



Collaborate. Innovate. Transform.

# Comunicação de Sinistros

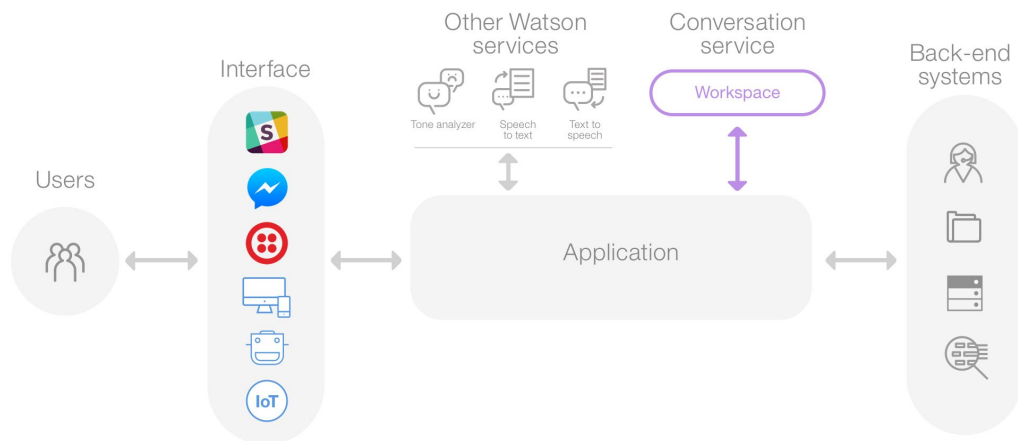
Com IBM Watson Conversation

Rodrigo Sclosa  
Março 2017

# O que é o Watson Conversation?

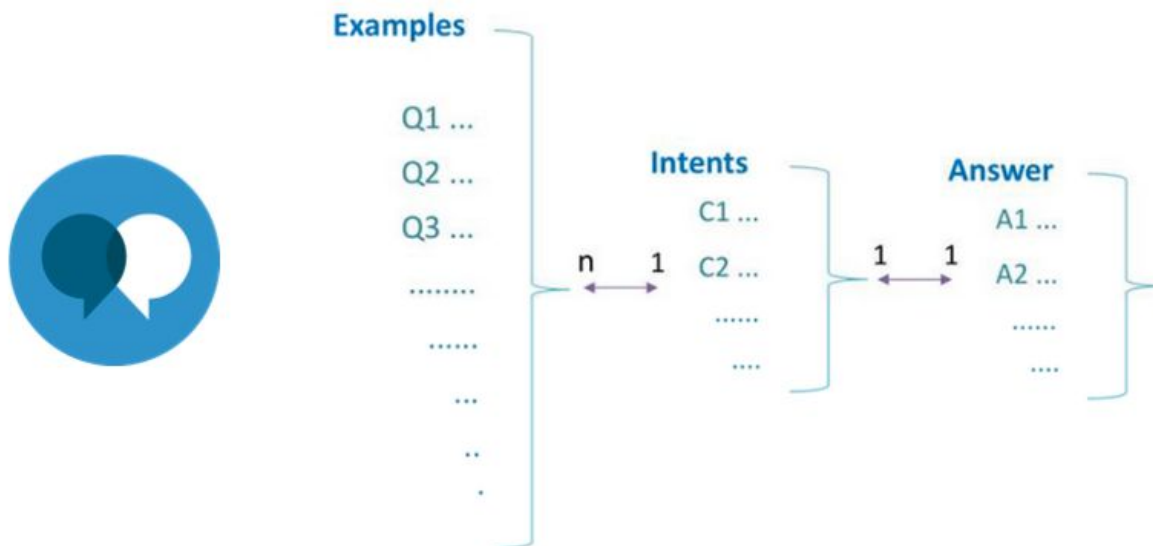
API capaz de entender um input em linguagem natural, através de técnicas de aprendizagem de máquina, e responder ao usuário de uma maneira que simule uma conversa entre humanos através do componente de Diálogo.

Principalmente utilizado para criar Assistentes Virtuais (Bots) e possui logs para verificar a qualidade do Assistente Virtual, possibilitando correções e melhorias.



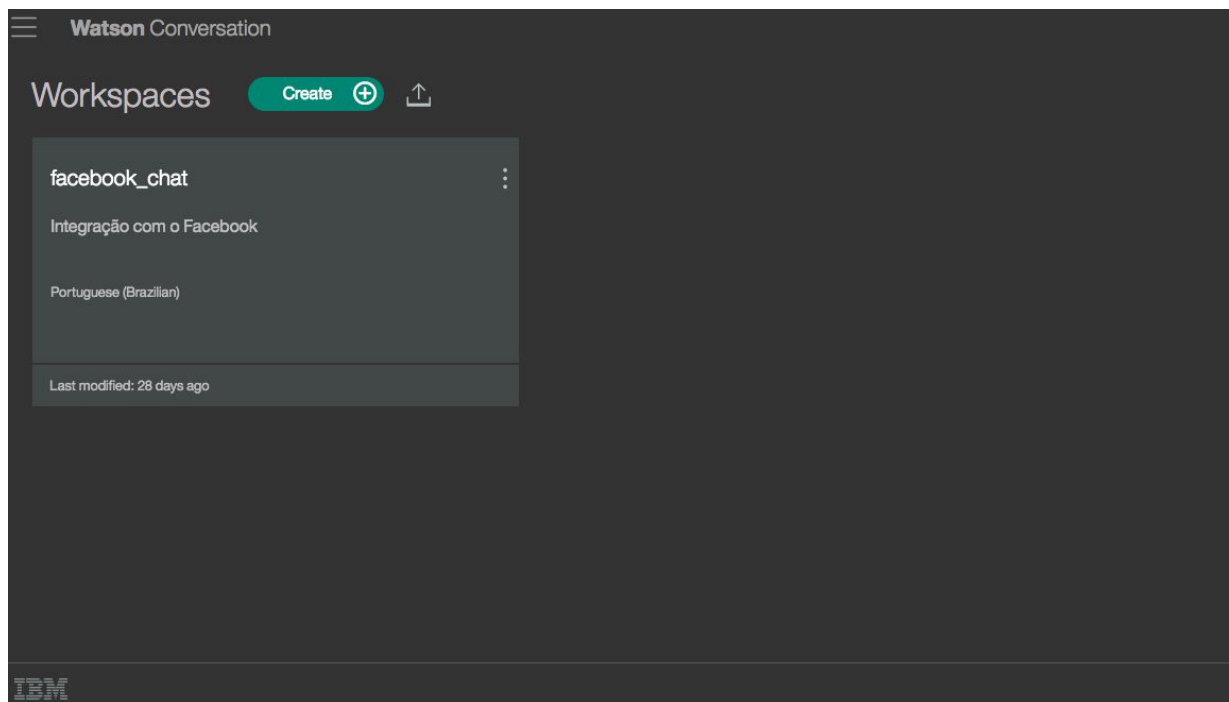
# O que é o Watson Conversation?

Relação entre Exemplos x Intenções x Respostas



# O que é o Watson Conversation?

Ferramenta de criação/edição: <https://www.ibmwatsonconversation.com>

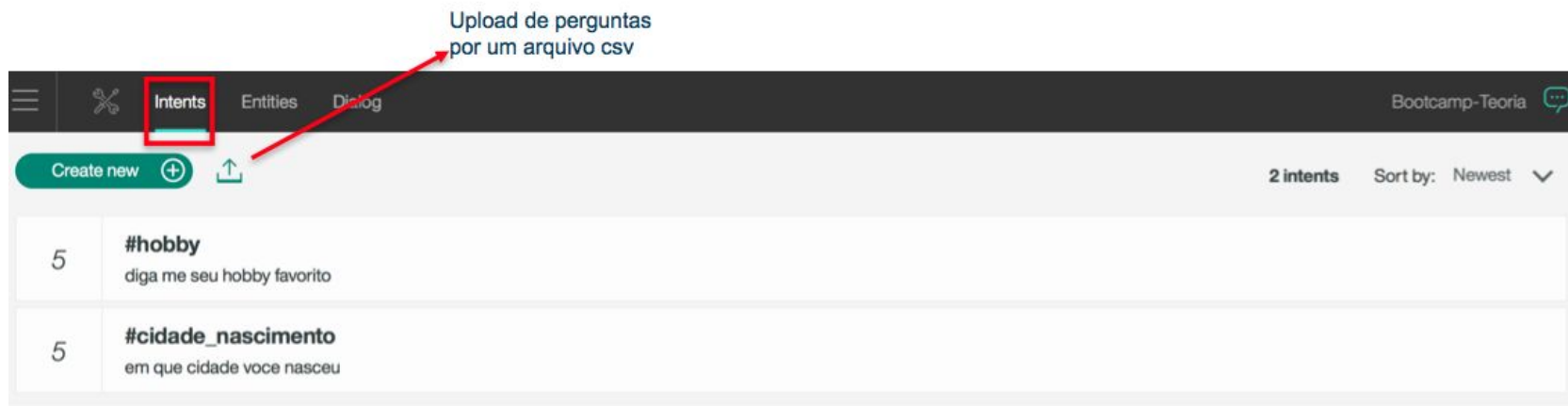


# O que é o Watson Conversation?

Composto por 4 componentes principais:

## 1. Intenções:

- Contém todas perguntas e suas respectivas variações agrupadas de acordo com a intenção.

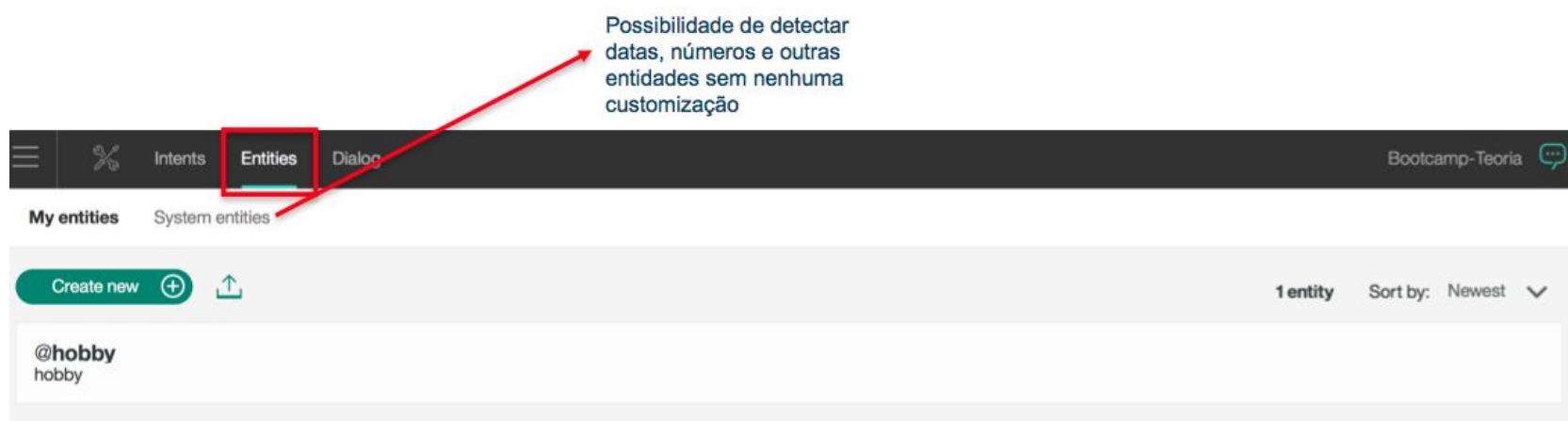


# O que é o Watson Conversation?

Composto por 4 componentes principais:

## 2. Entidades:

- Responsável por detectar entidades (fragmentos de texto, dados relevantes) e suas variações em cada mensagem recebida.

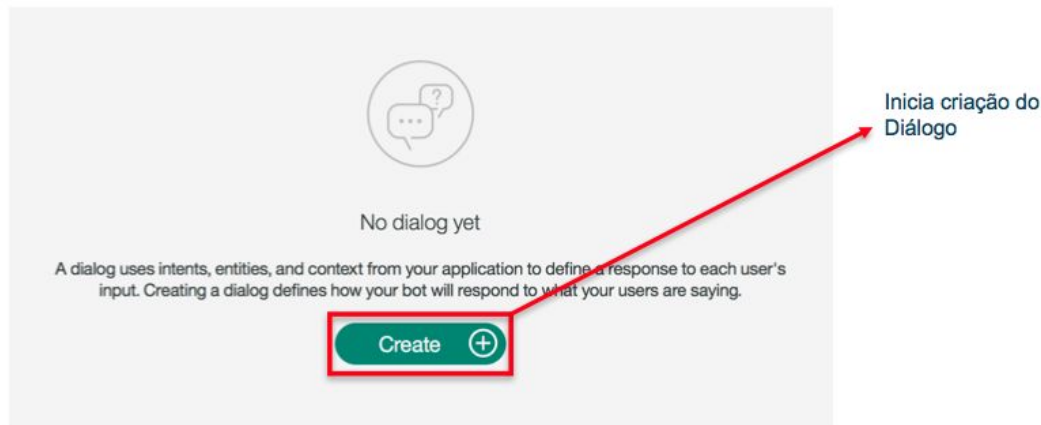


# O que é o Watson Conversation?

Composto por 4 componentes principais:

## 3. Diálogo:

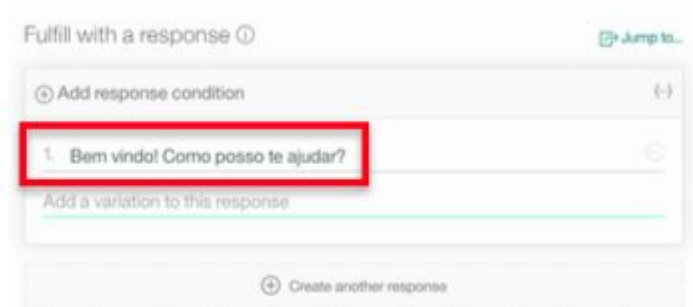
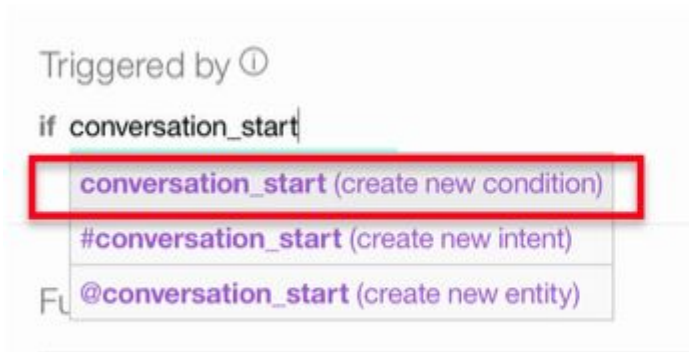
- Responsável por todas mensagens enviadas ao usuário, desde boas vindas, respostas e perguntas de desambiguação.



# O que é o Watson Conversation?

## 3. Diálogo:

- Esse nó (**conversation\_start**) é responsável por enviar a primeira mensagem ao usuário acessar o Bot.

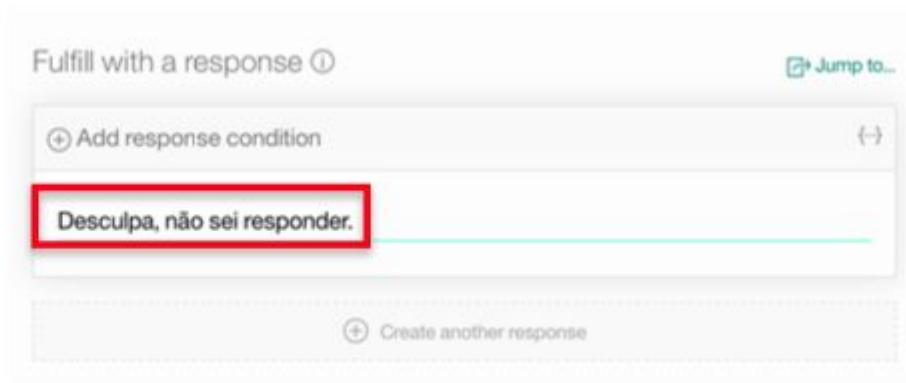
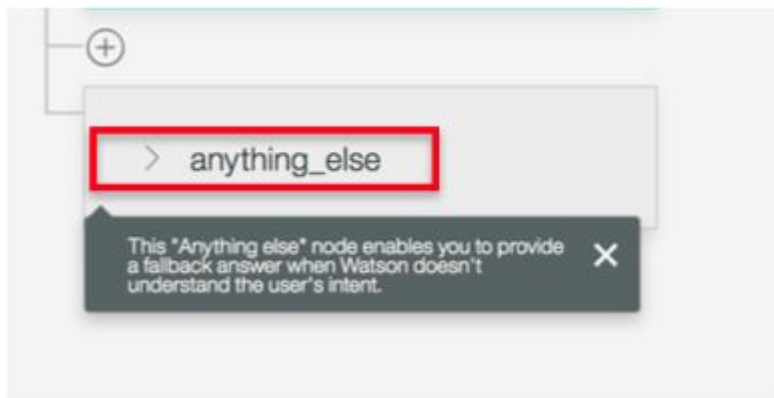




# O que é o Watson Conversation?

## 3. Diálogo:

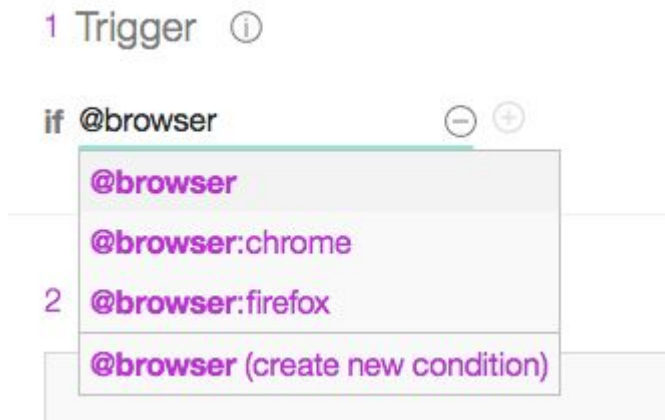
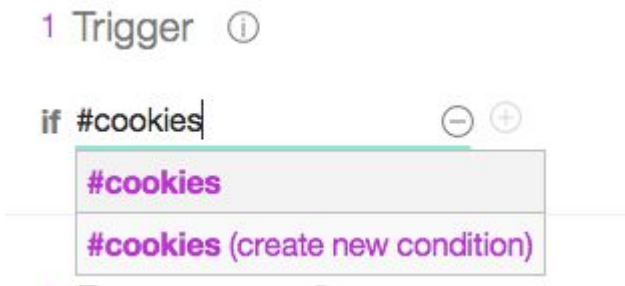
- Automaticamente, o nó **Anything else** é criado e utilizado para quando nenhuma condição for verdadeira dado uma certa pergunta do usuário.



# O que é o Watson Conversation?

## 3. Diálogo:

- Para criar um nó cuja regra é baseada em uma **intenção**, selecione a opção com o símbolo **#** na frente. Agora se a regra for baseada em **entidades**, selecione a opção com @ na frente.



# O que é o Watson Conversation?

## 3. Diálogo:

- As respostas para cada nó devem ser adicionadas ao campo **Responses**.

2 Responses ⓘ [Jump to...](#)

⊕ Add response condition {...}

1. Qual navegador você tem? Posso falar de Chrome ou Firefox. ⓘ

Add a variation to this response

# O que é o Watson Conversation?

## 3. Diálogo:

- Variáveis de **contexto** podem ser adicionadas utilizando a edição Avançada da resposta e também utilizada como condições em nós.

### 2 Responses ⓘ

[Jump to...](#)

```
{
  "context": {
    "browser": "Chrome"
  },
  "output": {}
}
```

### 1 Trigger ⓘ

if \$browser == "Chrome" [-](#) [+](#)

### 2 Responses ⓘ

[Jump to...](#)

[+](#) Add response condition [{...}](#)

1. Na Barra de Ferramentas, Clique em Mais -> Mais Ferramentas [-](#)

Add a variation to this response

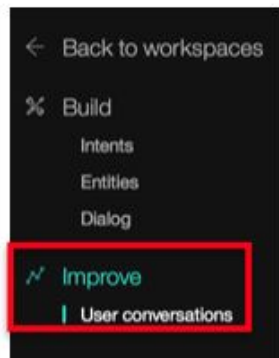
2. Seu browser é: <? context.browser ?> [-](#)

# O que é o Watson Conversation?

Composto por 4 componentes principais:

## 4. Melhorias:

- Responsável por guardar os logs das conversas com o serviço e analisar quais intenções podem ser melhoradas.



# O que é o Watson Conversation?

API Rest para integração:

- <https://gateway.watsonplatform.net/conversation/api/v1>

Documentação:

- <https://www.ibm.com/watson/developercloud/doc/conversation/index.html>

SDK Oficial (Node.js, Python e Java):

- <https://www.ibm.com/watson/developercloud/conversation/api/v1/>

App de Exemplo em Node.js:

- <https://github.com/watson-developer-cloud/conversation-simple>

# O que é o Watson Conversation?

## API Explorer:

- <https://watson-api-explorer.mybluemix.net/apis/conversation-v1>

 **Watson API Explorer**

### Conversation

The IBM Watson™ Conversation service combines machine learning, natural language understanding, and integrated dialog tools to create conversation flows between your apps and your users.

For more information about this service, see the documentation:

<http://www.ibm.com/watson/developercloud/doc/conversation>

#### workspaces

Show/Hide | List Operations | Expand Operations

GET	/v1/workspaces	List workspaces
POST	/v1/workspaces	Create workspace
DELETE	/v1/workspaces/{workspace_id}	Delete workspace
GET	/v1/workspaces/{workspace_id}	Get information about a workspace
POST	/v1/workspaces/{workspace_id}	Update workspace

#### message

Show/Hide | List Operations | Expand Operations

POST	/v1/workspaces/{workspace_id}/message	Get a response to a user's input
------	---------------------------------------	----------------------------------

#### counterexamples

Show/Hide | List Operations | Expand Operations

GET	/v1/workspaces/{workspace_id}/counterexamples	Get counterexamples
-----	---	---------------------

POST	/v1/workspaces/{workspace_id}/counterexamples	Create counterexample
DELETE	/v1/workspaces/{workspace_id}/counterexamples/{text}	Delete counterexample
GET	/v1/workspaces/{workspace_id}/counterexamples/{text}	Get counterexample
POST	/v1/workspaces/{workspace_id}/counterexamples/{text}	Update counterexample

#### intents

Show/Hide | List Operations | Expand Operations

GET	/v1/workspaces/{workspace_id}/intents	List intents
POST	/v1/workspaces/{workspace_id}/intents	Create intent
DELETE	/v1/workspaces/{workspace_id}/intents/{intent}	Delete intent
GET	/v1/workspaces/{workspace_id}/intents/{intent}	Get intent
POST	/v1/workspaces/{workspace_id}/intents/{intent}	Update intent

#### examples

Show/Hide | List Operations | Expand Operations

GET	/v1/workspaces/{workspace_id}/intents/{intent}/examples	Get user input examples
POST	/v1/workspaces/{workspace_id}/intents/{intent}/examples	Create user input example
DELETE	/v1/workspaces/{workspace_id}/intents/{intent}/examples/{text}	Delete user input example
GET	/v1/workspaces/{workspace_id}/intents/{intent}/examples/{text}	Get user input example
POST	/v1/workspaces/{workspace_id}/intents/{intent}/examples/{text}	Update user input example

[ BASE URL : /conversation/api . API VERSION : 1.0 ]

# DICAS AVANÇADAS



# Dicas avançadas - Watson Conversation

- Variáveis globais:

Variável	Definição
<i>intents[ ]</i>	Lista de intenções encontrada para o input do usuário
<i>entities[ ]</i>	Lista de entidades encontrada para o input do usuário
<i>input</i>	JSON object que guarda o input do usuário
<i>output</i>	JSON object que será enviado para a interface (Usualmente contem as respostas)
<i>context</i>	JSON object que mantém o estado da conversa
<i>conversation_start</i>	Variável "true" durante a primeira interação do usuário
<i>anything_else</i>	Último nó do Diálogo e utilizado quando nenhum outro nó foi encontrado como resposta para o usuário

# Dicas avançadas - Watson Conversation

## ATENÇÃO!

- O Conversation é **stateless**, ou seja, você é responsável por manter o estado da conversa armazenando o objeto **context** e enviando de volta em cada interação com a API. Você pode incluir variáveis e objetos próprios, mas não modifique os objetos do Conversation!

```
    "context": {  
      "conversation_id": "ed8e90f4-be95-4aea-968e-7233982ee48f",  
      "system": {  
        "dialog_stack": [  
          {  
            "dialog_node": "node_5_1490203216504"  
          }  
        ],  
        "dialog_turn_counter": 3,  
        "dialog_request_counter": 3,  
        "_node_output_map": {  
          "node_3_1487877118514": [  
            0,  
            2,  
            1,  
            0  
          ],  
          "node_5_1490124766209": [  
            0  
          ],  
          "node_5_1490203216504": [  
            0  
          ]  
        }  
      },  
      "acao": "vidro quebrado"  
    }  
  }
```

# Dicas avançadas - Watson Conversation

## Usando SpEL (Spring Expression Language)

- As funcionalidades do WCS podem ser melhoradas utilizando SpEL. Permite consultar e manipular objetos em tempo de execução.
- **Algumas funcionalidades:**
  - Literal/boolean/expressões relacionais
  - Expressão regular
  - Acesso a propriedades, arrays, listas, mapas
  - Operadores relacionais e manipulação de métodos
  - Manipulação de Array
  - Operador ternário (Elvis operator)
- **Referência:** <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/expressions.html>

# Dicas avançadas - Watson Conversation

## Abreviações:

Abreviação	Sintaxe SpEL completa	Objetivo
#intent_name	intent == 'intent_name'	Intent
! #intent_name	intent != 'intent_name'	
NOT #intent_name	intent != 'intent_name'	
#intent_a or #intent_b	(intent == 'intent_a'    intent == 'intent_b')	
@entity_name	entities['entity_name']?.value	Entity
@entity_name == entity_value	entities['entity_name']?.value == entity_value	
@entity_name != entity_value	entities['entity_name']?.value != entity_value	
@entity_name:entity_value	entities['entity_name']?.contains('entity_value')	
\$context_var_name:context_var_value	context['context_variable_name'] == 'context_var_value'	Context

# Dicas avançadas - Watson Conversation

## SpEL, como usar?

- Condições dos nós são SpEL!
- Também possível de usar nos output e nas variáveis de contexto:
  - **<? Expressão SpEL ?>**
- Por exemplo, como acessar a confiança das intenções retornadas?
  - As intenções preditas estão no array **intents[]**
  - Estão ordenadas da intenção de maior confiança para a menor
  - Cada elemento tem duas propriedades: intent & confidence:
    - A confiança da intenção **<? intents[0].intent ?>** e **<? intents[0].confidence ?>!**

# Dicas avançadas - Watson Conversation

## Alguns exemplos práticos:

- Contador:
  - Contador pode ser uma informação de contexto mantida durante toda interação com o WCS.
  - Use variáveis de contexto:

```
{  
    "context": {  
        "counter": "<? context.counter + 1 ?>"  
    }  
}
```
- Para checar o valor em alguma condição:
  - **\$counter > 5**

# Dicas avançadas - Watson Conversation

## Alguns exemplos práticos:

- Manipulando arrays:
  - Considere a seguinte array inicializada previamente:

```
{  
  "context": {  
    "attributes": ["gps", "4g", "black"],  
    "size": 3  
  }  
}
```
  - Para adicionar mais atributos de acordo com o input do usuário:

```
{  
  "context": {  
    "attributes": "<? $attributes.append(input.text) ?>"  
    "size": "<? $attributes.size() ?>"  
  }  
}
```

# Dicas avançadas - Watson Conversation

## Alguns exemplos práticos:

- Objetos JSON complexos:
  - Possibilidade de guardar qualquer objeto JSON no contexto.
    - Não é necessário serializar para um valor em String.
    - Mantenha os dados (semi)-estruturados da maneira que desejar.
- Exemplo:

```
{  
  "context": {  
    "user": {  
      "username": "rodrigogs",  
      "first_name": "Rodrigo",  
      "last_name": "Sclosa",  
      "role": "admin",  
      "channel": "facebook_bot"  
    }  
  }  
}
```

Para manipular esse objeto:

- Checar se propriedade existe:
  - **`$user.has('username')`**
- Remover propriedade:
  - **`$user.remove('channel')`**



# Dicas avançadas - Watson Conversation

## Alguns exemplos práticos:

- Lidando com Strings:
  - String é o objeto mais importante do WCS.
  - Operações suportadas:
    - Sem argumentos: **length()**, **isEmpty()**, **trim()**
    - Argumentos String: **toUpperCase('str')**, **toLowerCase('str')**, **endsWith('str')**, **startsWith('str')**
    - Argumentos Inteiros: **substring(startIdx, endIdx)**
    - Argumentos Objeto: **append(obj)**
    - Regex: **matches(regExp)**, **extract(regExp, groupIdx)**, **split(regExp)**

# Dicas avançadas - Watson Conversation

## Alguns exemplos práticos:

- Possibilidade de utilizar expressões regulares:
  - Condição no nó: **"input.text.matches('[0-9]+')"**
- Extraíndo e guardando o valor encontrado:

```
{  
  "context": {  
    "number": "<?input.text.extract('[0-9]+', 0)?>"  
  }  
}
```

# Dicas avançadas - Watson Conversation

## Alguns exemplos práticos:

- Convertendo números:
  - Importante para evitar comparar Strings com números.
  - Se a conversão falhar, será retornado null
    - Logo, atenção para não receber nullpointer exceptions!

Operação	Descrição
toInt()	Converte objeto/campo para inteiro
toLong()	Converte objeto/campo para long
toDouble()	Converte objeto/campo para double

# DEMONSTRAÇÃO

# Demonstração

- Aplicativo de exemplo:
  - <https://comunicacao-sinistros-chat.mybluemix.net>
- Workspace:
  - <https://www.ibmwatsonconversation.com/us-south/3f68f108-ead9-43de-851f-d82b68c5db7b/workspaces>

## Demonstração

- Repositório no GitHub
  - <https://github.com/rodrigosclosa/cafeconhecimentowatson>





Collaborate. Innovate. Transform.

# Comunicação de Sinistros

Com IBM Watson Visual Recognition

Denis de Matos Santaterra  
Março 2017

# Visual Recognition

- API que permite analisar e entender o conteúdo de imagens.
- Utiliza as últimas tecnologias referente a Deep Learning

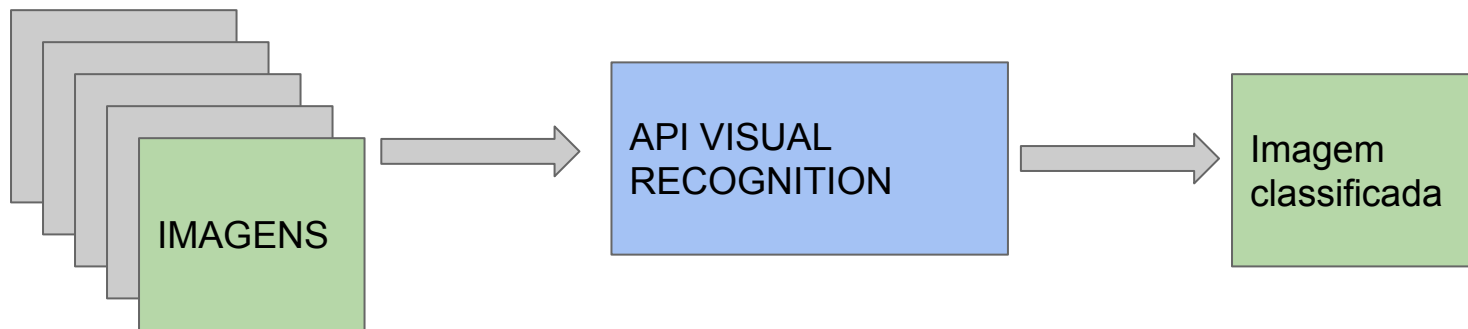
Permite:

- Classificar imagens utilizando os classificadores padrões (out of the box)
- Detecção de faces (posição), atributos como sexo, idade e verificação se a imagem corresponde a alguma pessoa famosa
- Detecção de Textos
- Criação de classificadores customizados
- Busca por similaridade em banco de dados de imagens



## Possíveis casos de uso

- Possibilidade de identificar automaticamente quem está entrando em um edifício por motivos de segurança
- Monitorar linhas de produção para garantir que está seguindo a especificação
- Categorização de imagens médicas
- Filtragem de conteúdo adulto
- **Classificar ocorrências em um Sinistro utilizando imagens**



# Criando Visual Recognition

- Cadastre-se no Bluemix: <https://console.ng.bluemix.net/>
- Acesse: Catalog -> Watson -> Visual Recognition

The screenshot displays the IBM Bluemix Catalog interface. At the top, a dark navigation bar includes the 'Docs' link, a '24 Trial Days Remaining' status indicator, and regional/tenant information ('CI&T | US South : TreinamentoCafeConhecimento : dev'). The main header features the 'IBM Bluemix Catalog' title and three primary navigation tabs: 'Catalog' (highlighted with a red box and labeled '1'), 'Support', and 'Manage'.

On the left side, a sidebar lists various categories under 'Services'. The 'Watson' category is highlighted with a red box and labeled '2', with a right-pointing arrow indicating further options. Other categories listed include Boilerplates, Cloud Foundry Apps, Containers, OpenWhisk, Mobile, Data & Analytics, Internet of Things, APIs, Network, Storage, Security, DevOps, Application Services, and Integrate.

The main content area displays a grid of service cards. Each card includes an icon, a title, a brief description, and an 'IBM' logo. The 'Visual Recognition' card is highlighted with a red box and labeled '3'. It features an icon of a magnifying glass over a document and the text: 'Visual Recognition', 'Find meaning in visual content!', 'Analyze images for scenes.', and the 'IBM' logo.

Other visible service cards include: 'Natural Language Understanding' (Analyze text to extract meta-data from content such as), 'Personality Insights' (The Watson Personality Insights derives insights from), 'Retrieve and Rank' (Add machine learning enhanced search capabilities to your), 'Speech to Text' (Low-latency, streaming transcription), 'Text to Speech' (Synthesizes natural-sounding speech from text.), and 'Tone Analyzer' (Tone Analyzer uses linguistic analysis to detect three types).

# Criando Visual Recognition

- Insira o service name e credential name -> Create
- Para obter a credencial click em **Service credentials** -> view credentials

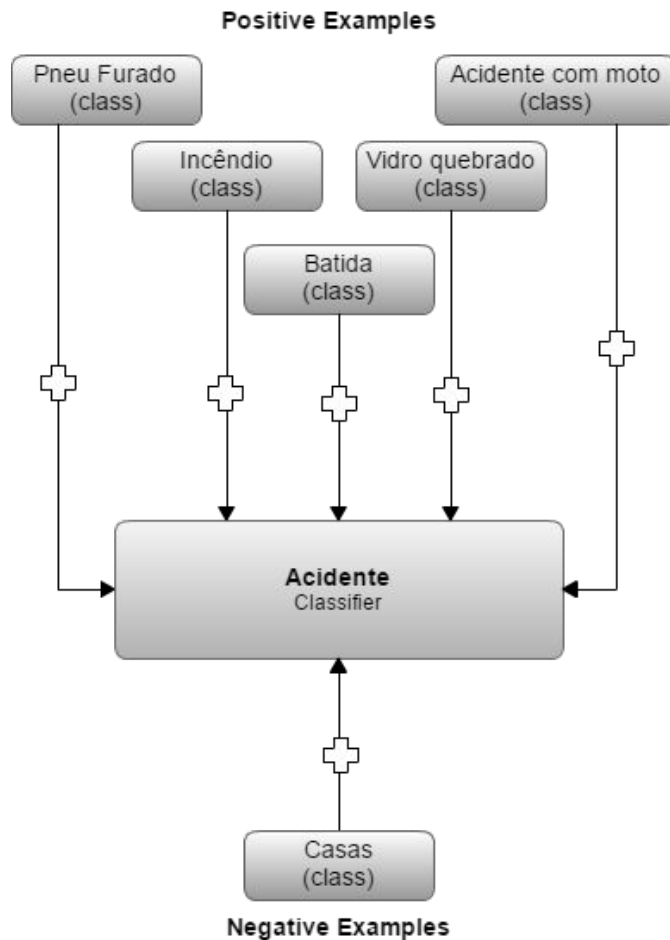
Para testar utilize o facilitador (API Explorer):

<https://watson-api-explorer.mybluemix.net/apis/visual-recognition-v3>

Faça um teste para o endpoint POST /classify

Faça um teste para o endpoint POST /detect\_faces

# Estrutura para criação de um classificador customizado



# Classificador customizado

- Classificador customizado: acesse a pasta \Insurance
- Vamos utilizar o Postman

[https://gateway-a.watsonplatform.net/visual-recognition/api/v3/classifiers?api\\_key=<<API\\_KEY>>&version=2016-05-20](https://gateway-a.watsonplatform.net/visual-recognition/api/v3/classifiers?api_key=<<API_KEY>>&version=2016-05-20)

https://gateway-a.wat x + No Environment

POST https://gateway-a.watsonplatform.net/visual-recognition/api/v3/classifiers?api\_key=5d6030c93... Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	name	Acidente	
<input checked="" type="checkbox"/>	batida_positive_examples	<input type="button" value="Escolher arquivos"/> batida.zip	
<input checked="" type="checkbox"/>	com_moto_positive_examples	<input type="button" value="Escolher arquivos"/> com_moto.zip	
<input checked="" type="checkbox"/>	incendio_positive_examples	<input type="button" value="Escolher arquivos"/> incendio.zip	
<input checked="" type="checkbox"/>	pneu_furado_positive_examples	<input type="button" value="Escolher arquivos"/> pneu_furado.zip	
<input checked="" type="checkbox"/>	vidro_quebrado_positive_examples	<input type="button" value="Escolher arquivos"/> vidro_quebrado.zip	
	New key	value	

# DEMONSTRAÇÃO

# Documentação

- Endpoint: <https://gateway-a.watsonplatform.net/visual-recognition/api>
- Autenticação: API Key obtida pelos detalhes do serviço no Bluemix
  - <https://www.ibm.com/watson/developercloud/doc/visual-recognition/>
- API: <https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/>

# Temos novidades :)

[https://sites.google.com/s/oB\\_FLHULEkF4xQUZ1YUF1REJReTg/p/oB\\_FLHULEkF4xOFR4cU5JMDVVMzA/edit](https://sites.google.com/s/oB_FLHULEkF4xQUZ1YUF1REJReTg/p/oB_FLHULEkF4xOFR4cU5JMDVVMzA/edit)





