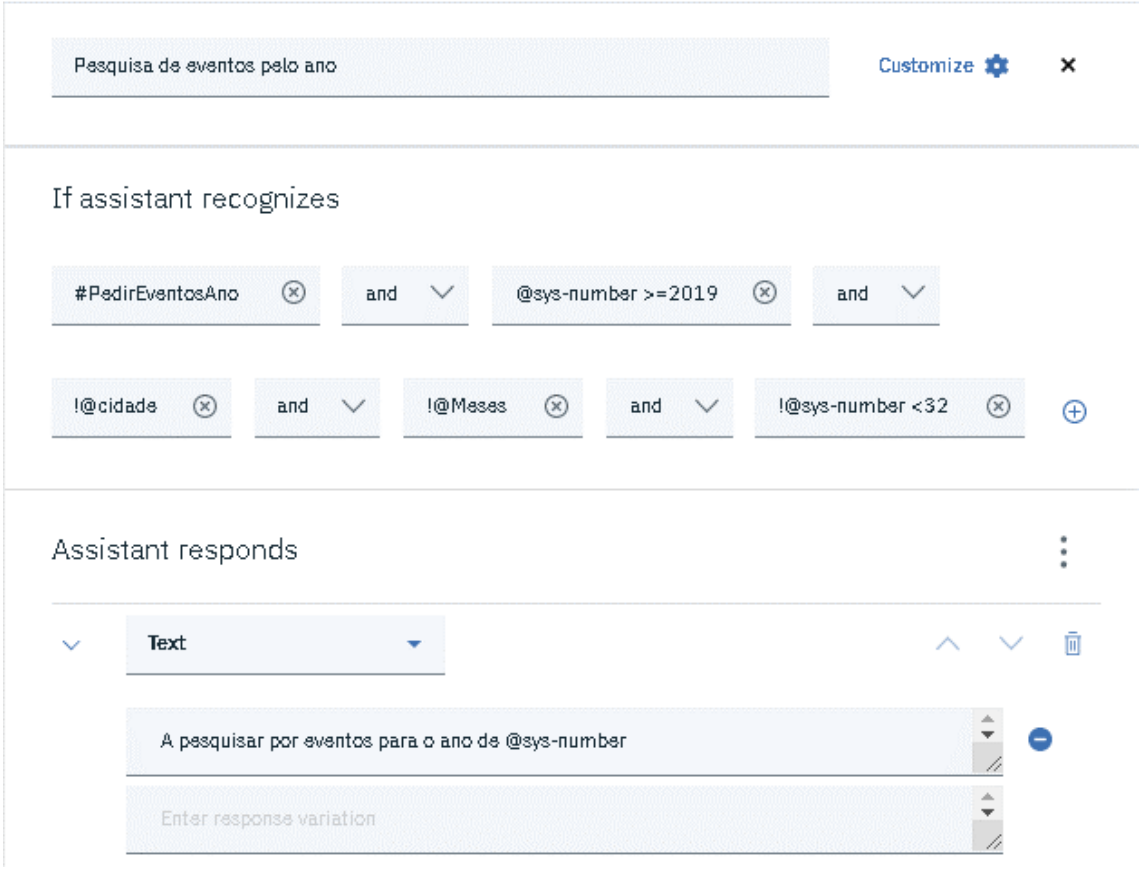

Figura 36- Condições: Pesquisa pelo ano I

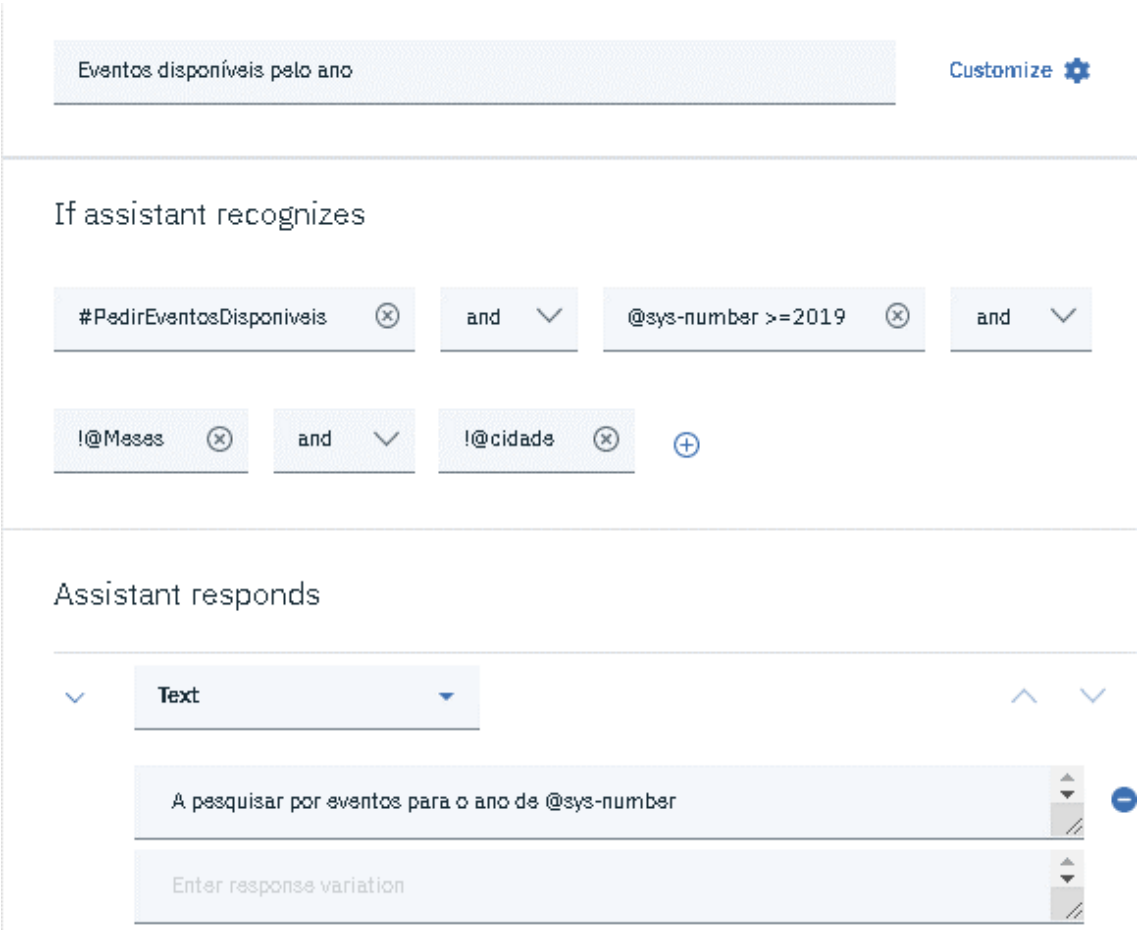

Figura 37- Condições: Pesquisa pelo ano II

Na figura 37, 38, e 39 apresentam as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pelo dia e mês.

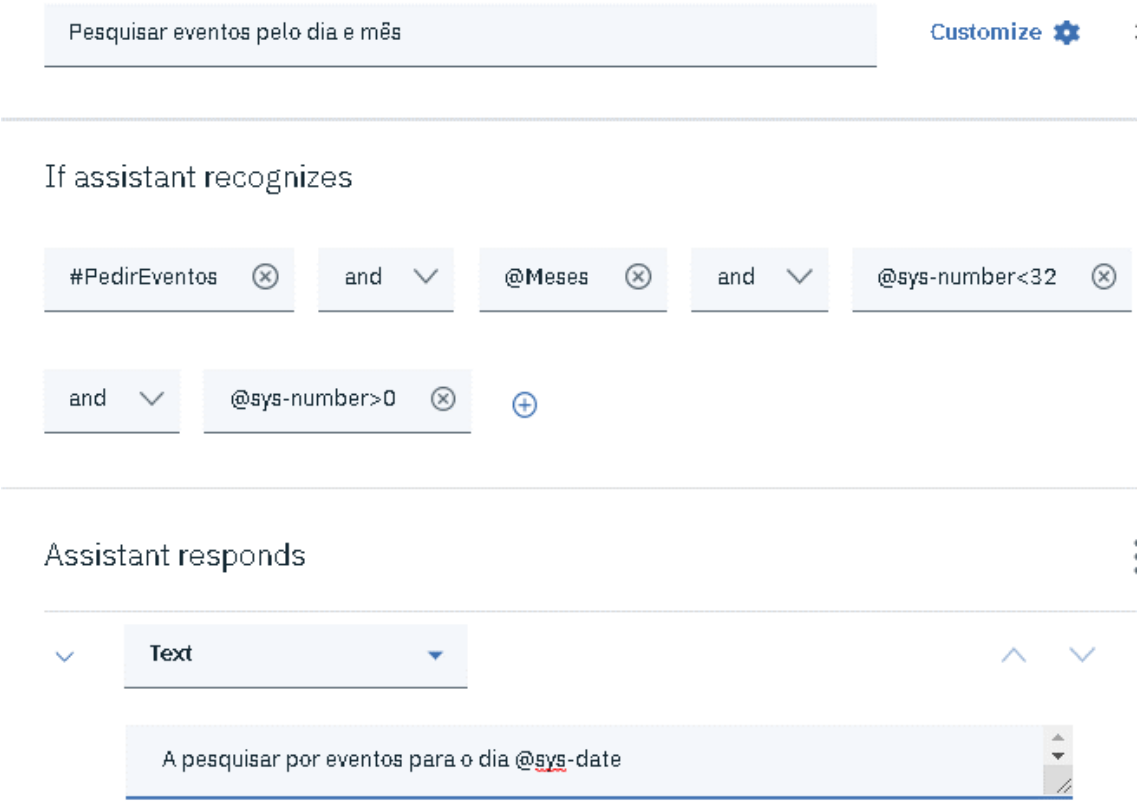

Figura 38- Condições: Pesquisa pelo dia e mês I

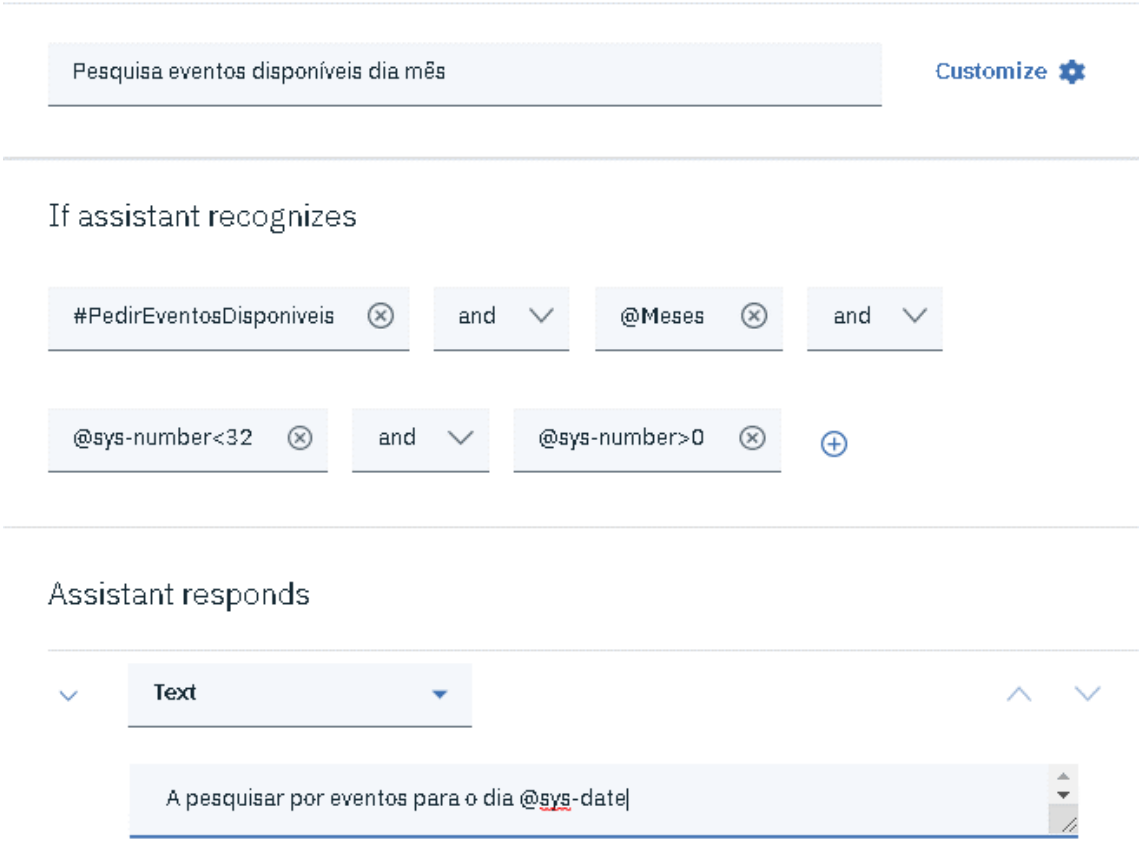

Figura 39- Condições: Pesquisa pelo dia e mês II

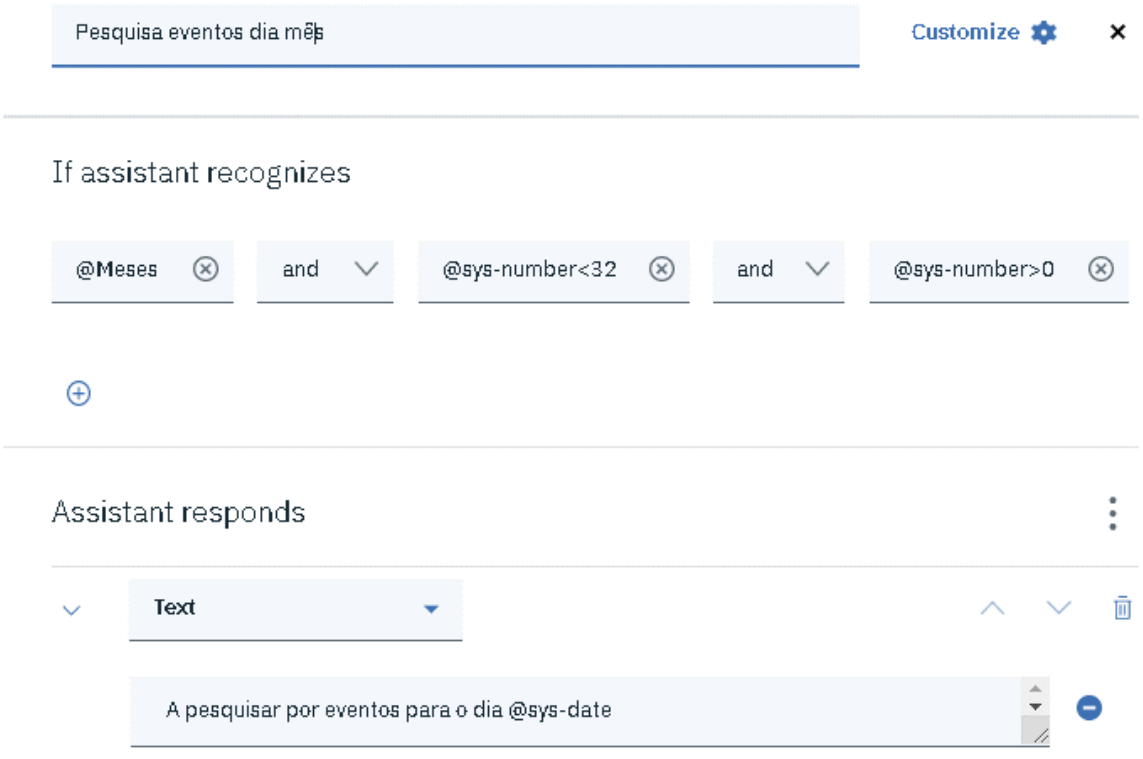

Figura 40- Condições: Pesquisa pelo dia e mês II

Na figura 41, 42, e 43 apresentam as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pelo mês.

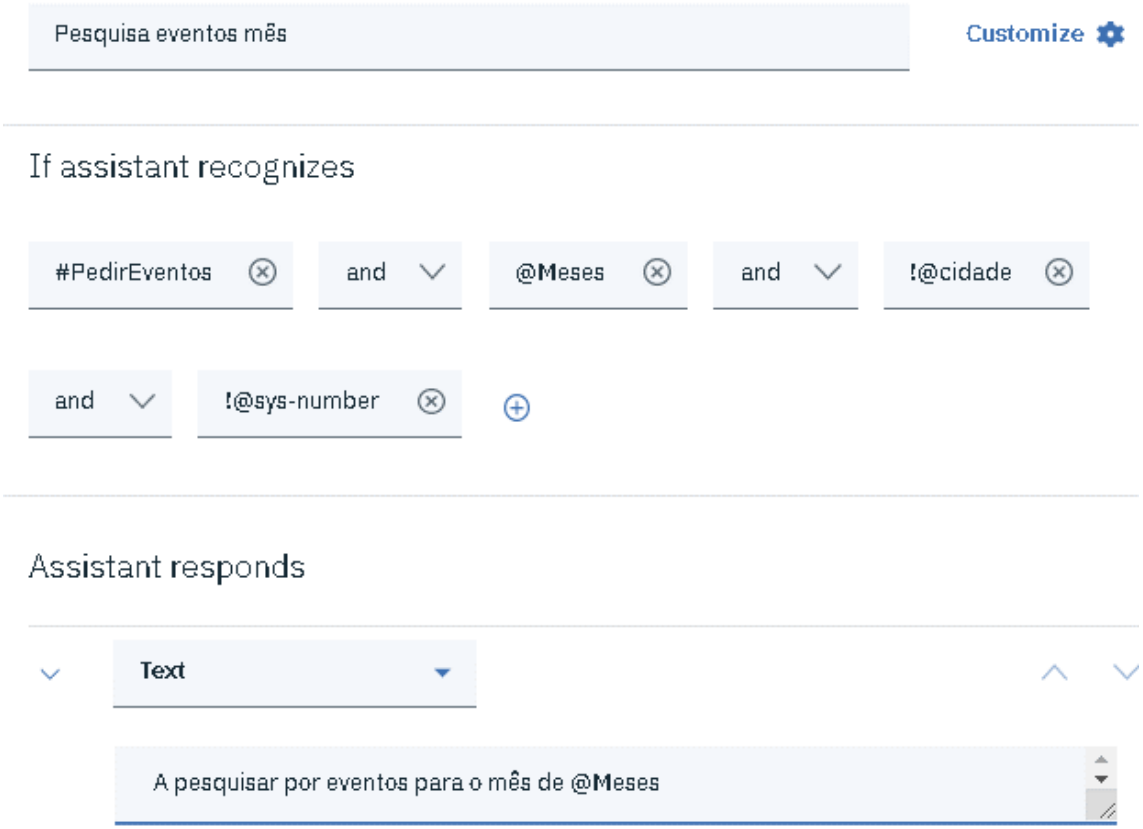

Figura 41- Condições: Pesquisa pelo mês I

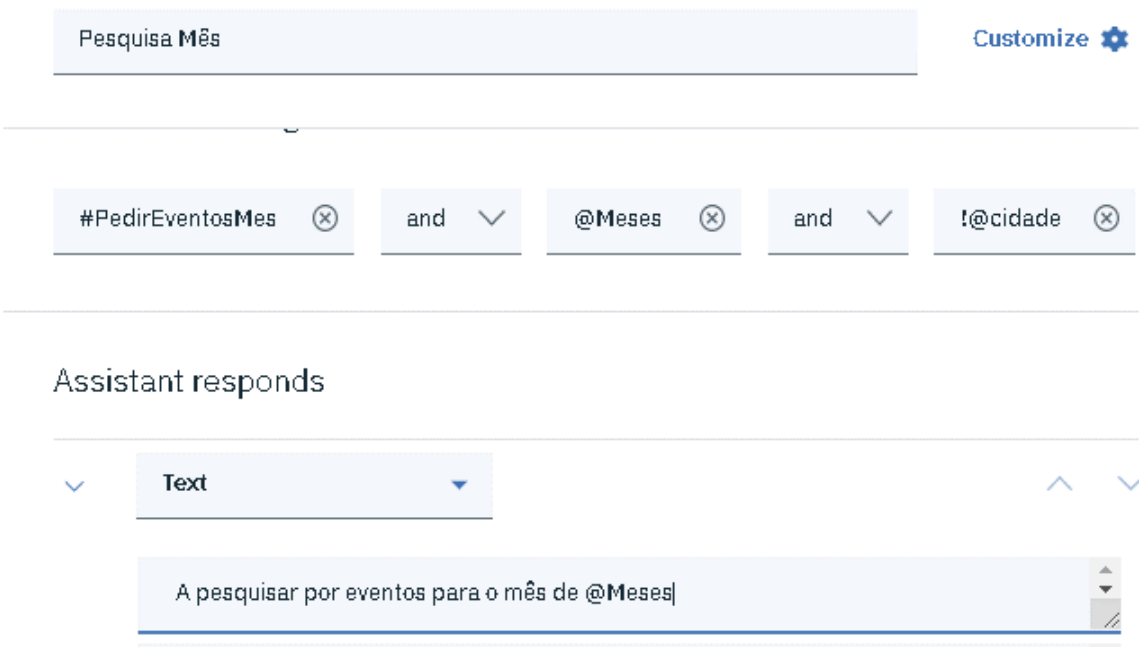

Figura 42- Condições: Pesquisa pelo mês II

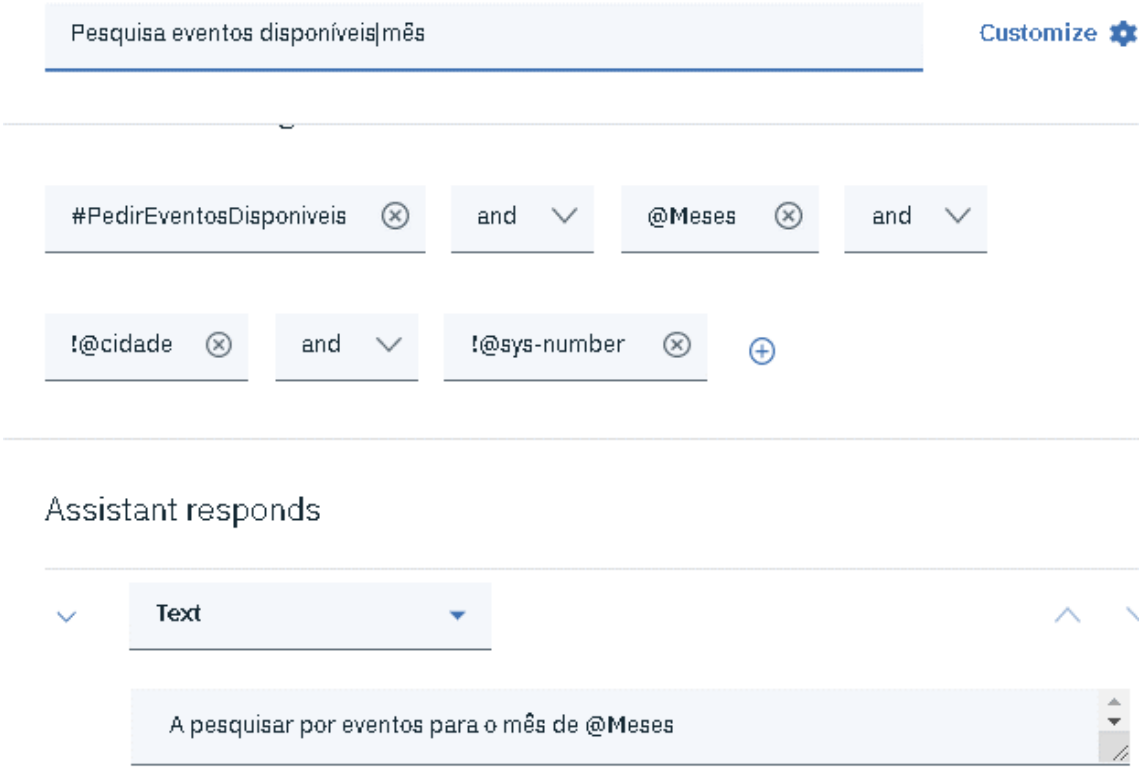

Figura 43- Condições: Pesquisa pelo mês III

Nas figuras 43, 44 e 45 apresentam as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pela cidade.

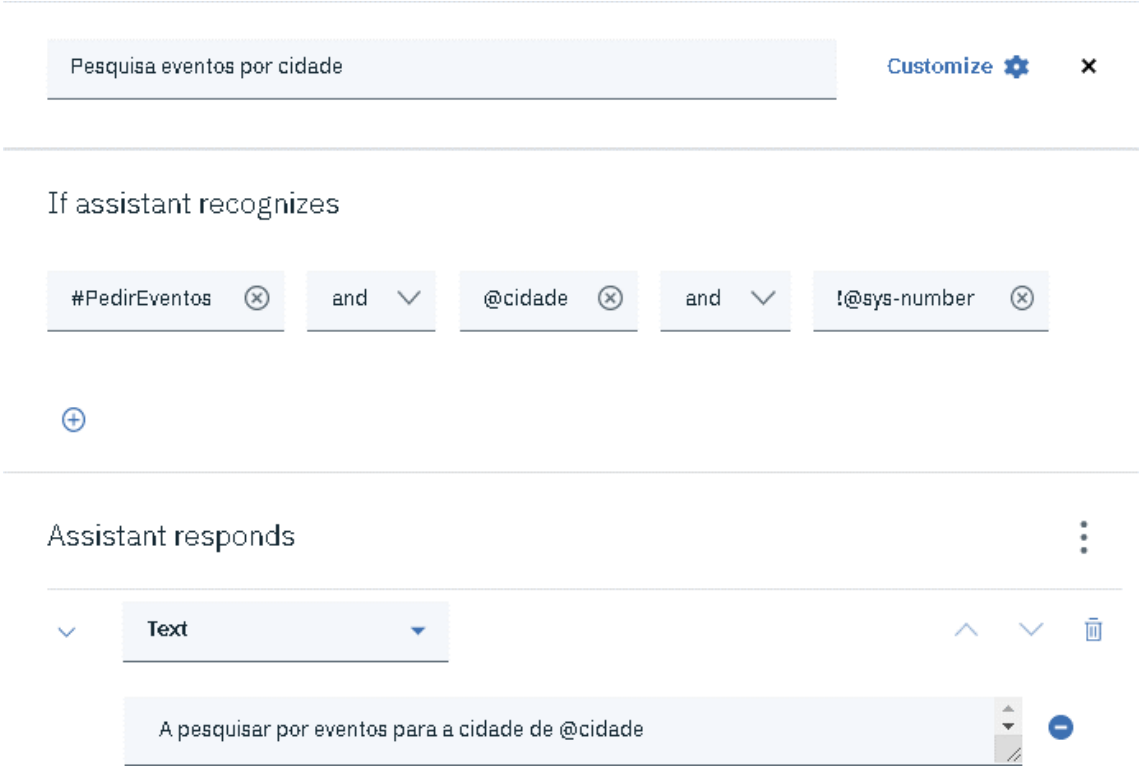

Figura 44- Condições: Pesquisa pela cidade I

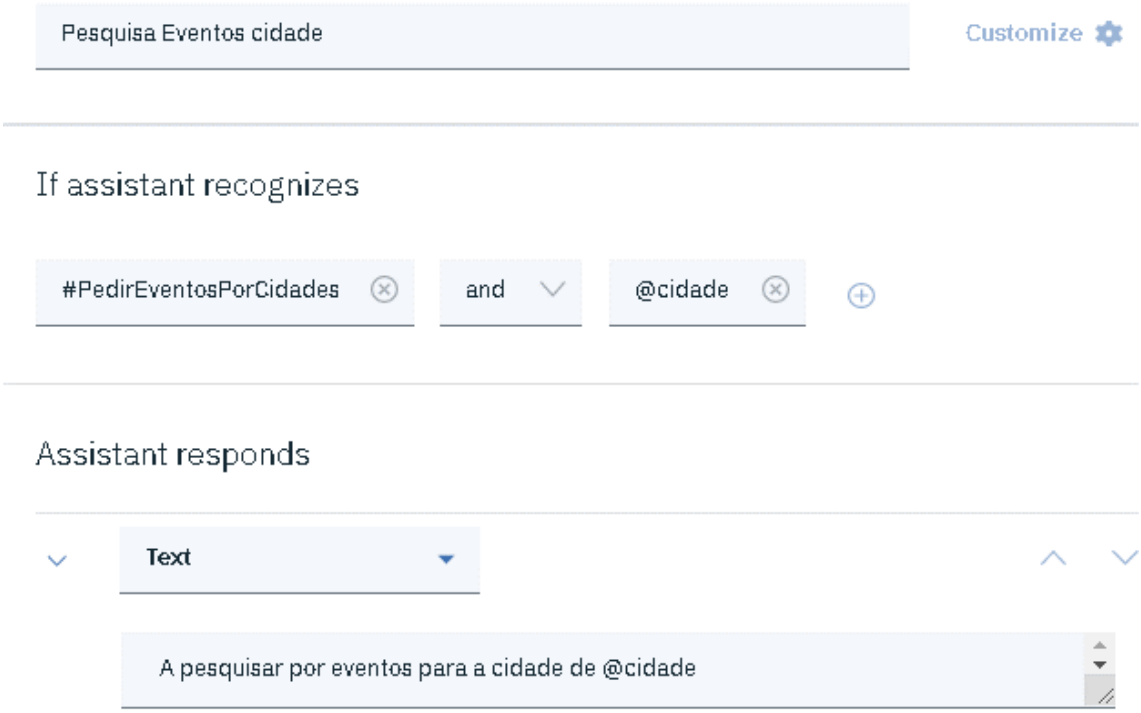

Figura 45- Condições: Pesquisa pela cidade II

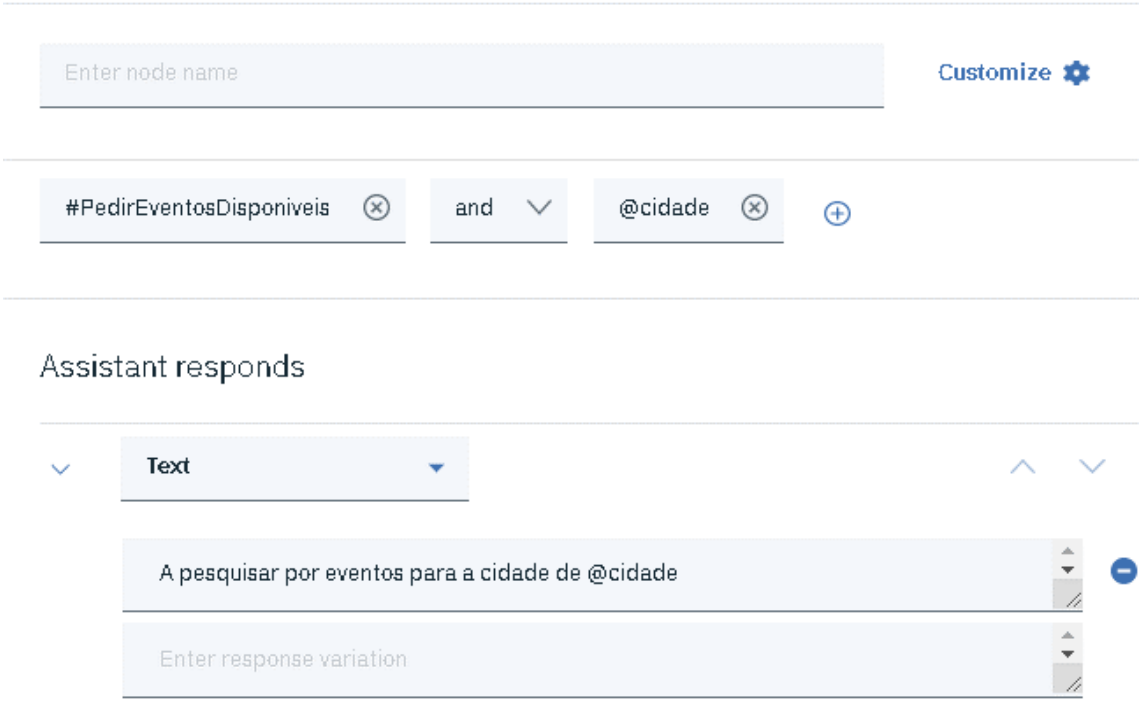

Figura 46- Condições: Pesquisa pela cidade III

Na figura 46 apresenta as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pelo mês e ano.

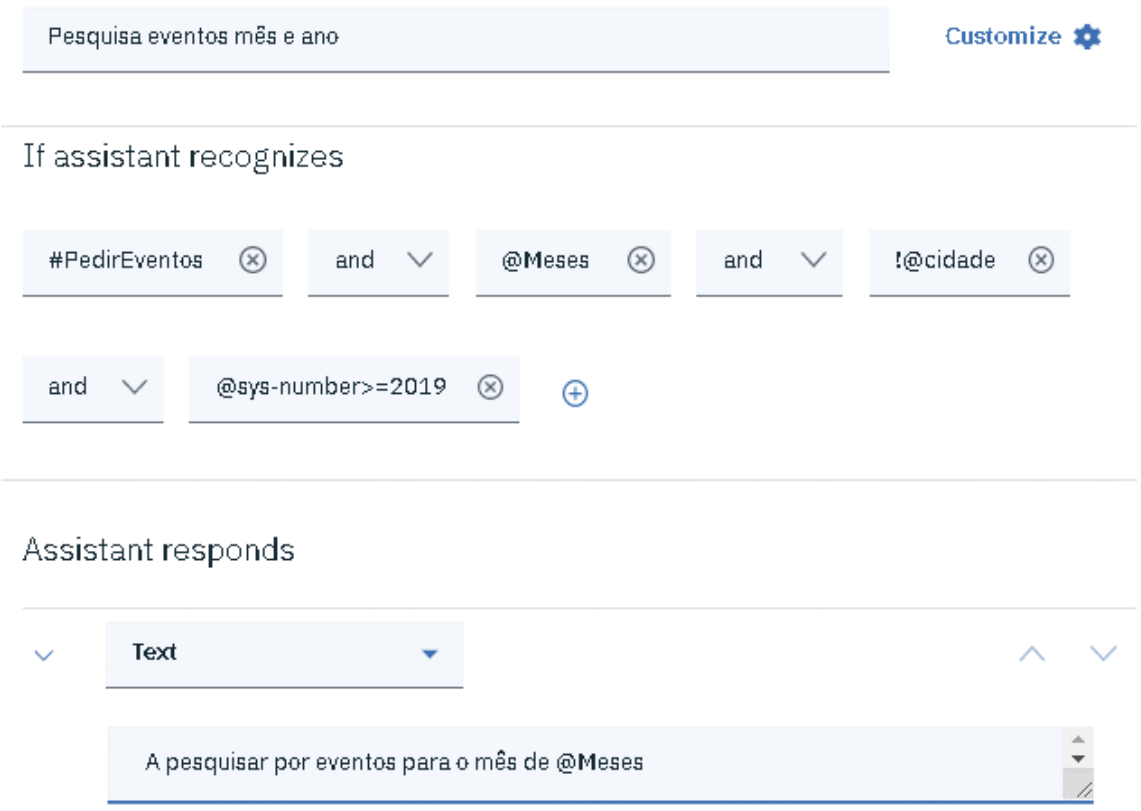

Figura 47- Condições: Pesquisa pelo mês e ano

Nas figuras 47, 48 e 49 apresenta as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pela cidade e mês.

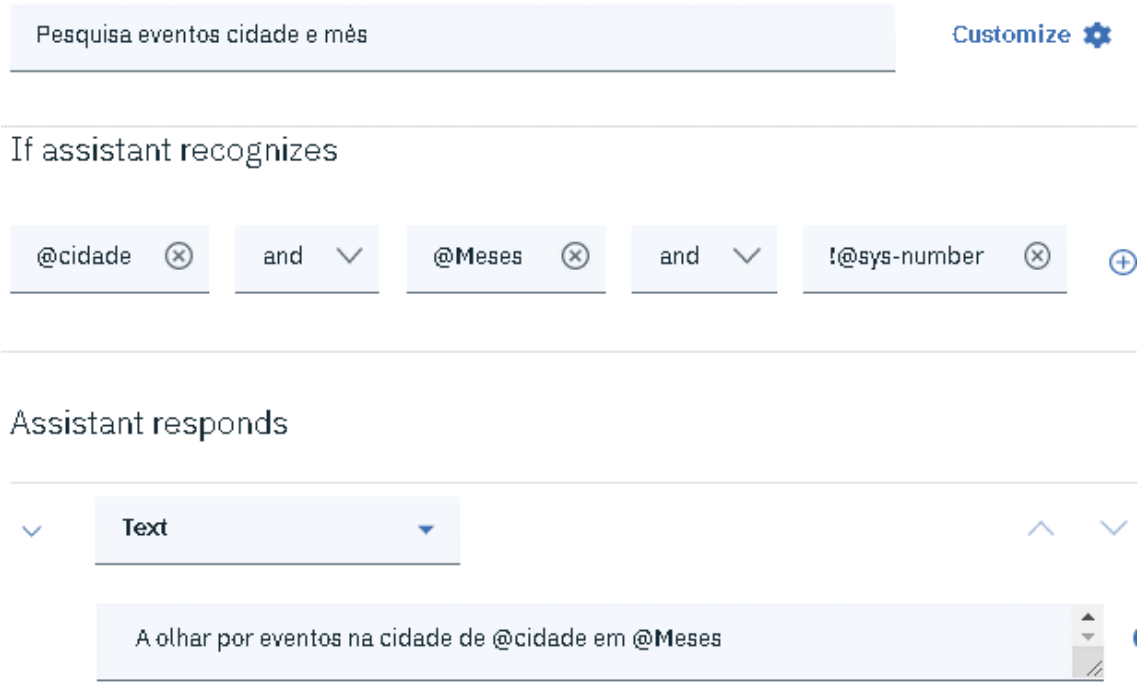

Figura 48- Condições: Pesquisa pela cidade e mês I

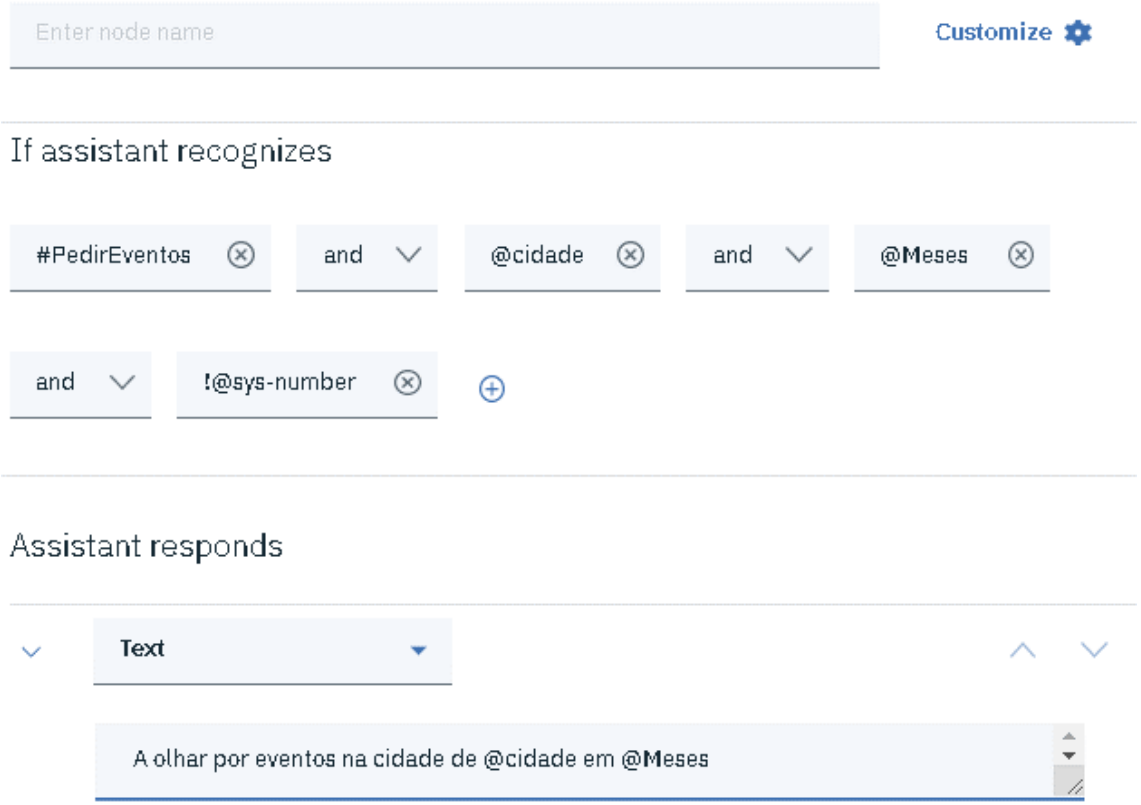

Figura 49- Condições: Pesquisa pela cidade e mês II

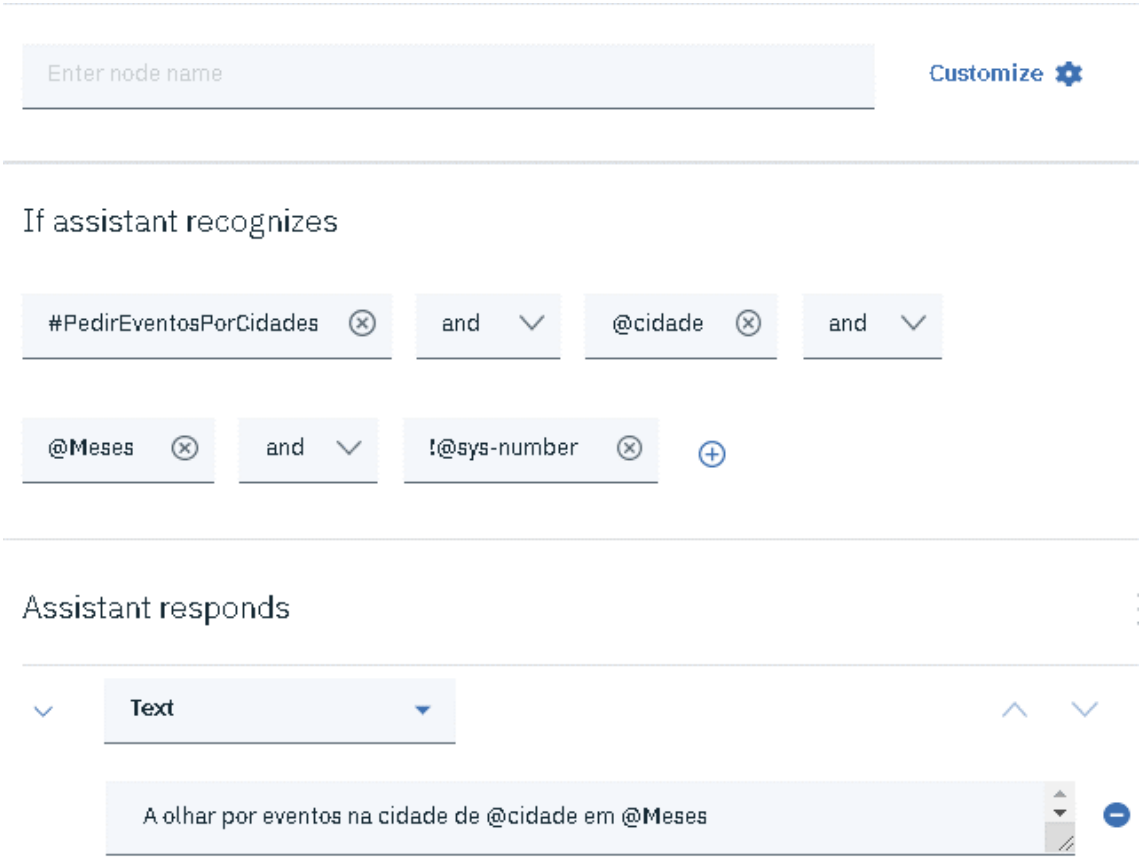

Figura 50- Condições: Pesquisa pela cidade e mês III

Nas figuras 50 apresenta as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pela cidade e ano.

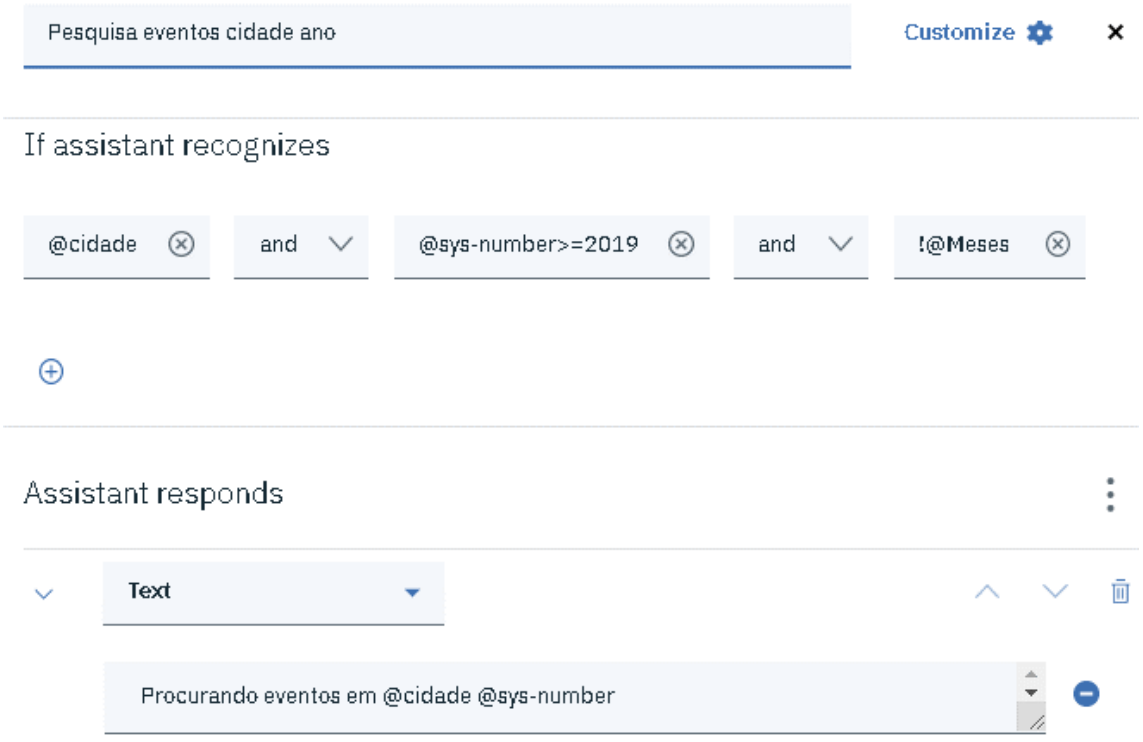

Figura 51- Condições: Pesquisa pela cidade e ano

Nas figuras 51 apresenta as condições que a entrada do utilizador deve cumprir para acionar a pesquisa de eventos pela cidade, mês e ano.

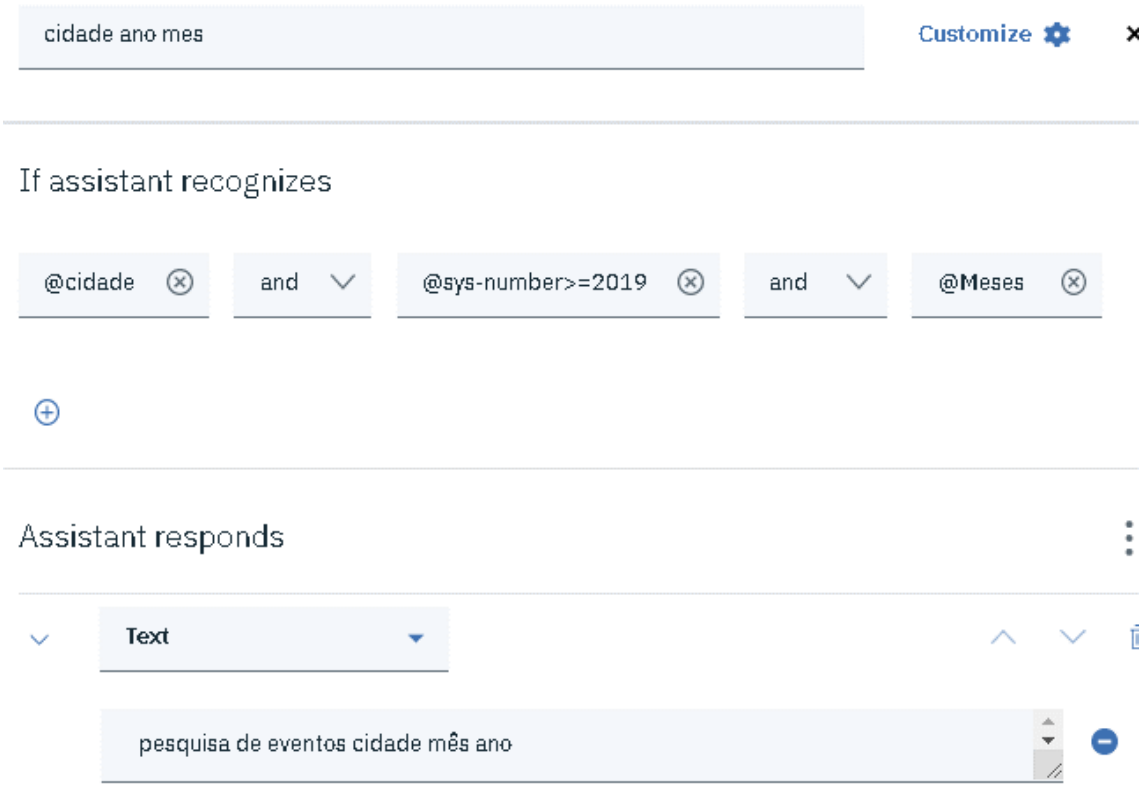

Figura 52- Condições: Pesquisa pela cidade, mês e ano

Nas figuras 51 apresenta as condições que a entrada do utilizador deve cumprir para acionar os agradecimentos.

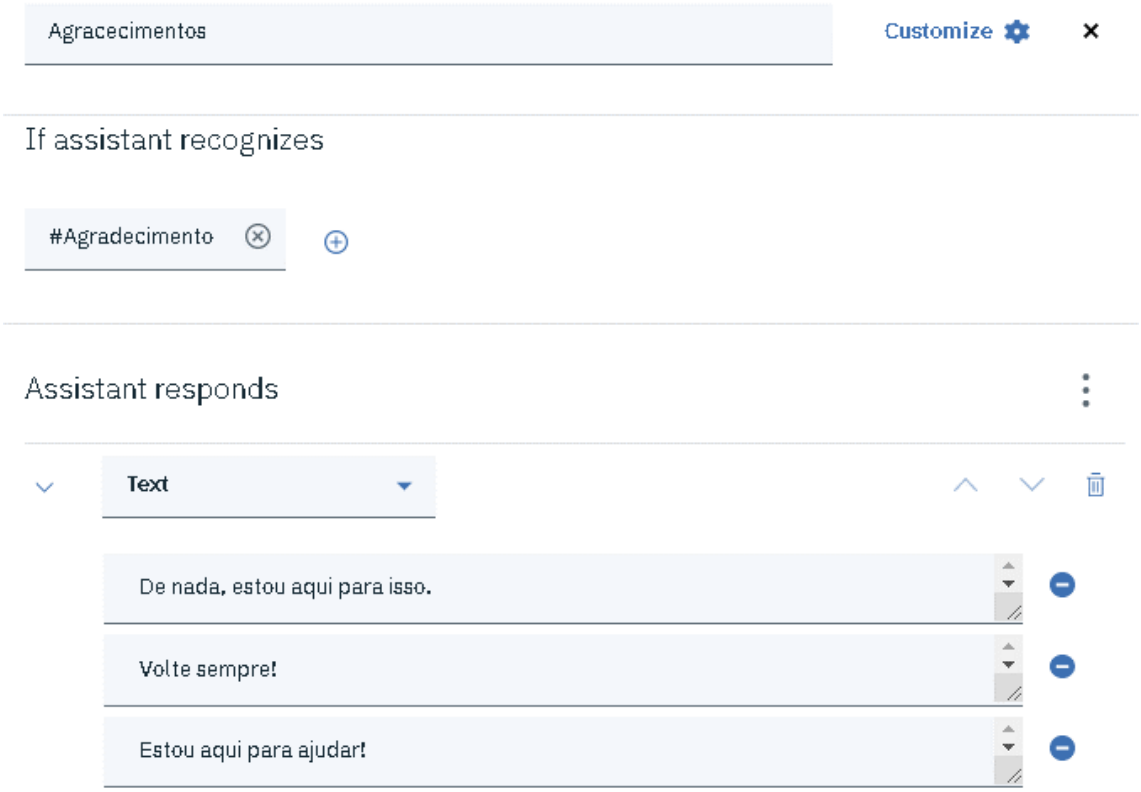

Figura 53- Condições: Agradecimentos

Na figura 53 apresenta a condição para quando a entrada do utilizador não cumpre nenhuma das condições definidas.

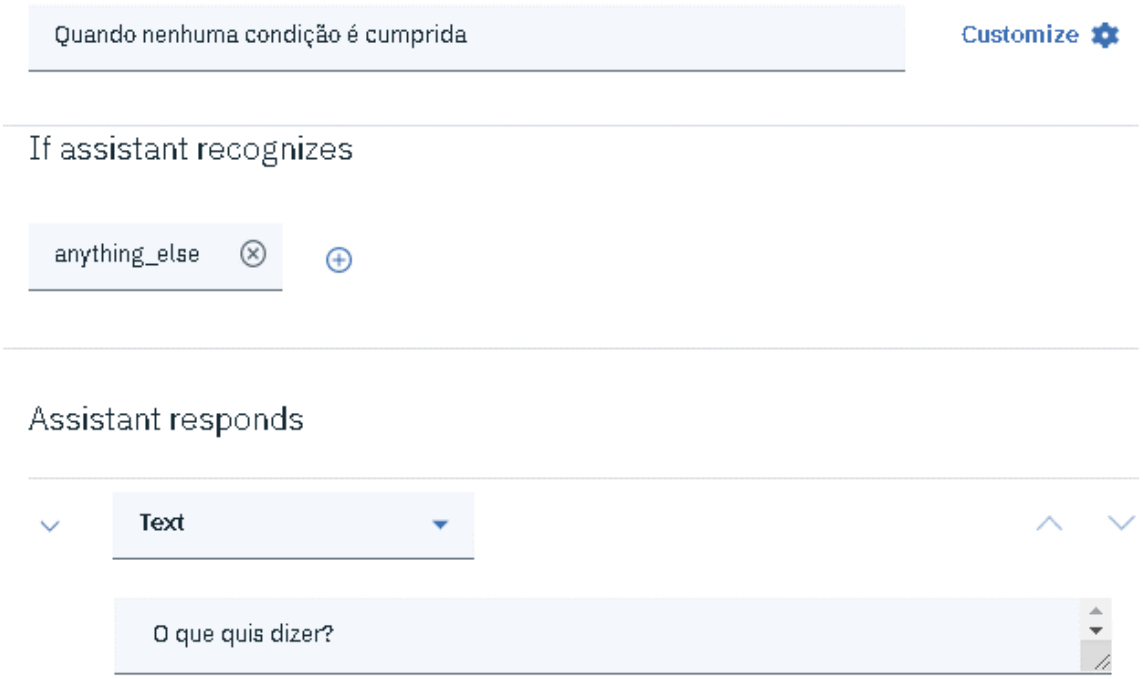

Figura 54- Condições para quando não se cumpre nenhuma condição

Anexo A7
Janela do botão “?”

O botão “?” quando pressionado abre uma janela que tens instruções para os utilizadores quanto as formas de pesquisa. A figura 54 apresenta a janela mencionada.

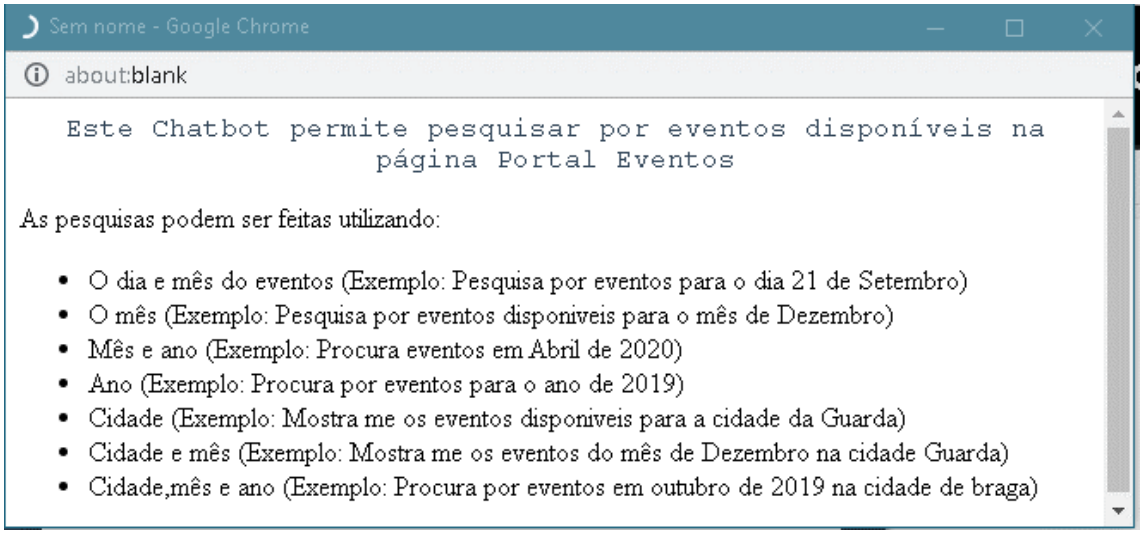

Figura 55- Janela de ajuda ao utilizador