

# Desenvolvendo Chatbots com Watson Conversation

Stéfany Mazon ([smazon@br.ibm.com](mailto:smazon@br.ibm.com))

06/Abr/2018

Developer Advocate & Tech Community Engagement  
IBM

Para falar de Watson vale um breve histórico: o Watson foi lançado em 2011 e em sua primeira aparição no Jeopardy, ganhando de dois super especialistas do jogo. Posteriormente, com muito estudo e...

## Mas o que é "esse Watson"?

Para falar de Watson vale um breve histórico: o Watson foi lançado em **2011** e em sua primeira aparição no **Jeopardy**, ganhando de dois super especialistas do jogo. Posteriormente, com muito estudo e desenvolvimento, o Watson tornou-se **APIs** disponíveis na [IBM Cloud](#).

O Watson Conversation, especificamente, trata-se de uma **API para desenvolvimento de Bots**, com uma **interface simples** para que até mesmo uma pessoa que não seja de TI consiga desenvolver e ensinar conteúdo ao bot.

## O que faremos aqui?

Feitas as devidas apresentações, neste tutorial faremos um chatbot para pedido de pizza! Nesse caso, **mais importante do que o tema do Bot é o entendimento da interface do Watson Conversation**. O tutorial ficou um pouco longo pois explicarei cada parte da API, qualquer dúvida só comentar ok?

## Criando a API:

Para criar a API você precisa criar uma conta na IBM Cloud a qual te dará acesso à diversos serviços tanto de Watson como de Infra e Plataforma para sempre!! ( não , **você não terá que colocar o seu cartão de crédito ao utilizar a camada free**).

Pois bem, com a conta criada você terá que ir no catálogo e lá no final da página selecionar o serviço de Conversation:

The screenshot shows the IBM Cloud Catalog interface. On the left is a navigation sidebar with categories like Infrastructure, Platform, and Watson. The main area displays the 'Watson' section with the heading 'Build cognitive apps that help enhance, scale, and accelerate human expertise.' Below this, several Watson services are listed in a grid. The 'Conversation' service is highlighted with a red rectangular box. It includes an icon of two speech bubbles, the title 'Conversation', a description 'Add a natural language interface to your application to automate', and two buttons labeled 'Lite' and 'IBM'. Other visible services include Discovery, Knowledge Studio, Language Translator, Natural Language Classifier, Natural Language Understanding, Personality Insights, Speech to Text, Text to Speech, Tone Analyzer, and Visual Recognition, each with their respective icons, descriptions, and 'Lite'/'IBM' buttons.

Ao clicar no Conversation abrirá a página abaixo, no meu caso eu mudei o nome do serviço para esse tutorial, mas essa ação não é necessária.

View all

## Conversation

Add a natural language interface to your application to automate interactions with your end users. Common applications include virtual agents and chat bots that can integrate and communicate on any channel or device. Train Watson Conversation service through an easy-to-use web application, designed so you can quickly build natural conversation flows between your apps and users, and deploy scalable, cost effective solutions.

**Service name:**  
tutorial-BotsBR

**Choose a region/location to deploy in:**  
US South

**Choose an organization:**  
smazon@br.ibm.com

**Choose a space:**  
dev

### Images

Click an image to enlarge and view screen captures, slides, or videos. Screen caps show the user interface for the service after it has been provisioned.

**View Docs**

**AUTHOR** IBM  
**PUBLISHED** 03/02/2018  
**TYPE** Service  
**LOCATION** Sydney, Germany, United Kingdom, US South

**Need Help?**  
[Contact IBM Cloud Sales](#)

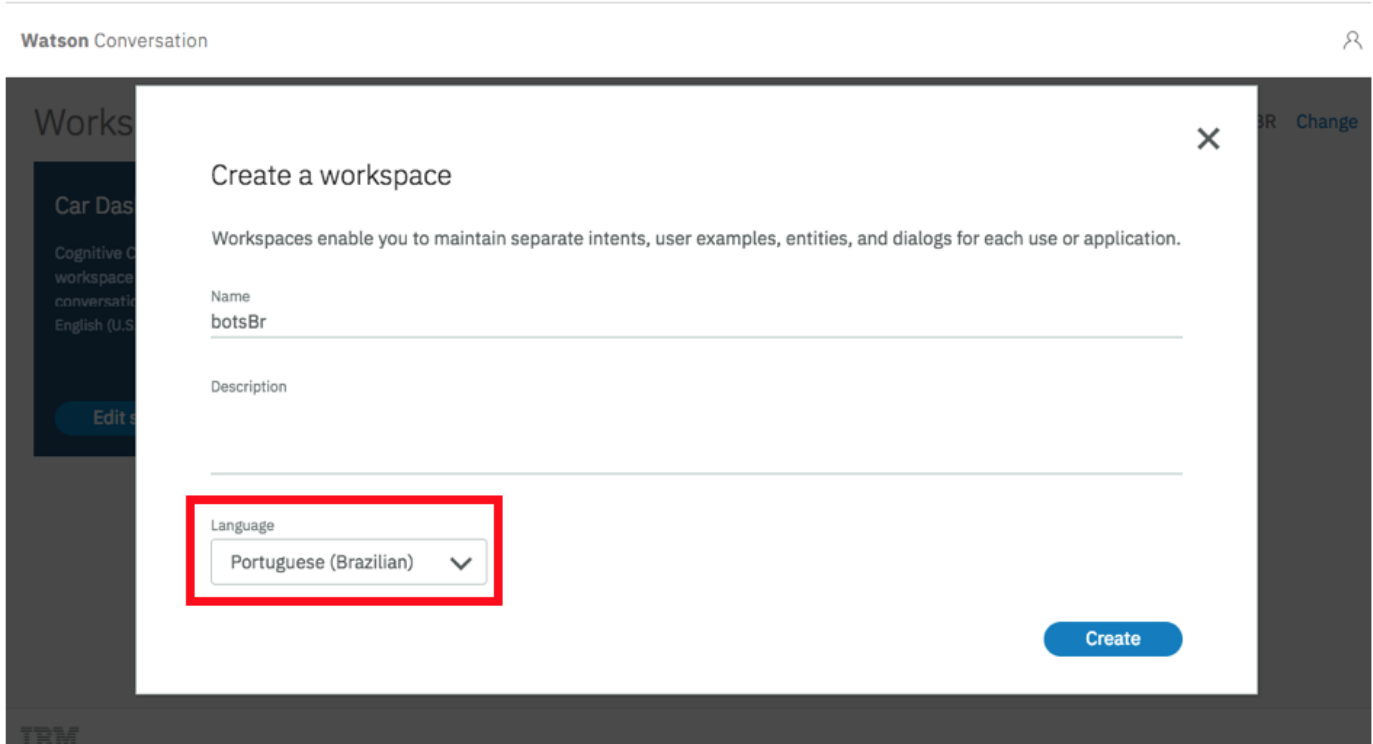
**Estimate Monthly Cost**  
[Cost Calculator](#)

**Create**

Após clicar em Create, você será direcionado à página do Conversation. É legal saber que neste caso trabalharemos com o **Toolkit** mas é possível fazer o desenvolvimento na mão com os **SDKs de Watson** a utilizando as credenciais da API.

Dentro do toolkit do Watson Conversation faremos nosso primeiro **Workspace**, que nada mais é que um **ambiente de conhecimento específico** para o seu bot.

Outro ponto importante é não esquecer de selecionar **a língua para português!** Deste modo ele utilizará todos os modelos de NLP (Natural Language Processing) próprios da nossa língua.



Watson Conversation

Works

Car Das

Cognitive C  
workspace  
conversati  
English (U.S

Edit

IBM

Create a workspace

Workspaces enable you to maintain separate intents, user examples, entities, and dialogs for each use or application.

Name  
botsBr

Description

Language  
Portuguese (Brazilian) ▼

Create

Pronto, já temos nosso Conversation configurado. Agora teremos que populá-lo com **intenções e entidades**.

## Intenções:

*Uma intenção trata-se da **ação** atrelada às perguntas realizadas pelo usuário. Isto é, **o que o usuário procura ao falar algo**; e sim, podemos falar a mesma coisa de diversas maneiras, sendo praticamente impossível treinar todas as opções de interação. Deste modo, no Conversation nós damos exemplos de frases e posteriormente o sistema generaliza para identificação de outras intenções comuns.*

Neste caso eu adicionei algumas opções de como o usuário entraria em contato com o bot para pedir uma pizza. E caso o usuário falasse : "to querendo uma pizza aí meu" (mesmo que esse exemplo não está no meu treinamento), ele entenderia que a **intenção** é **#pedir\_pizza**.

[<](#) | #pedir\_pizza

**Intent name**  
#pedir\_pizza

**Description**  
Add a description to this intent

**Add user examples**  
quero pedir uma pizza

Add example

☐ User examples (4) ▼

☐ eu gostaria de fazer um pedido

☐ iniciar pedido de pizza

☐ me vê uma pizza

☐ por favor quero fazer meu pedido

Feito isso, você pode adicionar diversas outras intenções como **#saudacao**, **#despedida**, **#informações**... Sempre levando em consideração o que você quer que o seu bot saiba responder. Mas, além da intenção temos que adicionar os complementos desta ação, então vamos às entidades.

## Entidades:

Entidades são conhecidas como os **complementos de informação**. Neste exemplo os complementos serão o **sabor da pizza**, o **tipo de massa**, **CEP** e **data de entrega** (poderíamos pedir outras informações, mas para este caso, já é mais do que suficiente).

É legal de se saber que o Watson Conversation tem **system entities** prontas que não precisam ser treinadas. Sendo assim, o primeiro passo é habilitar a **@sys-date** para garantir que o bot entenda quando você falar que quer uma pizza para **amanhã**, no **natal** ou no **ano novo**...

Intents	<b>Entities</b>	Dialog
My entities	<b>System entities</b>	
These are common entities created by IBM that could be used across any use case. They are ready to use as soon as you add them to your workspace. *System entities cannot be edited. <a href="#">Learn more</a>		
Name (5) ▼	Description	Status
> @sys-currency	Extracts currency values from user examples including the amount and the unit. (20 cents)	<input type="checkbox"/> Off
> @sys-date	Extracts date mentions (Sexta-feira)	<input checked="" type="checkbox"/> On
> @sys-number	Extracts numbers mentioned from user examples as digits or written as numbers. (21)	<input type="checkbox"/> Off
> @sys-percentage	Extracts amounts from user examples including the number and the % sign. (15%)	<input type="checkbox"/> Off
> @sys-time	Extracts time mentions (em 10)	<input type="checkbox"/> Off

Caso você queira utilizar as outras entidades de sistema é só habilitá-las!

Pois bem, além das entidades de sistema, teremos que criar as outras entidades como abaixo. Notem que ao colocar os exemplos podemos tanto adicionar **sinônimos**, como no caso de Mussarela—queijo , ou adicionar **patterns** (Expressões Regulares), como no caso do CEP—**[0-9]{5}-[0-9]{3}**

<
@tipo\_massa

Entity name  
@tipo\_massa

Value name  
Massa Grossa

Synonyms  
grossa

Add value

Entity values (1)
Type

Massa fina
Synonyms
fina

<
@sabor

Entity name  
@sabor

Value name  
Pepperoni

Synonyms  
Enter synonym

Add value

Entity values (2)
Type

Atum
Synonyms

Mussarela
Synonyms
queijo

<
@informacao\_entrega

Entity name  
@informacao\_entrega

Value name  
Enter value

Synonyms  
Enter synonym

Add value

Entity values (1)
Type

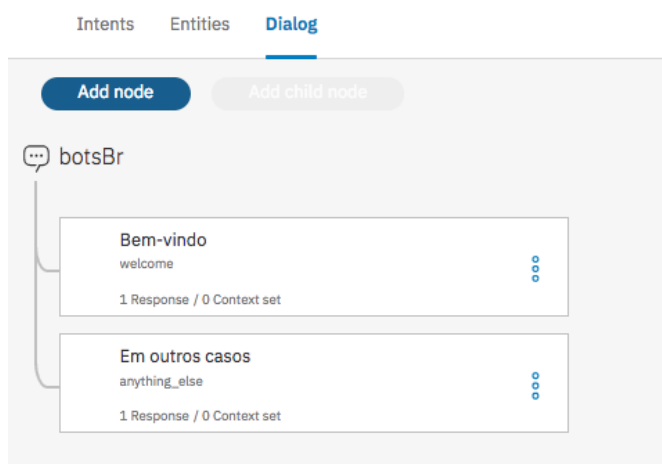
CEP
Patterns
[0-9]{5}-[0-9]{3}

Pronto, o nosso bot está populado, mas agora precisamos criar as regras de resposta e o fluxo de conversa.

## Diálogo:

Ao criar o diálogo note que duas caixas são criadas, a de "Bem-Vindo" e "Em outros casos".

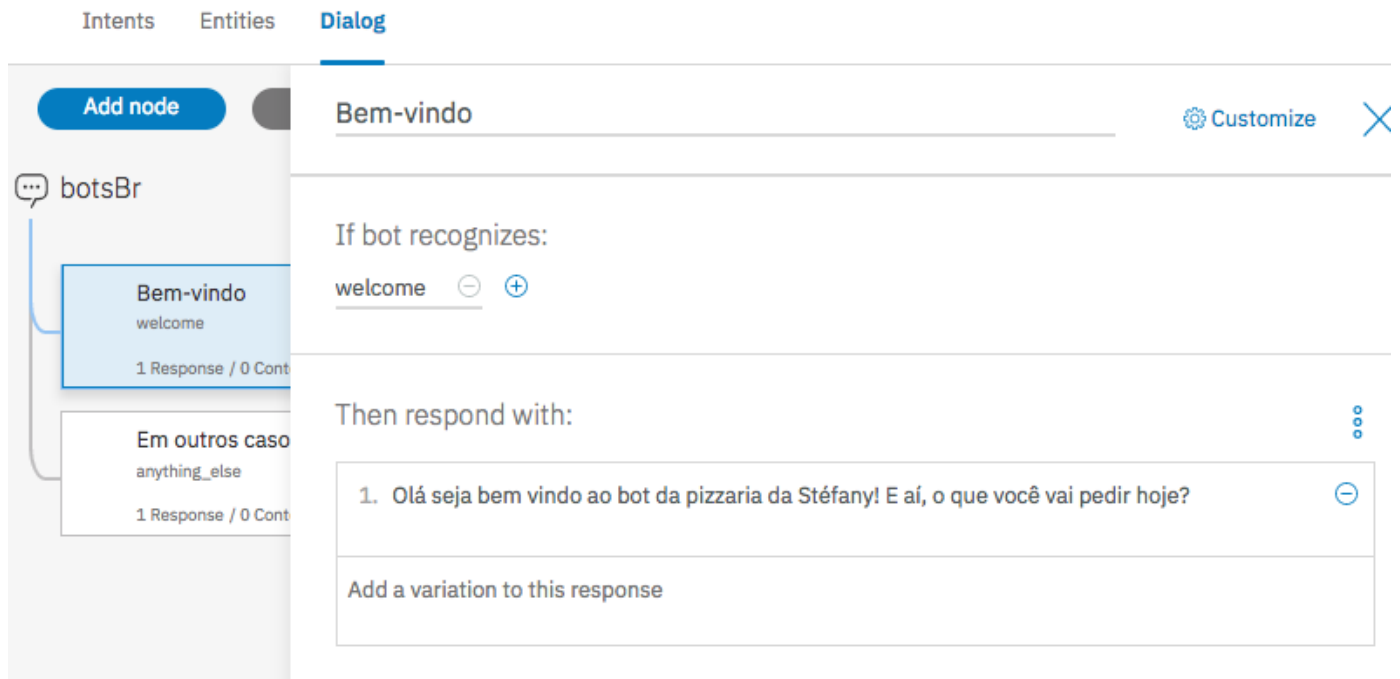
Essas duas variáveis de sistema (**welcome** e **anything else**) são utilizadas para definição da mensagem enviada pelo bot quando:



### a. o usuário entra na interface

b. **o bot não encontra nenhuma resposta correlacionada ao que o usuário digitou** (o famoso "não sei falar sobre isso" ou "sou um robô que está sendo treinado").

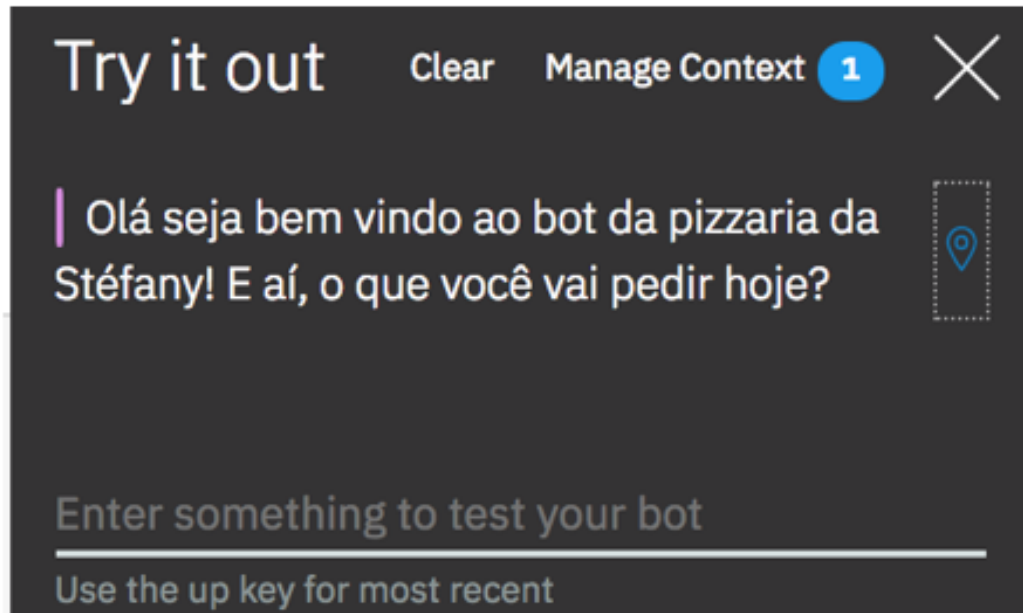
Ao clicar na caixa de welcome podemos trocar a mensagem conforme abaixo:



É possível adicionar variações para a mesma resposta de modo sequencial ou randômico

Para testar as alterações basta clicar no símbolo de “chat” à direita da tela:

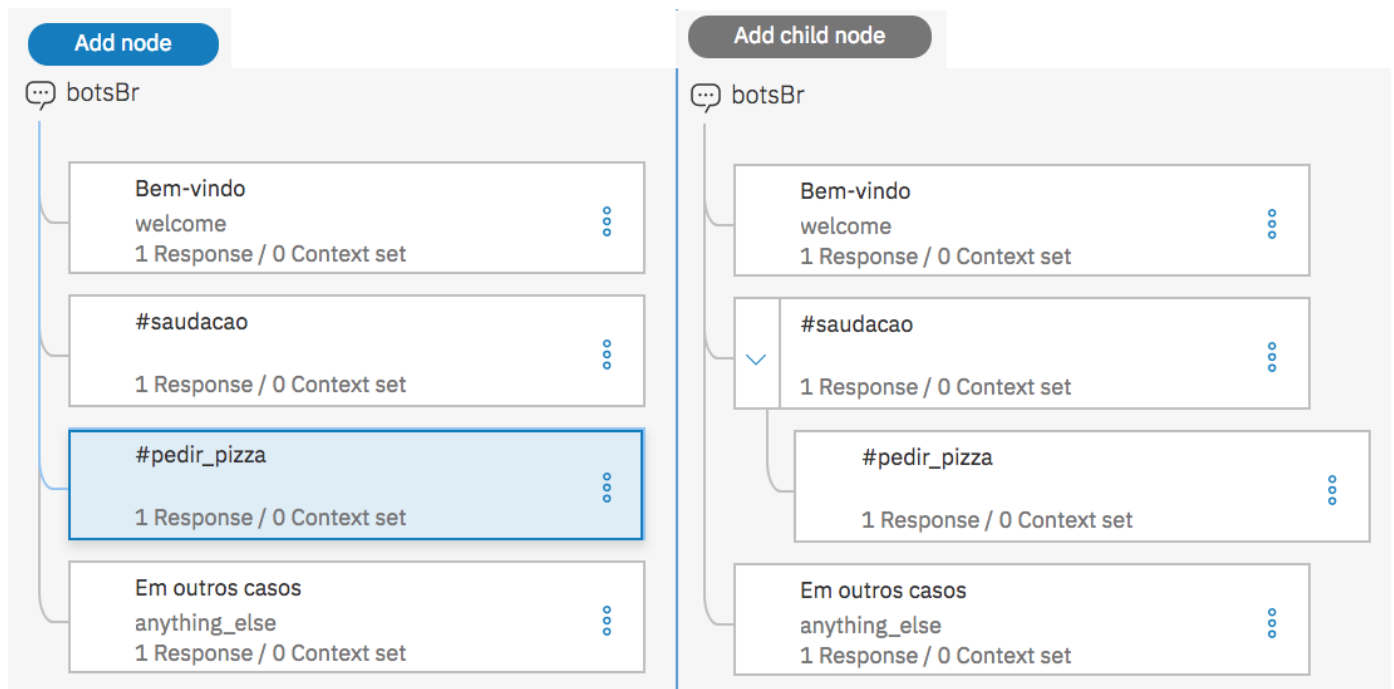




Esse debug será essencial para ver se o bot está sempre caindo no nó correto do diálogo

Agora precisaremos adicionar um novo nó. Notem que há opção de adicionar **nós** ou **nós filhos**. Adicionando um nó ele não terá correlação com o anterior enquanto o nó filho só ocorre se o pai ocorrer primeiro.

Por exemplo: se adicionássemos uma intenção de saudação além da pedir pizza, vejamos o que ocorreria:



No caso à esquerda, não necessariamente o usuário precisa enviar um “oi” para posteriormente pedir uma pizza. Já no caso à direita, o nó de “pedir\_pizza” só ocorre após uma saudação.

Isso é extremamente importante para condições que necessitam **seguir uma sequência**.

Desta forma, para o nosso caso, utilizaremos a primeira opção.

If bot recognizes:

#saudacao  

Then respond with:

1. Opa e aí? Já decidiu o que vai pedir?



Add a variation to this response



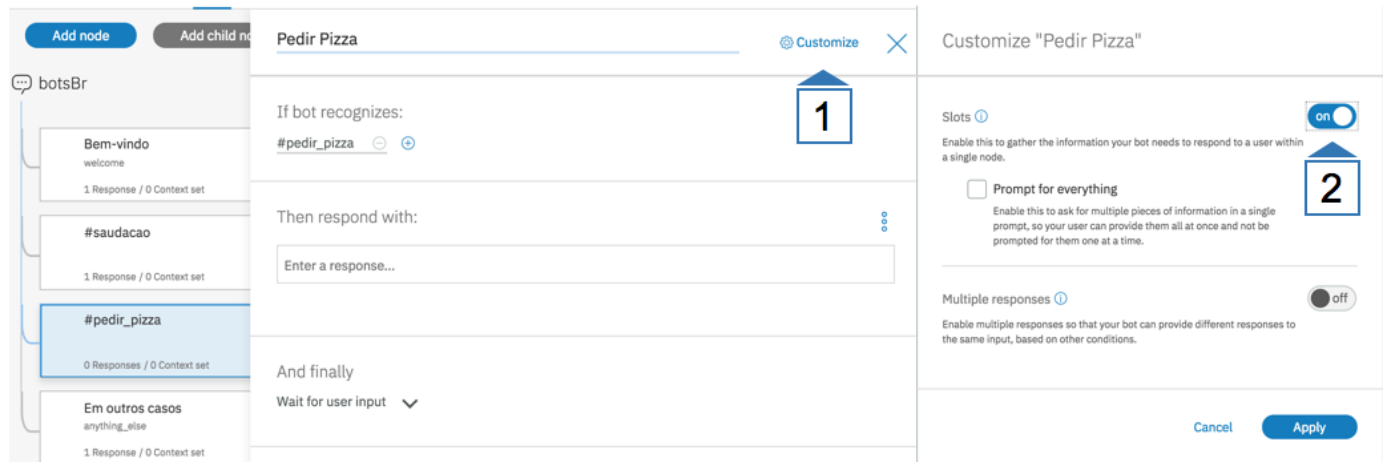
Notem que os nós serão sempre um "if then else"- se ele reconhecer uma condição responda com essa informação, se não passe para o próximo nó.

Até aí simples , Stéfany! Mas como fazemos já que para a pizzaria temos 4 **variáveis a serem coletadas: o sabor, o tipo de massa, a data e o CEP de entrega?**

Vamos criar o nó com a intenção de **pedir\_pizza**, mas ele é um caso particular no qual utilizaremos os **Slots**:

*Slots são utilizados para **coleta de informações ao longo da conversa**, de um modo alternativo à formulários. O mais legal é que ao interagir com o bot o usuário **não precisa enviar as informações em uma ordem específica** pois o slot trata variável por variável.*

Para o nosso caso, o bot coletará as informações necessárias mas de um modo inteligente. Vejam a imagem abaixo na qual eu habilito os slots e adiciono uma condição:



1. Abrir aba de **customização** de resposta;
2. Habilitar os slots;

The screenshot displays the IBM Watson Conversation Designer interface. On the left, the 'Pedir Pizza' dialog is shown with a flow: 'If bot recognizes: #pedir\_pizza' (3) leads to 'Then check for:' (4) with 'Check for: @sabor' (4), 'Save it as: \$sabor' (5), and 'If not present, ask: Qual sabor vc quer?' (6). The 'Type' is 'Required' (7). On the right, the 'Configure slot 1' panel shows 'Check for: @sabor' and 'Save it as: \$sabor'. It includes a 'Slot is required' indicator and a 'When user responds, if @sabor is...' section with 'Found:' (input field) and 'Not found:' (8) with the message 'Você não informou um sabor válido...'.

3. **Adicionar um slot** (uma nova variável);

4. Adicionar uma **nova condição**. Neste caso por exemplo, eu verifico se o usuário indicou o sabor de pizza com a entidade @sabor;

5. Criação de **variável de contexto**, que posteriormente pode ser manipulada. Toda variável de contexto no conversation é indicada por um \$;

6. Inserir o texto que o bot deve enviar se não reconhecer a variável (neste caso, ele pergunta "Qual sabor vc quer?");

7. Nesta engrenagem abriremos o painel de **configurações da variável**, mostrado na imagem à direita;

8. Neste painel vemos que adicionei uma informação no **"not found"**. Isto é, se o usuário continuar enviando alguma informação que não corresponda à condição indicada, o bot enviará outra mensagem. Como neste caso "você não informou um sabor válido". Neste mesmo painel, poderia ser adicionada uma informação no **"found"** no caso em que a variável fosse reconhecida.

Uma vez que esse fluxo foi detalhado, realizei o mesmo processo para todas as variáveis.

Pedir Pizza

Customize

×

If bot recognizes:

#pedir\_pizza

Then check for:

Manage handlers

	Check for	Save it as	If not present, ask	Type		
1	@sabor	\$sabor	Qual sabor vc quer?	Required	⚙	🗑
2	@tipo_massa	\$tipo_massa	Qual o tipo de massa?	Required	⚙	🗑
3	@informacao_entrega	\$informacao_entrega	Informe o CEP de entrega	Required	⚙	🗑
4	@sys-date	\$date	Informe a data de entrega	Required	⚙	🗑

+ Add slot

Para o CEP em específico, considerando que queremos que o bot guarde **literalmente** o que for escrito pelo usuário, vamos mudar as características da variável de contexto:

Configure slot 3

Check for:

@informacao\_entrega

Save it as:

\$informacao\_entrega

If \$informacao\_entrega is not present then ask:

Informe o CEP de entrega

When user responds, if @informacao\_entrega is...

Found:

Enter a response...

Not found:

1. Informe um cep válido

Add a variation to this response

Cancel

Save

9

Enable condition

Enable conditional responses

Open JSON editor

## Configure slot 3

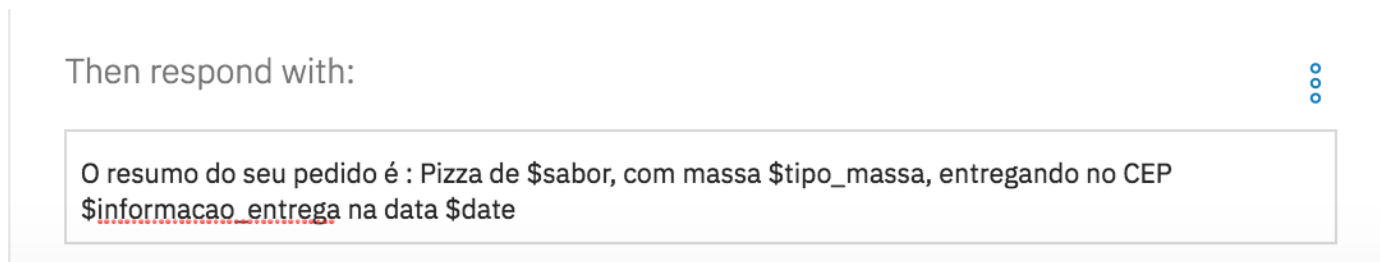
```

1 {
2   "context": {
3     "informacao_entrega": "@informacao_entrega.literal"
4   }
5 }

```

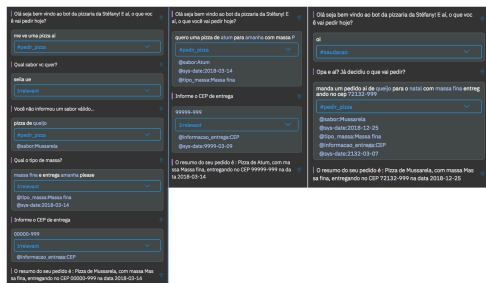
A variável de contexto tem a seguinte configuração: `informação_entrega.literal`

Por fim, queremos que o bot indique o resumo do pedido. Podemos manipular as variáveis para criar uma resposta:



Ué, não entendi essa inteligência aí?! Pois bem, utilizando os slots, se o usuário digitar “quero uma pizza”, o bot perguntará variável por variável como um atendente. Porém, suponhamos que o usuário digite “me vê uma pizza de queijo com massa fina para amanhã”, o bot só perguntará o CEP de entrega, pois as outras variáveis já foram reconhecidas.

## E então.... Vamos testar!



Nestes exemplos, fiz três tipos diferentes de interação com o bot e ele conseguiu reconhecer as variáveis de modo correto!

## The end!

Agora com os conceitos mais claros, é só popular o bot com mais conteúdo e me chamar para comer a pizza! (eu gosto de catupiry, hein?!)

Existem outras funcionalidades do conversation, como o Jump, que eu tratarei em próximos tutoriais.

E ah, se quiser acessar essa demo prontinha, veja o link abaixo:

[Assemble a pizza-ordering chatbot dialog - IBM Code](#) The new Watson Conversation Slots feature allows you to create a complex dialog with fewer nodes. Using slots in this...[developer.ibm.com](#)

No meu git tem o Código pra importação do Workspace também:

[amazon/demoChatbotBR](#) Contribute to demoChatbotBR development by creating an account on [GitHub.github.com](#)

Valeu!



## Sobre o autor

### Stéfany Mazon



Engineer, IBM developer advocate, TEDx Speaker. Passionate about innovation, new technologies, artificial intelligence and IoT!

© Copyright IBM Corporation 2018. Todos os direitos reservados.

([www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml))

[Marcas Registradas](#)

([www.ibm.com/developerworks/br/ibm/trademarks/](http://www.ibm.com/developerworks/br/ibm/trademarks/))