

AI & CHATBOT

Aula 04 – Watson Assistant e
Variáveis de Contexto e Slots

Prof. Erick

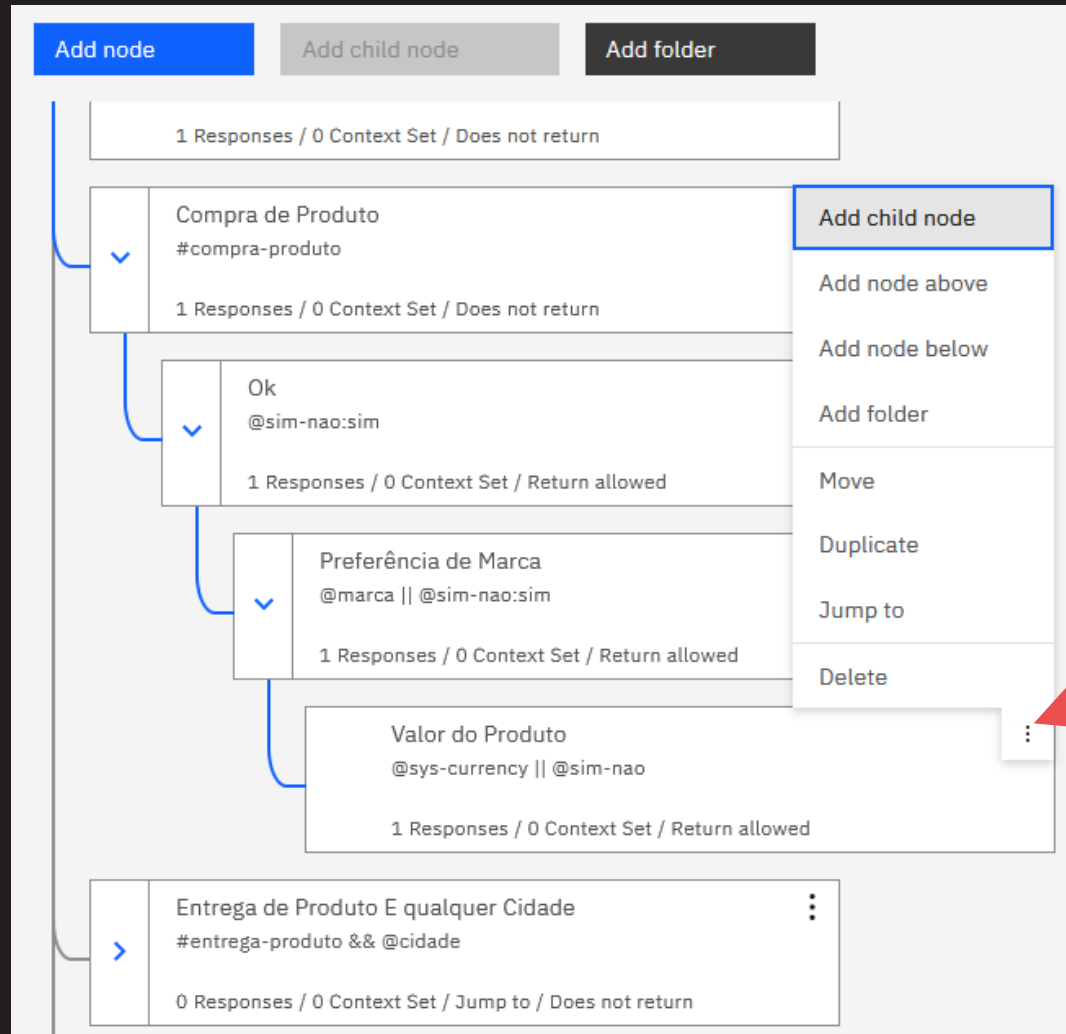
Slides Adaptados do Prof.
Henrique Ferreira.

FIAP
GRADUAÇÃO

Finalizando o diálogo

Criando um diálogo para vender produtos

Finalizando o diálogo de compra de produto



- Para finalizar o fluxo, vamos criar um nó confirmando que o produto foi adicionado ao carrinho.
- Adicionando o nó de aceite de compra;

Finalizando o diálogo de compra de produto

The screenshot displays the Microsoft Bot Framework Composer interface, divided into a flow editor on the left and a configuration pane on the right.

Flow Editor (Left):

- Buttons: "Add child node" and "Add for" are visible at the top.
- Flow Structure:
 - Node 1: "Ok para adicionar produto" (Node name will be shown to customers for disambiguation so use something descriptive. [Settings](#))
 - Node 2: "Preferência de Marca" (Parameters: @marca || @sim-nao:sim)
 - Node 3: "Valor do Produto" (Parameters: @sys-currency || @sim-nao)
 - Node 4: "Ok" (Parameters: @sim-nao:sim)
- Each node shows "1 Responses / 0 Context Set / Return allowed".

Configuration Pane (Right):

- Node Name:** "Ok para adicionar produto" (with a [Settings](#) link).
- Node Description:** "Node name will be shown to customers for disambiguation so use something descriptive."
- If assistant recognizes:** A list containing "@sim-nao:sim" with a trash icon and a plus sign.
- Assistant responds:**
 - Response type: "Text" (dropdown menu).
 - Response text: "O produto foi adicionado ao seu carrinho."
 - Response variation input: "Enter response variation".
 - Response variations are set to **sequential**. Set to [random](#) | [multiline](#).
 - [Learn more](#) link.

Finalizando o diálogo de compra de produto

- Na sequência, crie a intenção de agradecimento pelo atendimento.

Intent name

Name your intent to match a customer's question or goal

#agradecimento

Description (optional)

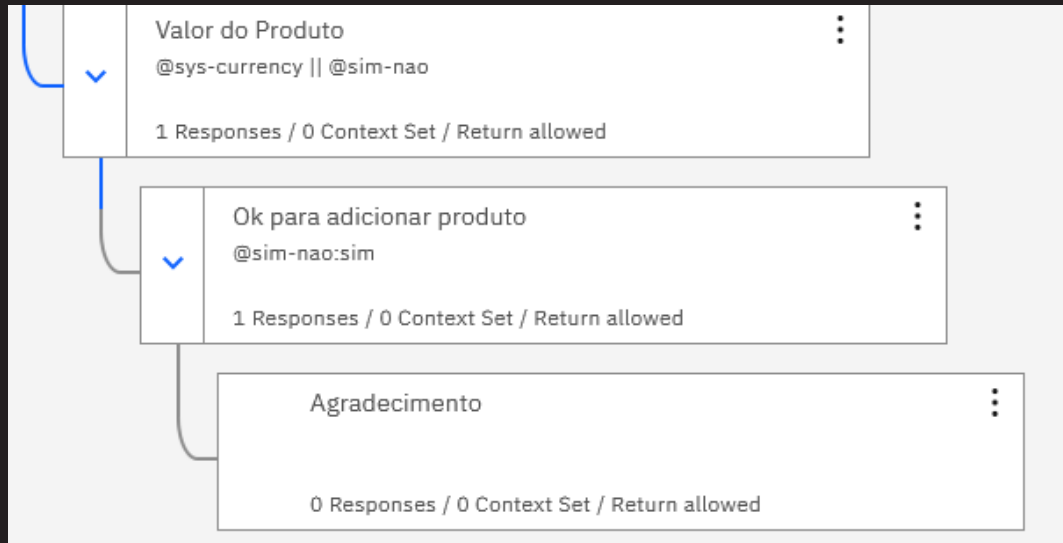
Intenção de agradecimento

Finalizando o diálogo de compra de produto

- E adicione os exemplos:

<input type="checkbox"/>	User examples (8) ↑
<input type="checkbox"/>	Legal, valeu!
<input type="checkbox"/>	Muito obrigado senhor bot
<input type="checkbox"/>	Obrigada você pela ajuda
<input type="checkbox"/>	Obrigado pela ajuda
<input type="checkbox"/>	Show robô obrigado
<input type="checkbox"/>	Tamo junto
<input type="checkbox"/>	Valeu bot, você me ajudou demais!
<input type="checkbox"/>	Você me ajudou, muito obrigado

Finalizando o diálogo de compra de produto

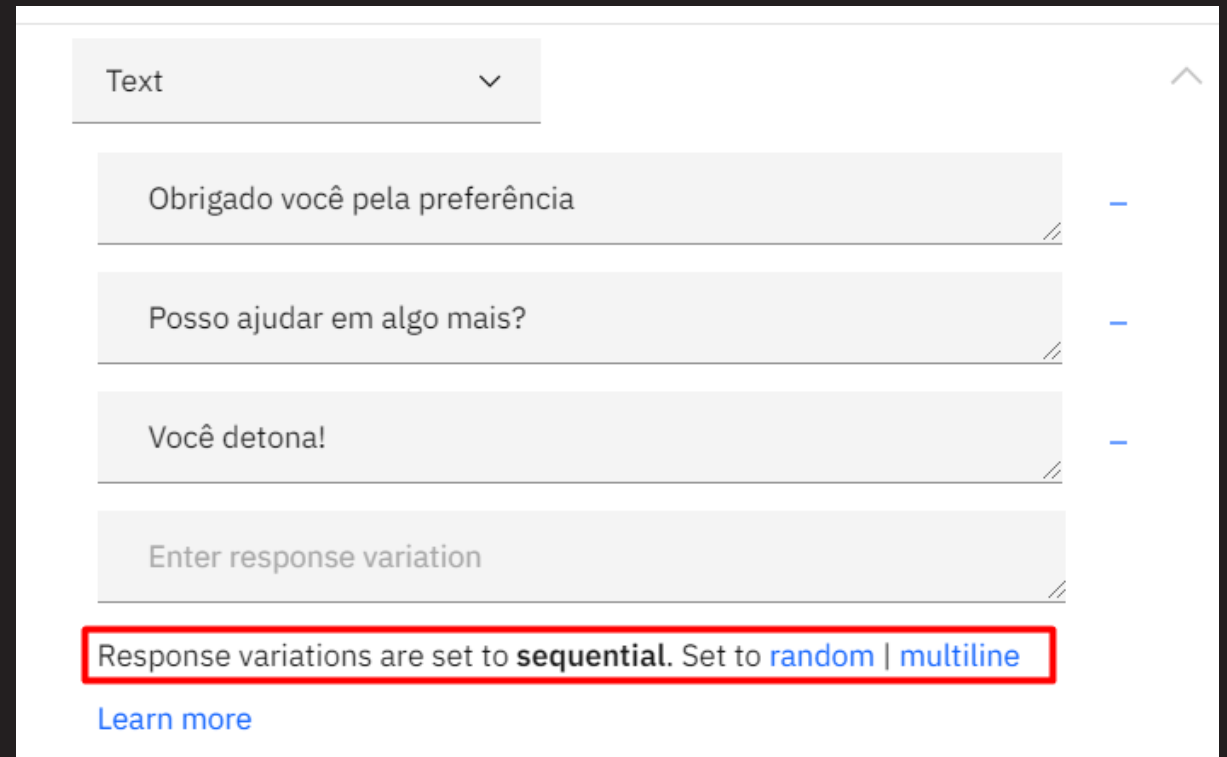


- Vamos adicionar agora o nó de conclusão;
- Para a resposta não ser sempre a mesma, nesse caso, podemos adicionar outras opções de resposta.

The screenshot shows the configuration for the 'Agradecimento' node. At the top, the node name 'Agradecimento' is displayed. Below it, a note states: 'Node name will be shown to customers for disambiguation so use something descriptive.' with a 'Settings' link. The 'If assistant recognizes' section contains a tag '#agradecimento' with minus and plus icons. The 'Assistant responds' section has a 'Text' dropdown menu. Below this, there is a text input field containing 'Obrigado você pela preferência', followed by a 'Enter response variation' field. At the bottom, a note says 'Response variations are set to sequential. Set to random | multiline' with a 'Learn more' link. A red arrow points from the text 'Podemos adicionar outras opções de resposta.' in the list to the 'Obrigado você pela preferência' text input field.

Finalizando o diálogo de compra de produto

- Veja alguns exemplos de outras possibilidades de resposta.
- Além disso, é possível definir como elas irão aparecer.
 - Sequencial
 - Aleatória
 - Múltiplas linhas (todas as repostas aparecem de uma vez)

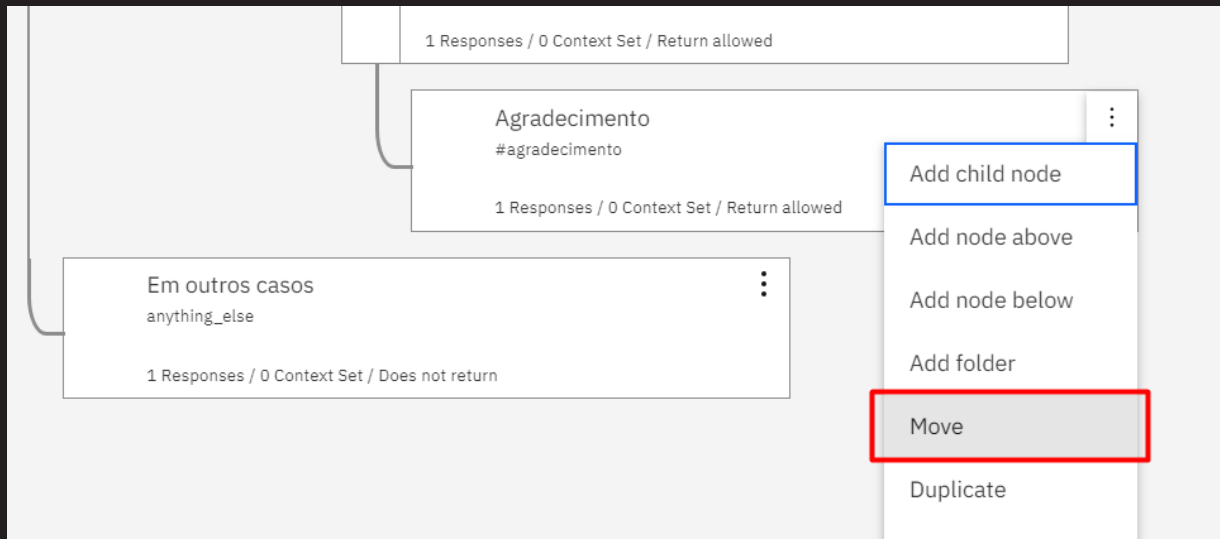


The screenshot shows a configuration interface for a chatbot. At the top, there is a dropdown menu labeled "Text" with a downward arrow. Below this, there are four text input fields, each with a blue minus sign on the right side. The first three fields contain the text: "Obrigado você pela preferência", "Posso ajudar em algo mais?", and "Você detona!". The fourth field is empty and contains the placeholder text "Enter response variation". Below these fields, there is a red-bordered box containing the text: "Response variations are set to **sequential**. Set to [random](#) | [multiline](#)". At the bottom of the interface, there is a blue link that says "Learn more".

Agradecendo a todos

Posicionando o nó de agradecimento para qualquer diálogo

Reposicionamento do nó Agradecimento



- O nó agradecimento, poderá ser utilizando nos outros fluxos da conversa.
- Portanto, vamos muda-lo para o fluxo raiz do nosso Assistant.
- Clique nos 3 pontos do nó Agradecimento e selecione Move.

Reposicionamento do nó Agradecimento

- Depois clique no nó Em outros casos e selecione Above Node para posicioná-lo acima. E pronto!



Lembrando do que foi dito

Armazenando informações em variáveis de contexto.



Criando uma variável de contexto dentro do diálogo

- Voltamos ao nó Preferência de Marca e na parte de resposta, clicamos nos 3 pontinhos e escolhemos Open JSON editor:

The screenshot displays the Dialogflow console interface. On the left, a flowchart shows the sequence of nodes: 'Compra de Produto', 'Ok', 'Preferência de Marca', 'Valor do Produto', and 'Entrega de Produto'. The 'Preferência de Marca' node is selected, and its configuration is shown on the right. The 'If assistant recognizes' section contains the conditions '@marca' or '@sim-nao:sim'. The 'Assistant responds' section shows a 'Text' response type with the message 'E você tem algum valor máximo para comprar esse produto?'. A red box highlights the three-dot menu icon in the top right corner of the response editor, with a red arrow pointing to it.

Criando uma variável de contexto dentro do diálogo


Preferência de Marca


Customize  


Node name will be shown to customers for disambiguation so use something descriptive.

Settings

If assistant recognizes


@marca 

or 

@sim-nao:sim 

+

Assistant responds



```
1 {
2   "output": {
3     "generic": [
4       {
5         "values": [
6           {
7             "text": "E você tem algum valor máximo para comprar esse
produto? "
8           }
9         ],
10        "response_type": "text",
11        "selection_policy": "sequential"
12      }
13    ]
14  }
15 }
```

- Deve aparecer algo assim;

- Isto daqui são dados armazenados no padrão JSON (JavaScript Object Notation).

Criando uma variável de contexto dentro do diálogo

- Vamos criar nossa variável de contexto e passar para ela o valor do @marca

Assistant responds

```
3     generic : [
4       {
5         "values": [
6           {
7             "text": "E você tem algum valor máximo para comprar
esse produto? "
8           }
9         ],
10        "response_type": "text",
11        "selection_policy": "sequential"
12      }
13    ]
14  },
15  "context": {
16    "marca_escolhida": "@marca.literal"
17  }
18 }
```

Show more

Usando a variável de contexto dentro do diálogo

The screenshot displays a chatbot configuration interface. On the left, a sidebar shows a list of nodes. The main area is titled 'Valor do Produto' and includes a 'Customize' button and a 'Settings' link. Below the title, a note states: 'Node name will be shown to customers for disambiguation so use something descriptive.' The configuration is divided into two sections: 'If assistant recognizes' and 'Assistant responds'.

In the 'If assistant recognizes' section, the conditions are set to '@sys-currency' or '@sim-nao'.

In the 'Assistant responds' section, the response type is 'Text'. The response text is: 'Eu tenho um celular da \$marca_escolhida que é até @sys-currency reais. Gostaria de adicionar ao carrinho?'. The variable '\$marca_escolhida' is highlighted with a red box.

Below the response text is a field labeled 'Enter response variation'.

The sidebar on the left shows a list of nodes, including 'Valor do Produto' which is currently selected. The node details for 'Valor do Produto' are: '@sys-currency || @sim-nao' and '1 Responses / 0 Context Set / Return allowed'.

Testando

Try it out Clear Manage Context 2

Oi, tudo bem? Posso te ajudar?

Quero ajuda para comprar um celular

#compra-produto

Legal vou te ajudar. Irei fazer algumas perguntas para selecionar o melhor produto para você, ok?

sim

Irrelevant

@sim-nao:sim

Você tem preferência por alguma marca de celular?

Try it out Clear Manage Context 3

sim, Samsung

Irrelevant

@sim-nao:sim
@marca:samsung

E você tem algum valor máximo para comprar esse produto?

até 1200 reais

Irrelevant

@sys-currency:1200

Eu tenho um celular da Samsung que é até 1200 reais. Gostaria de adicionar ao carrinho?

Context variables

\$ Enter variable name

\$marca_escolhida

"Samsung"

\$metadata

{"user_id":"10def365-d9b6-458d-a8ea-"

\$timezone

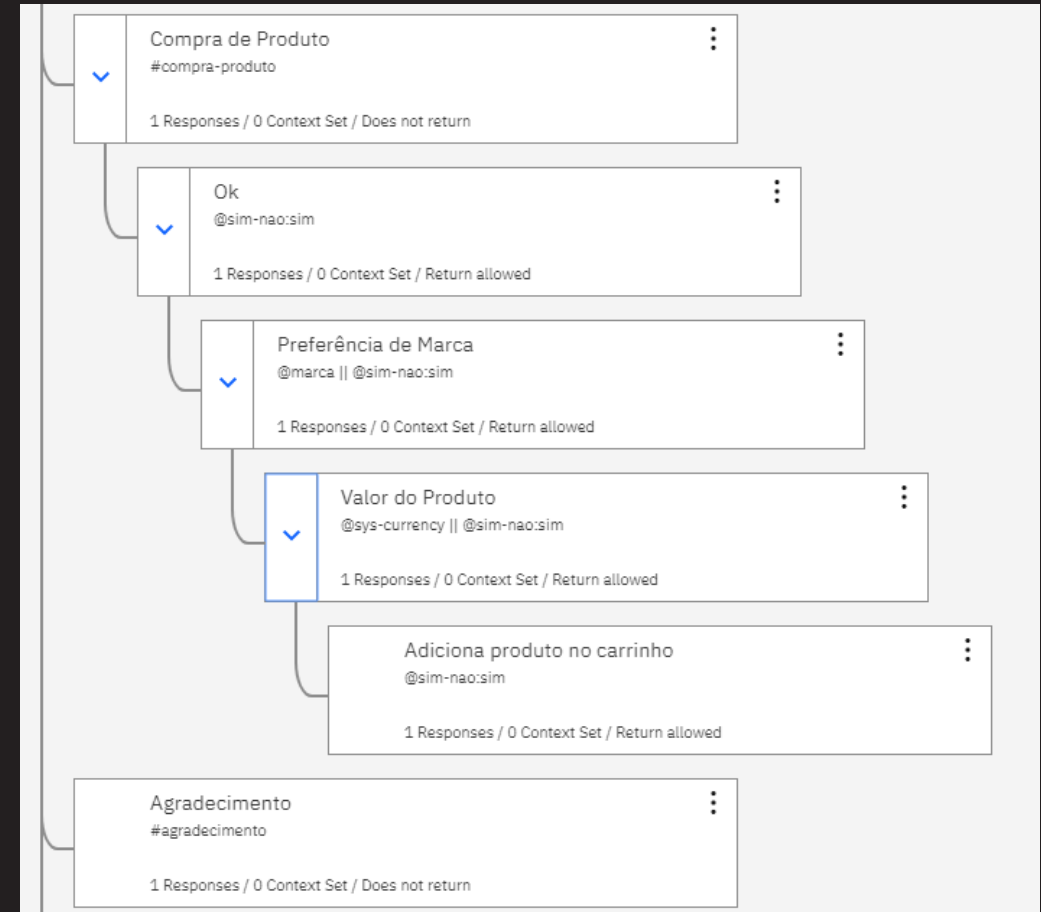
"America/Sao_Paulo"

Melhorando o diálogo

Transformando em um único nó.

Transformando em um único nó.

- Veja como ficou nossa árvore.
- Ela está grande demais, e isso pode causar problemas de desempenho, não só computacional, mas também na manutenção do fluxo da conversa.
- Vamos melhorar esse fluxo.

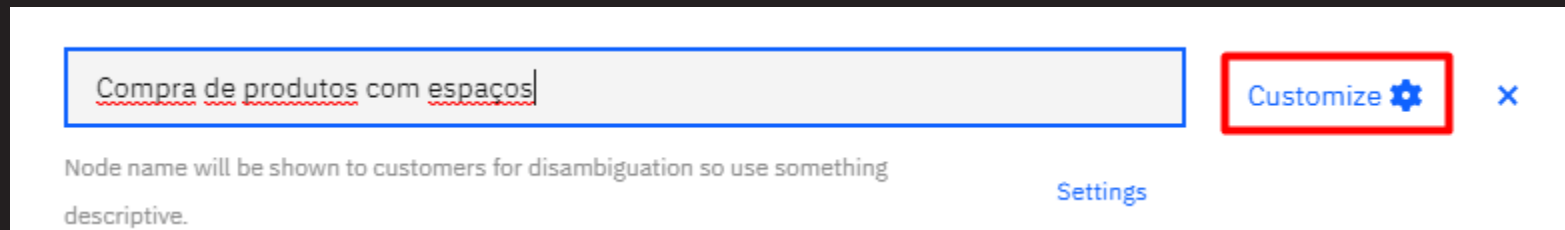


Transformando em um único nó.

- Um dos pontos que mais geram problemas em árvores grandes são as manutenções, assim como a reutilização tanto das informações, quanto dos nós.
- Dessa maneira, o Watson disponibiliza para nós alguns recursos do Assistant para nos auxiliar nessas tarefas.
- Vamos criar um novo nó chamado Compra de produto com espaços, para deixarmos os dois fluxos como comparação.

Transformando em um único nó.

- Antes de continuarmos a configuração que conhecemos, vamos selecionar a opção Customize, disponível logo ao lado do nome.



Compra de produtos com espaços

Node name will be shown to customers for disambiguation so use something descriptive.

Customize ⚙️


Settings

Transformando em um único nó.

- Habilite a opção de slots e clique em Apply

Customize "Compra de produtos com espaços"

Customize node Digressions

Slots ⓘ  On

Enable this to gather the information your bot needs to respond to a user within a single node.

☐ Prompt for everything

Enable this to ask for multiple pieces of information in a single prompt, so your user can provide them all at once and not be prompted for them one at a time.


Transformando em um único nó.

- Vamos preencher a intenção para reconhecimento e também a primeira condição de sim e não.

If assistant recognizes





#compra-produto - +

Then check for 0 Manage handlers

	Check for	Save it as	If not present, ask	Type	
1	@sim-nao	\$sn	Legal, vou te	Required	 -

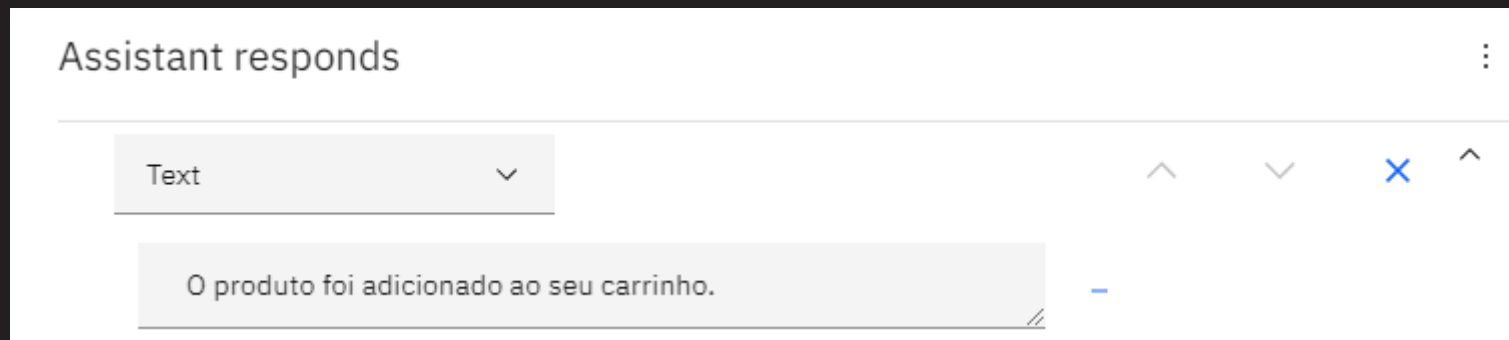
Transformando em um único nó

- Vamos continuar preenchendo os slots de acordo com o diálogo. Esse é o resultado:

	Check for	Save it as	If not present, ask	Type		
1	@sim-nao	\$sn	Legal, vou te	Required		—
2	@marca	\$marca	Você tem pr	Required		—
3	@sys-curren	\$currency	E você tem a	Required		—
^						
4	@sim-nao	\$sim_nao	Eu tenho um	Required		—


Transformando em um único nó

- No slot 4, aproveitamos e utilizamos as variáveis salvas \$marca e \$currency para utilizá-las na pergunta:
 - Eu tenho um celular da marca \$marca que é até \$currency reais. Gostaria de adicionar ao carrinho?
- Agora colocamos a resposta final:





Transformando em um único nó

- Hora de testar!
- Perceba que funcionou a marca e a moeda dentro da sequência de espaço.
- Pronto! Agora nosso chatbot ficou mais leve e de fácil manutenção =D


E você tem algum valor máximo para comprar esse produto? 

1000 reais

#agradecimento  

@sys-currency:1000

@sys-currency:1000

Eu tenho um celular da marca xiaomi que é até 1000 reais. Gostaria de adicionar ao carrinho? 

Capturando varias informações

Expressões Regulares, Slots e Variáveis de Contexto.

Vamos criar um nó de cadastro para pegar essas informações: precisamos de intenções e entidades

← | #cadastro

Captura a intenção do usuário de fazer um cadastro na loja/site

User example

Type a user example here

Add unique examples of what the user might say. (Pro tip: Add at least 5 unique examples to help Watson understand)

Add example

☐ User examples (7) ↑

- ☐ Ai bot, me ajuda nessa parte de cadastro
- ☐ Bot, você faz o meu cadastro?
- ☐ Como eu faço o cadastro?
- ☐ Preciso fazer cadastro, me ajuda?
- ☐ Quero ajuda para me cadastrar
- ☐ Quero fazer o meu cadastro na loja
- ☐ Quero me cadastrar no site

- Primeiramente temos que criar uma intenção para iniciar o cadastro. Adicionem mais exemplos.

Criando as variáveis que serão capturadas

- Agora, vamos criar as entidades cep, email e telefone para incluir os padrões a serem reconhecidos. Vamos criar cada uma delas separadamente.
- Vamos começar com o cep, basta digitar cep, desabilitar o “Fuzzy matching” e clicar em criar.

IBM Watson Assistant Lite Upgrade

← | Create new entity

Entity name
Name your entity to match the category of values that it will detect.

@cep

Create entity

Fuzzy matching ⓘ
☐ Off

Usando Expressões Regulares para capturar padrões

- Temos que selecionar “Patterns” e incluir o padrão para extrair o cep. Vamos supor que o cep tem dois padrões: 00000-000 ou 00000 000.
- Para isso vamos inserir o comando `\d` que procura por qualquer número e dentro dos parênteses a quantia de números, ou seja, 5 e depois 3 números.
- Isso implica que temos `\d{5}` que procurará por cinco número em sequência e `\d{3}` que procurará por três números em sequência.

The screenshot shows a web interface for defining a pattern for the entity '@cep'. The interface has a header with a back arrow and the entity name '@cep'. Below the header, there is a section for 'Entity name' with a text input field containing '@cep'. To the right of this section is a 'Fuzzy matching' toggle switch, which is currently turned off. Below the 'Entity name' section, there is a 'Value' section with a text input field containing 'cep'. To the right of the 'Value' section is a 'Patterns' section. The 'Patterns' section contains a dropdown menu with 'Patterns' selected, followed by a text input field containing the regular expression `(\d{5}-\d{3})`, and another text input field containing the regular expression `(\d{5} \d{3})`. The first two input fields are highlighted with red boxes. At the bottom left of the interface is a blue button labeled 'Add value'.

Usando Expressões Regulares para capturar padrões

- O resultado final do cep deveria ser como a imagem abaixo:

The screenshot shows a web interface for defining an entity named '@cep'. The 'Entity name' field contains '@cep'. The 'Fuzzy matching' toggle is turned off. The 'Value' field contains 'Type a value, e.g. Checking'. The 'Synonyms' dropdown is set to 'Synonyms'. The 'Synonyms' field contains 'Type a synonym, e.g. Deposit'. Below the input fields is an 'Add value' button. At the bottom, a table displays the defined values and their patterns.

Values (1) ↑	Type
<input type="checkbox"/> cep	Patterns (\d{5}-\d{3}), (\d{5} \d{3})

Usando Expressões Regulares para capturar padrões

- Criamos a entidade telefone, com os valores fixo e celular e com os padrões demonstrados abaixo:

← | @telefone

Last updated: há poucos segundos

Try

Entity name

@telefone

Fuzzy matching ?

Off

Name your entity to match the category of values that it will detect.

Value

Type a value

Synonyms

Type a synonym

+

Add value

☐ Values (2) ↓

Type

☐ fixo

Patterns

`(\d{2}) (\d{4}-\d{4}), (\d{2}) (\d{8})`

☐ celular

Synonyms

`(\d2) (\d{5}-\d{4})`

-

`(\d2) (\d{9})`

-

Usando Expressões Regulares para capturar padrões

- Criamos a entidade email, com o valor email e com o padrão demonstrado abaixo:

← | @email

Entity name
Name your entity to match the category of values that it will detect.

@email

Fuzzy matching ⓘ
☐ Off

Value

Type a value, e.g. Checking

Synonyms

Synonyms ▼

Type a synonym, e.g. Deposit

+

Add value

$$([a-zA-Z0-9+._-])+@[a-zA-Z0-9+._-]+\.[a-zA-Z0-9_-]+)$$
$$\backslashb[A-Za-z0-9._%+~]+@([A-Za-z0-9-]+\.)+[A-Za-z]{2,}\backslashb$$

<input type="checkbox"/> Values (1) ↑	Type
<input type="checkbox"/> email	Patterns $\backslashb[A-Za-z0-9._%+~]+@([A-Za-z0-9-]+\.)+[A-Za-z]{2,}\backslashb$

Vamos usar Slot para criar o nó de Cadastro

- Basta criar agora o nó de diálogo, inserir a intenção #cadastro e ativar o Slot:

Enter node name (optional) Customize

Node name will be shown to customers for disambiguation so use something descriptive. [Settings](#)

If assistant recognizes

#cadastro

Then check for [Manage handlers](#)

	Check for	Save it as	If not present, ask	Type
1	Enter condition	Enter variable	Enter prompt	Optional Settings

[Add slot](#)

Assistant responds

Text

Enter response text

Response variations are set to **sequential**. Set to [random](#) | [multiline](#)

[Learn more](#)



Customize "Untitled" ×

[Customize node](#) [Digressions](#)

Slots ? On

Enable this to gather the information your bot needs to respond to a user within a single node.

☐ Prompt for everything

Enable this to ask for multiple pieces of information in a single prompt, so your user can provide them all at once and not be prompted for them one at a time.

Webhooks Off

Enable this setting to send a POST request from this dialog node to the webhook URL. The URL and headers are defined in the Webhooks settings of the Options tab. After you enable this setting, the Multiple conditional responses setting is enabled automatically to support adding a response to show when the request is successful and another response to show if the request fails. [Learn more](#)

Multiple conditioned responses ? Off

Cancel Apply

34/51




Criando o Slot para o nó de Cadastro

- Basta criar adicionar as entidades para identificar cada uma delas e inserir um nome para variável de contexto, como segue :

If assistant recognizes

#cadastro - +

Then check for 0 Manage handlers

	Check for	Save it as	If not present, ask	Type	
1	@cep	\$cep	Qual o cep?	Required	<div> -</div>
2	@email	\$email	Qual o e-mail?	Required	<div> -</div>
3	@telefone	\$telefone	Qual o telefone?	Required	<div> -</div>

Criando o Slot para o nó de Cadastro

- Basta configurar cada slot individualmente:

Configure slot 1

Check for: @cep

Save it as: \$cep

If slot context variable is not present ask: Slot is required ⓘ

Text

Qual o cep?

Enter response variation

Response variations are set to **sequential**. Set to [random](#)

[Learn more](#)

Cancel Save

Slots, Variáveis de Contexto e Expressões Regulares

- Basta inserir `.literal` depois da referência da entidade para garantir que o valor extraído seja reconhecido e não a referência.

Configure slot 1

```
1 {
2   "context": {
3     "cep": "@cep"
4   }
5 }
```

If slot context variable is not present ask: Slot is required ⓘ

Text

Cancel Save



Configure slot 1

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

If slot context variable is not present ask: Slot is required ⓘ

Text

Cancel Save 37/51

Slots, Variáveis de Contexto e Expressões Regulares

Basta repetir o processo para as outras entidades: email e telefone

Configure slot 2

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

If slot context variable is not present ask: Slot is required ⓘ

Text

Cancel Save

Configure slot 3

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

If slot context variable is not present ask: Slot is required ⓘ



Text

Cancel Save

Slots, Variáveis de Contexto e Expressões Regulares


- Vamos adicionar um pergunta de cabeçalho para saber que estamos iniciando o cadastro:

Cadastro

Customize  







Node name will be shown to customers for disambiguation so use something descriptive. [Settings](#)


If assistant recognizes

#cadastro  +

Then check for

0 Manage handlers

	Check for	Save it as	If not present, ask	Type		
1	@cep	\$cep	Qual o cep?	Required		
2	@email	\$email	Qual o e-mail?	Required		
3	@telefone	\$telefone	Qual o telefon	Required		

Manage handlers for "Cadastro" 

Handlers are how your bot will respond when the users answer to a prompt is not found. These handlers will be checked before trying the "Not found" responses in a slot.

If answer to any prompt is not found and

Add handler +

Cancel Save

Slots, Variáveis de Contexto e Expressões Regulares

- Vamos adicionar um pergunta de cabeçalho para saber que estamos iniciando o cadastro:

Manage handlers for "Cadastro" ×

Handlers are how your bot will respond when the users answer to a prompt is not found. These handlers will be checked before trying the "Not found" responses in a slot.

If answer to any prompt is not found and

	If assistant recognizes	Respond with	
1	true	Ok, vou realizar seu cadastro.	⚙️ 🗑️

[Add handler +](#)

Cancel Save

Entrega de Produto
#entrega-produto

Slots, Variáveis de Contexto e Expressões Regulares

Testando:

Oi, tudo bem? Posso te ajudar?

quero realizar o cadastro

#cadastro

Ok, vou realizar seu cadastro

Você poderia me informar o seu CEP?

Se, contudo, errarmos o cep, o que acontecerá?



Oi, tudo bem? Posso te ajudar?

quero realizar o cadastro

#cadastro

Ok, vou realizar seu cadastro

Você poderia me informar o seu CEP?

04041004

Irrelevant

Ok, vou realizar seu cadastro

Você poderia me informar o seu CEP?

Slots, Variáveis de Contexto e Expressões Regulares

- Uma forma de evitar isso é fazer o handler reconhecer a intenção de cadastro:

Manage handlers for "Cadastro" ×

Handlers are how your bot will respond when the users answer to a prompt is not found. These handlers will be checked before trying the "Not found" responses in a slot.

If answer to any prompt is not found and

	If assistant recognizes	Respond with		
1	#cadastro	Ok, vou realizar seu cadastro	⚙️	🗑️

[Add handler +](#)

Cancel Save

Testando:

Olá. Como posso te ajudar? 📍

quero realizar o cadastro

#cadastro × ⌵ 👁️

Ok, vou realizar seu cadastro 📍

Você poderia me informar o seu CEP?

04041004

Irrelevant × ⌵ 👁️

Você poderia me informar o seu CEP? 📍

Slots, Variáveis de Contexto e Expressões Regulares

- Por fim, vamos terminar de configurar o nó de cadastro perguntando se o usuário confirma os dados:

Cadastro

Customize

Node name will be shown to customers for disambiguation so use something descriptive. [Settings](#)

[Add slot +](#)

Assistant responds

Text

Poderia confirmar que o seu cep é \$cep, o e-mail é \$email e o seu telefone é \$telefone

Enter response variation

Response variations are set to **sequential**. Set to [random](#) | [multiline](#)

[Learn more](#)

43/51

Testando

Try it out

Clear

Manage Context

3

×

Oi, tudo bem? Posso te ajudar?

quero realizar o cadastro na loja

#cadastro

Ok, vou realizar seu cadastro.

Qual o cep?

04023-018

Irrelevant

@cep:cep

Qual o e-mail?

Try it out

Clear

Manage Context

5

⋮

Qual o e-mail?

professor@teste.com

Irrelevant

@email:email

Qual o telefone?

11 55382650

Irrelevant

@telefone:fixo

@cep:cep

Poderia confirmar que o seu cep é 04023-018, o e-mail é professor@teste.com e o seu telefone é 11 55382650

Descanso

Do Professor =D

Exercícios

1. Faça o planejamento dos slots com a entidade de produto.
2. Termine o fluxo de cadastro pensando que o usuário pode responder sim ou não a confirmação dos dados. Em caso negativo, como você poderia lidar com a situação?
3. Crie um fluxo para capturar o nome do usuário e fazer o bot responder usando este nome. Dicas: você precisará criar uma variável de contexto e escolher entre criar uma entidade ou usar toda a entrada. Tente passar o valor `<? input.text ?>` como parâmetro para o contexto.

Estudo Complementar

Dicas, links e livros

Estudo Complementar

1. Documentação do Watson Assistant:
<https://cloud.ibm.com/docs/assistant?topic=assistant-dialog-runtime-context>
<https://cloud.ibm.com/docs/assistant?topic=assistant-dialog-methods>
<https://cloud.ibm.com/docs/assistant?topic=assistant-expression-language#expression-language-shorthand-context>
1. Exemplos de manipulação de variável de contexto:
<https://www.ibm.com/cloud/blog/enhance-chatbot-conversation-context-variables-system-entities>
2. Mais sobre JSON: <https://www.json.org/json-en.html>
3. Mais sobre Expressões Regulares:
https://en.wikipedia.org/wiki/Regular_expression

Próximos Passos

O que veremos na próxima aula

Na próxima aula...

- Programação com Node RED
- Integrando o Watson Assistant para utilizar outros serviços em nuvem

Copyright © 2022

Slides criados por Henrique Ferreira e Miguel Bozer, adaptados do material do Prof. Andrey Masiero e Marcelo Grave - FIAP

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).