AI & CHATBOT

Aula 18 – Disponibilização de Modelos

Prof. Henrique Ferreira

FIAP
GRADUAÇÃO

Modelos via Servidor Flask

Disponibilizando um modelo prétreinado

Disponibilizando Modelos de IA

- Após construirmos um modelo de IA (por exemplo, ao treinar um algoritmo de Aprendizado de Máquina), podemos estar interessados em usá-lo na nossa aplicação;
- Há várias formas de fazer isso mas a ideia básica é usar um backend para chamar o modelo, por exemplo, executando um script em Python no servidor;
- Podemos usar o backend em Node-RED (NodeJS) ou em Python (Flask) para fazer isso;

Disponibilizando Modelos de IA

- Primeiro precisamos exportar o Modelo Treinado.
- Se o modelo foi criado usando o Scikit-Learn, podemos exportá-lo no formato Joblib ou Pickle
- Gere um modelo como visto na aula prática de classificação (6° passo, produção):

A etapa de "salvar/baixar" o modelo pronto é chamada de **Persistência do Modelo**. Quando usamos um modelo pronto, é comum dizer que estamos usando um **modelo pré-treinado**.

```
1 import pickle
```

Pickle é um formato de arquivo de objeto python serializado na memória permanente. Ele é análogo ao JSON para dicionários ou o CSV para tabelas, mas ele funciona para objetos python, ou seja, é muito mais abrangente!

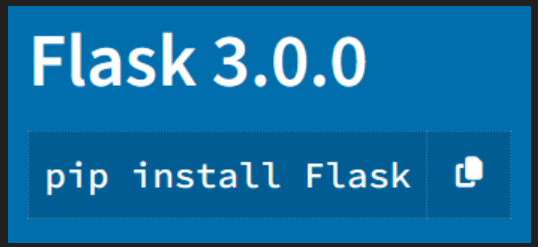
```
1 # vamos salvar em bytes (flag wb) para ser mais cross-platform (acessível a vários sistemas)
2 with open('meu_modelo_serializado.pickle', 'wb') as f:
3     pickle.dump(lda, f)
```

Pronto! Nosso modelo está salvo. Se quisermos usá-lo em um código em Python, podemos simplesmente carregá-lo:

Disponibilizando Modelos de IA

- O Flask é um framework web em Python que nos ajuda a criar um “software servidor” para o backend da nossa aplicação de IA;

- Instale o Flask no seu Python usando o pip:



- Exemplo de um Hello World em Flask:

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Olá mundo!"

if __name__ == "__main__":
    app.run()
```

Disponibilizando Modelos de IA

- Vamos criar um script em Python para disponibilizar o modelo.

```
1  from flask import Flask, request, jsonify
2  import numpy as np
3  import pickle
4
5  app = Flask(__name__)
6
7  # Modelo de exemplo - substitua isso pelo seu modelo treinado
8  # Aqui estamos carregando o modelo já treinado que está no arquivo JobLib
9  with open('C:/Users/Ferreira/Desktop/modelo_treinado_gafesp.pickle', 'rb') as f:
10     modelo = pickle.load(f)
```

Estamos carregando o arquivo do modelo de IA prétreinado aqui. Troque pelo caminho do seu arquivo..

Disponibilizando Modelos de IA

- Vamos criar um script em Python para disponibilizar o modelo.

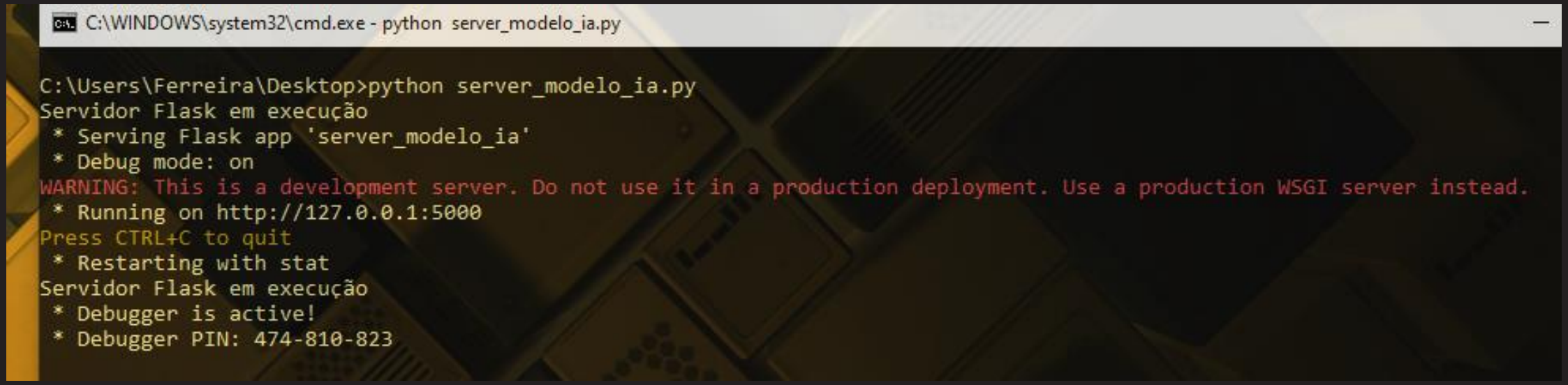
```
12  # Rota para receber os dados e fazer previsões
13  @app.route('/prever', methods=['GET'])
14  def prever():
15      # Obter parâmetros da solicitação GET
16      parametro1 = float(request.args.get('comp_abd'))
17      parametro2 = float(request.args.get('comp_ant'))
18
19      # Fazer previsões usando o modelo (substitua por suas próprias lógicas)
20      entrada = np.array([[parametro1, parametro2]])
21      resultado = modelo.predict(entrada)
22
23      # Retornar o resultado como JSON
24      return jsonify({'previsao': resultado.tolist()})
25
26  if __name__ == '__main__':
27      # Executar o aplicativo Flask
28      app.run(debug=True)
```

Variáveis de entrada do problema de classificação gaf_esp visto em aula

Saída do modelo

Disponibilizando Modelos de IA

- Executando o servidor Flask:



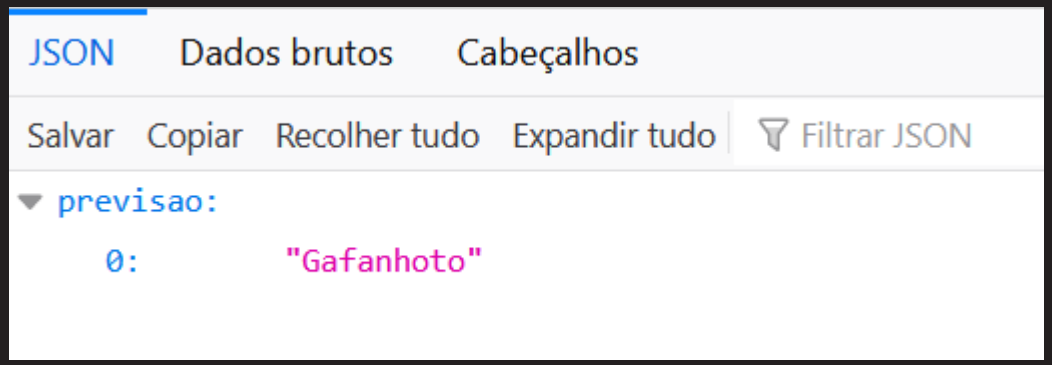
- No navegador, acesse o http://127.0.0.1:5000 com a composição da query:

http://127.0.0.1:5000/prever?comp_abd=1.0&comp_ant=2.0

Disponibilizando Modelos de IA

- http://127.0.0.1:5000/prever?comp_abd=1.0&comp_ant=2.0

A resposta será a predição do modelo quando comp_abd=1 e comp_ant=2



Você pode alterar o script para ter outras rotas, outras variáveis de entrada, ou gerar um resposta JSON mais elaborada.

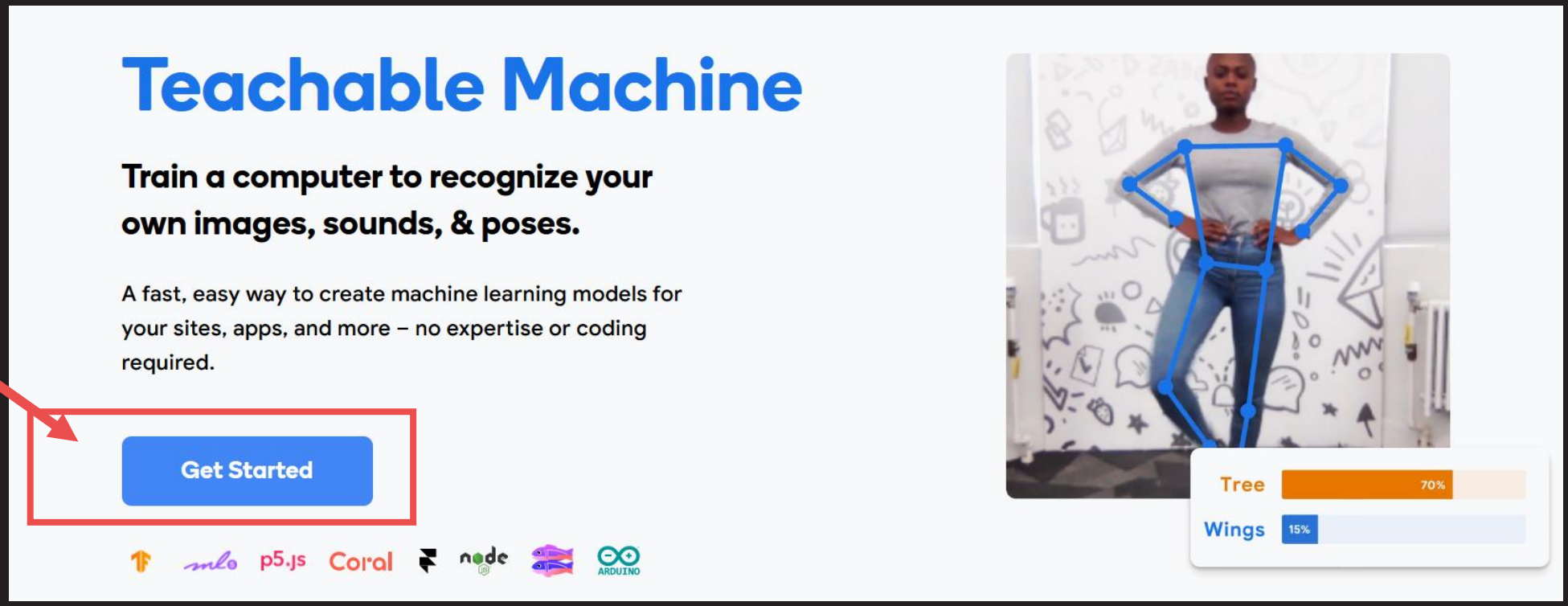
Modelos via Node-RED

Importando um Modelo do Teachable Machine

Teachable Machine

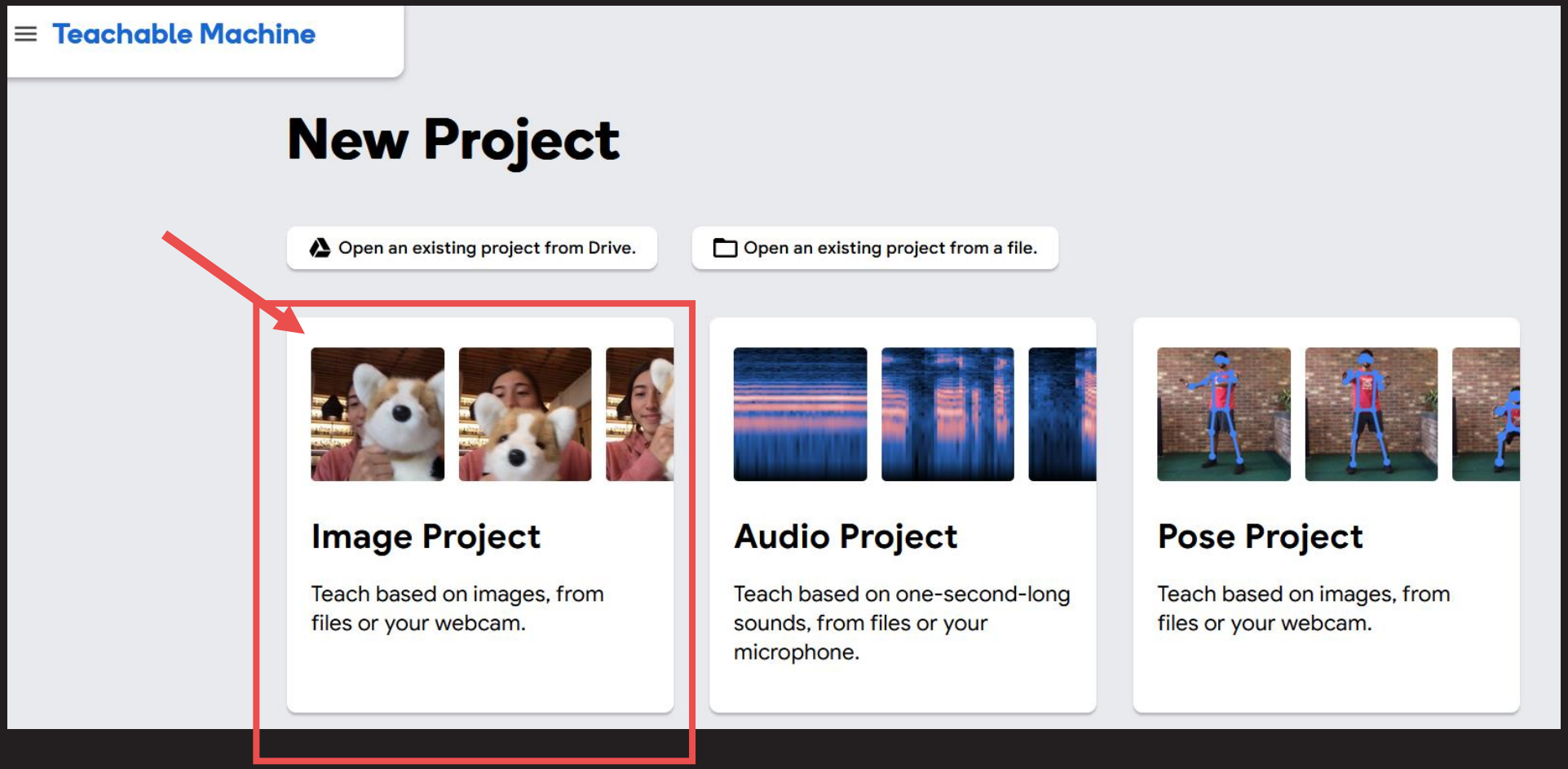
- O Teachable Machine é uma ferramenta do Google para ensinar sobre Machine Learning e Reconhecimento de Imagens:

https://teachablemachine.withgoogle.com/



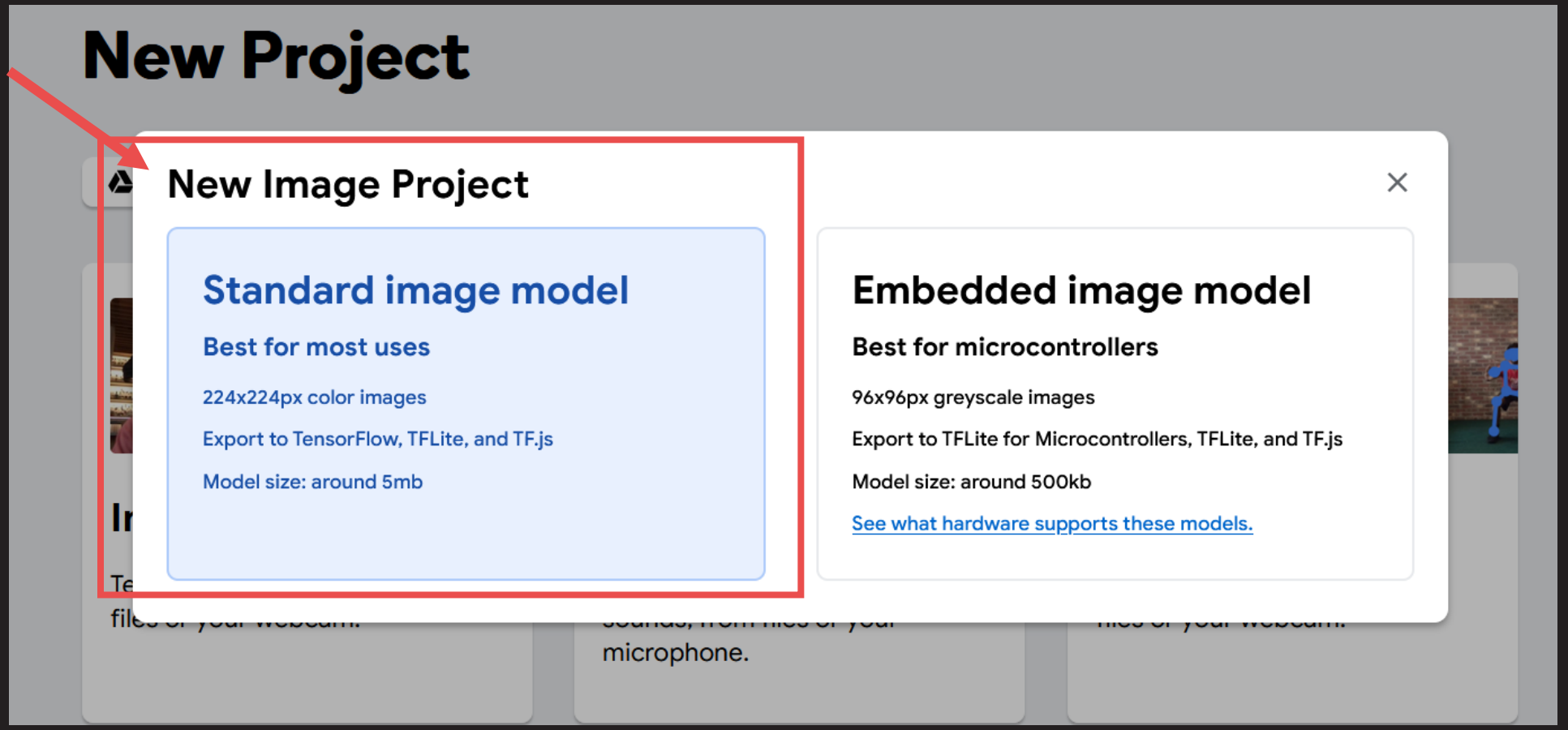
Teachable Machine

- Selecione a opção Image Project



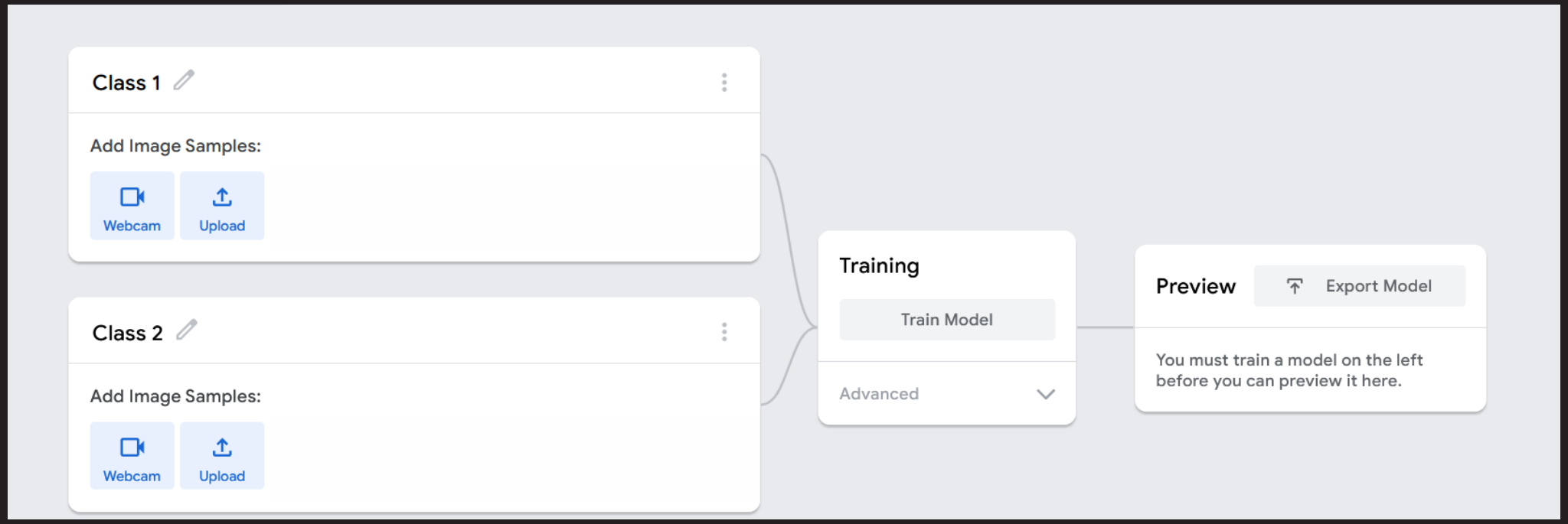
Teachable Machine

- Selecione a opção seguida de Standard image model



Teachable Machine

- Treine um classificador de imagens usando sua Webcam ou dados do seu computador;
- Lembre-se de rotular adequadamente cada conjunto de imagens;



TM + Node-RED

- Instale a lib do Teachable Machine no Node-RED:

node-red-contrib-teachable-machine 1.4.1

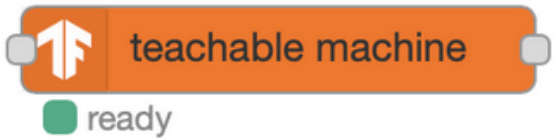
Simplifies integration with Teachable Machine models from Google

```
npm install node-red-contrib-teachable-machine
```

Node-RED node | pre-commit.ci passed | CI passing | npm v1.4.1 | downloads 10k | quality ★★★★½

code style standard | maintainability A | license Apache-2.0 | donate PayPal

A Node-RED node based in tensorflow.js that enables to run custom image classification trained models using Teachable Machine tool. All notable changes to this project will be documented in the CHANGELOG.md file.



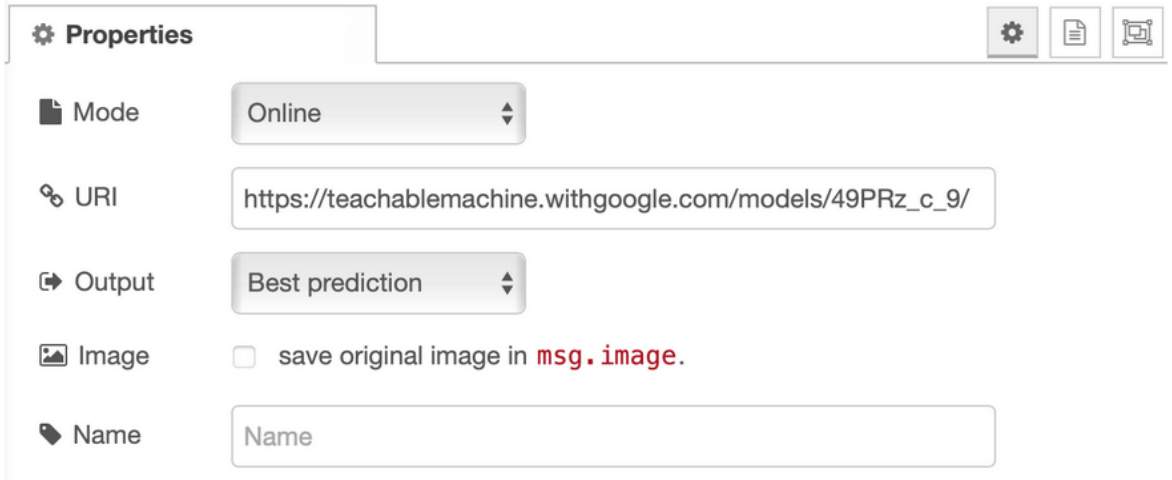
TM + Node-RED

- Os passos para utilização do nó estão em:

https://flows.nodered.org/node/node-red-contrib-teachable-machine

Online Mode

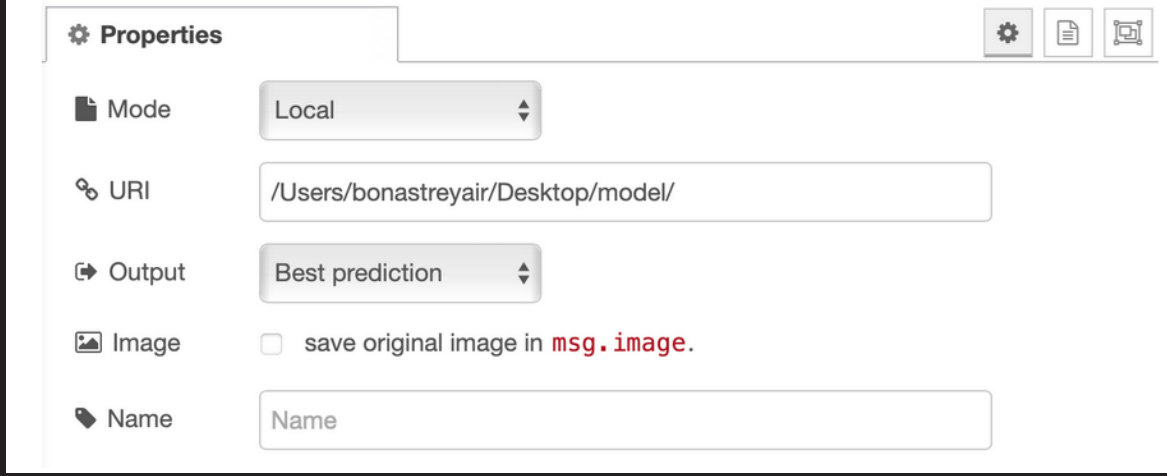
Select Online Mode and paste the saved URL in the node configuration. That URL hosts all the information to load your trained model. Make sure you copy all the given URL including the `https://...` and the `/` in the end.



Local Mode

Download all three files from the generated URL and save them locally in a folder maintaning the original filenames.

Select Local Mode and write down the absolute path of the folder that contain the three downloaded files in the node configuration. Make sure it ends with a `/` noting it is a folder.

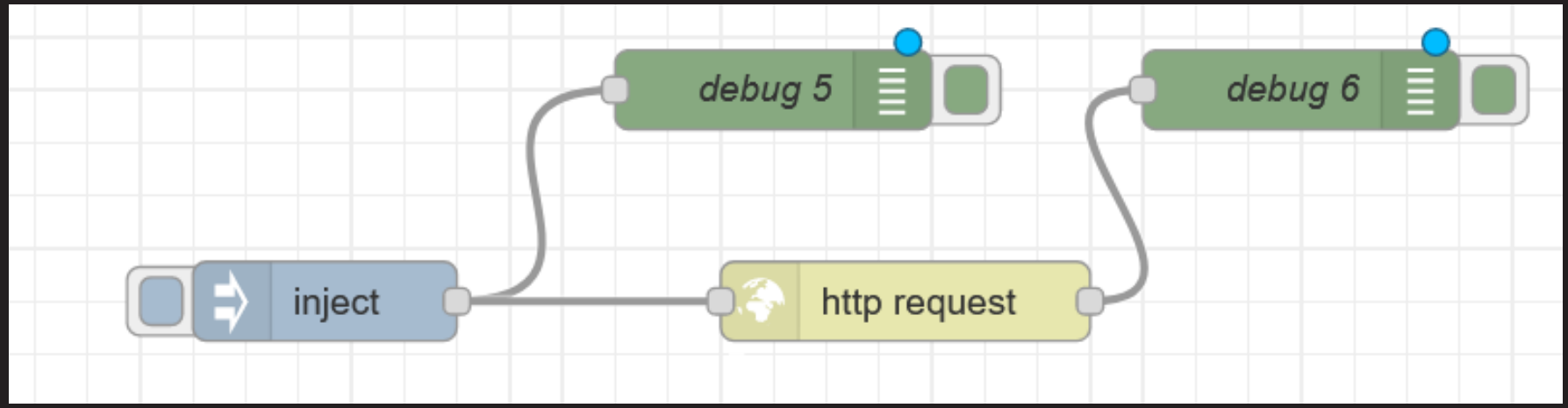


Flask + Node-RED I

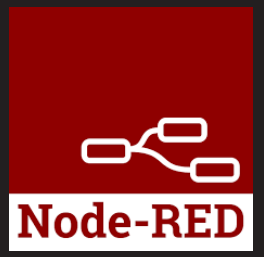
Comunicando dois servidores

Flask + Node-RED

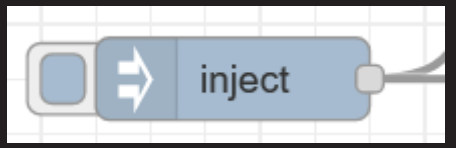
- Vamos fazer nossos Servidor Flask comunicar-se com o Servidor Node-RED;
- Primeiro, vamos aprender a configurar o nó http request.
- Monte o seguinte fluxo:

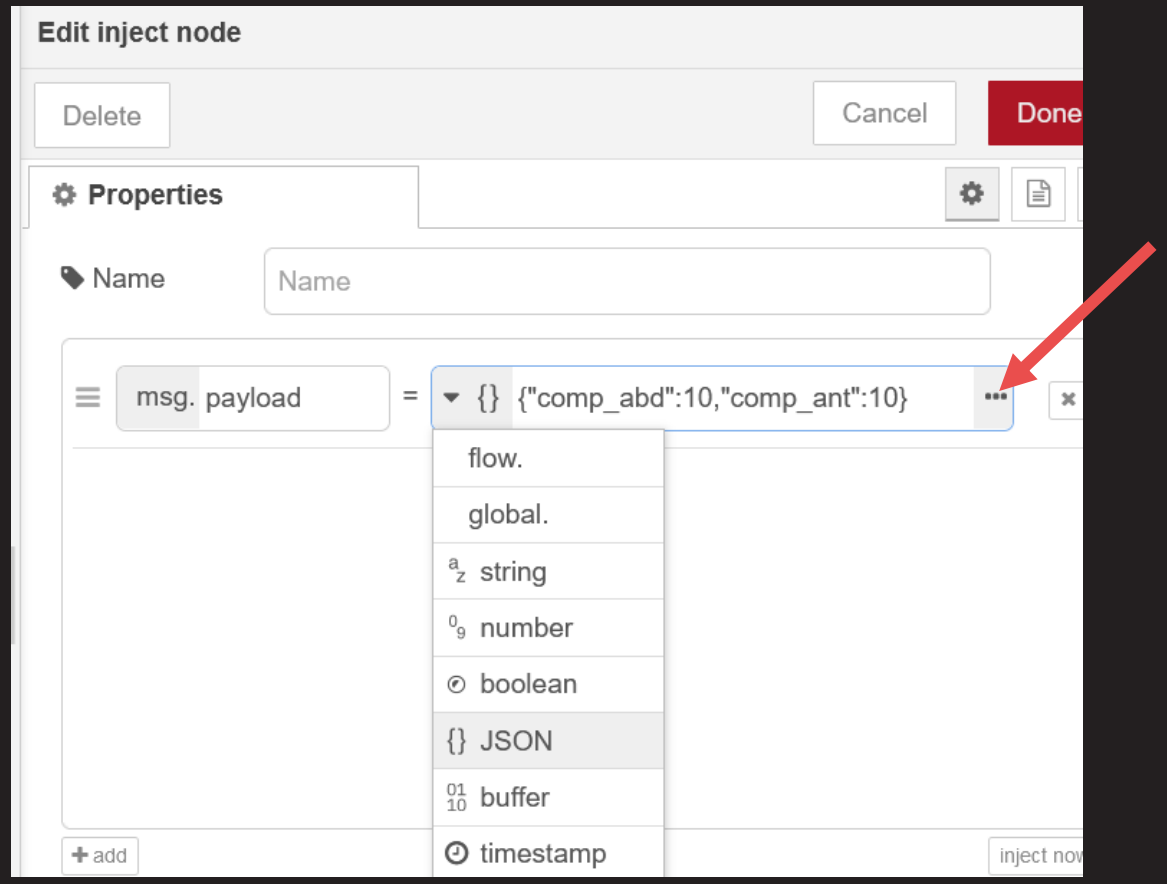


Flask + Node-RED



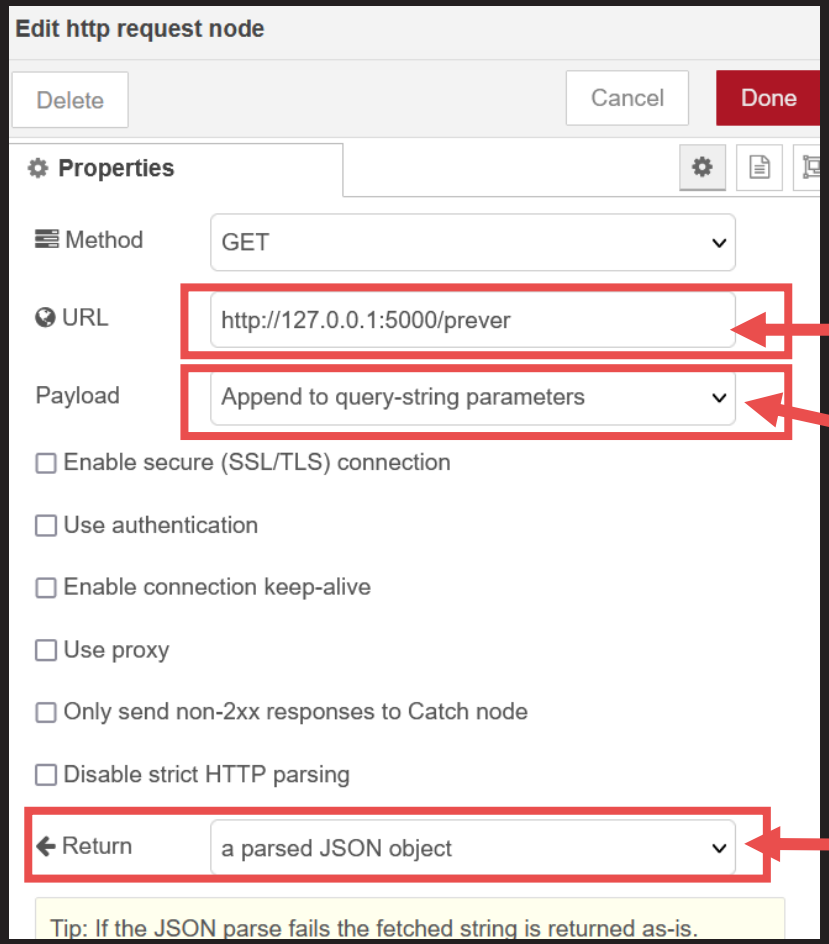
- No nó de inject escolha a opção JSON e crie um dicionário com as variáveis que nosso modelo de IA espera receber:

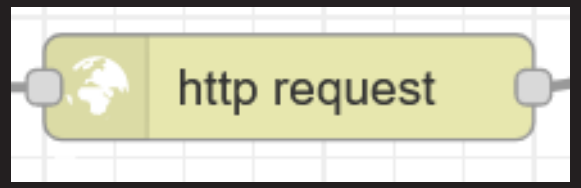




Flask + Node-RED

- No nó de http request





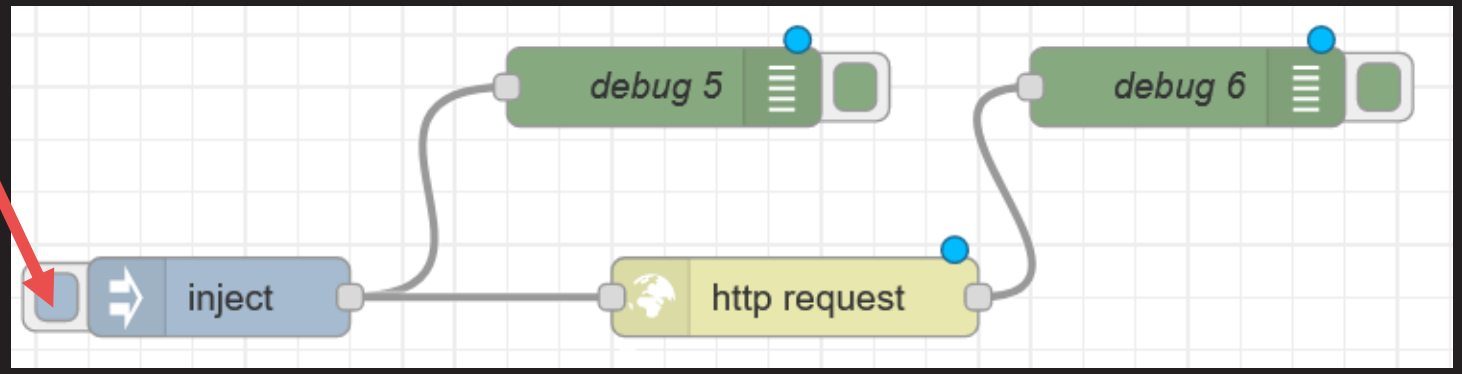
Coloque o IP e a rota do servidor Flask

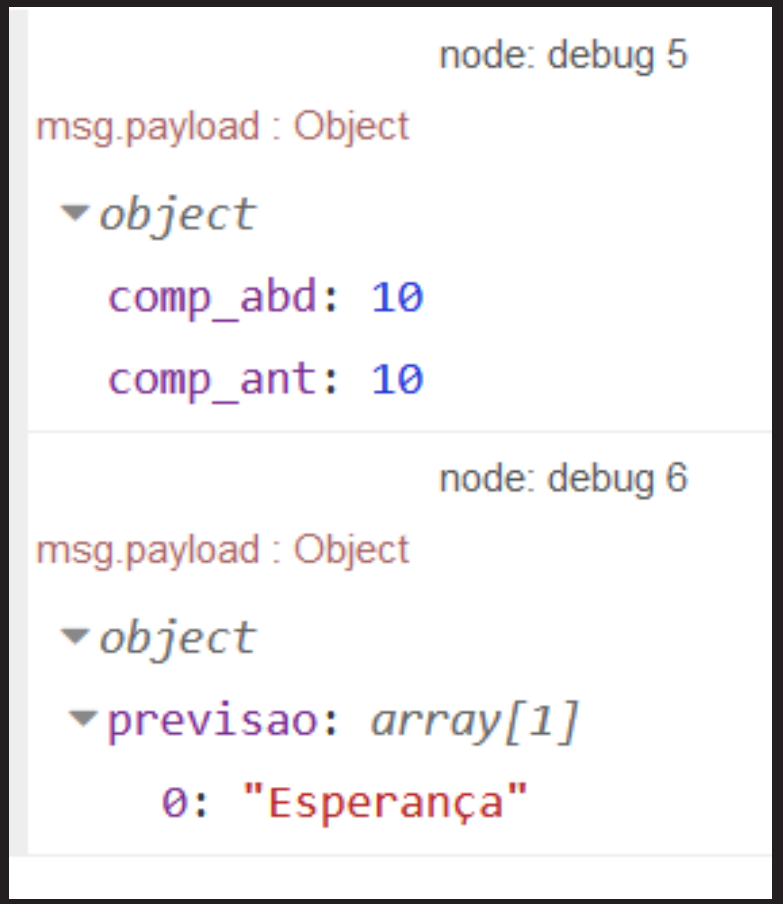
Coloque para receber a msg.payload como parâmetro de query do GET

Troque o tipo de retorno para vir formatado como objeto JSON

Flask + Node-RED

- Injete a mensagem. Você deverá ver a resposta do modelo de IA no menu de debug do Node-RED
- Na prática, seu servidor Node-RED comunicou-se com seu servidor Flask que está disponibilizando o modelo de IA.



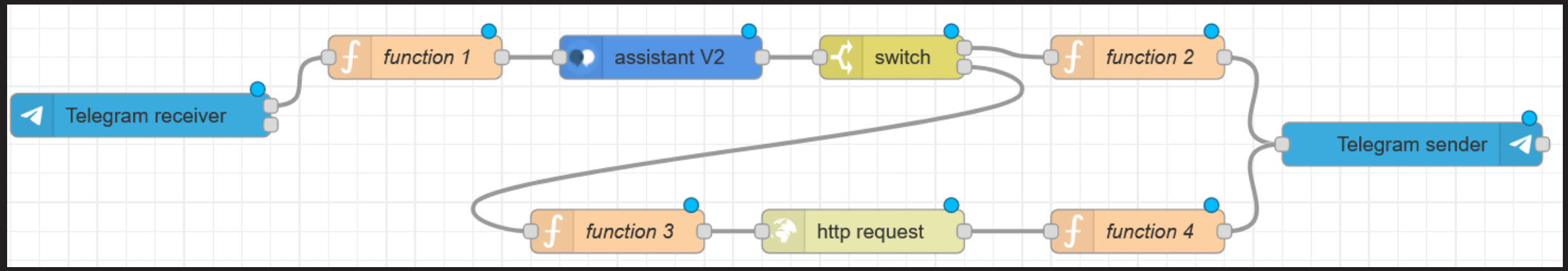


Flask + Node-RED II

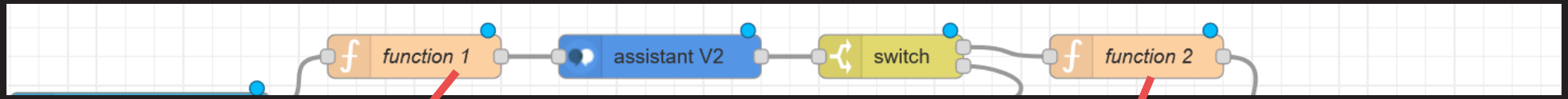
Integrando modelo preditivo de IA e chatbots

Flask + Node-RED + WA

- Com o servidor Flask rodando, vamos integrar um serviço de chatbot usando o Node-RED:



Flask + Node-RED + WA



Name function 1

Setup | On Start

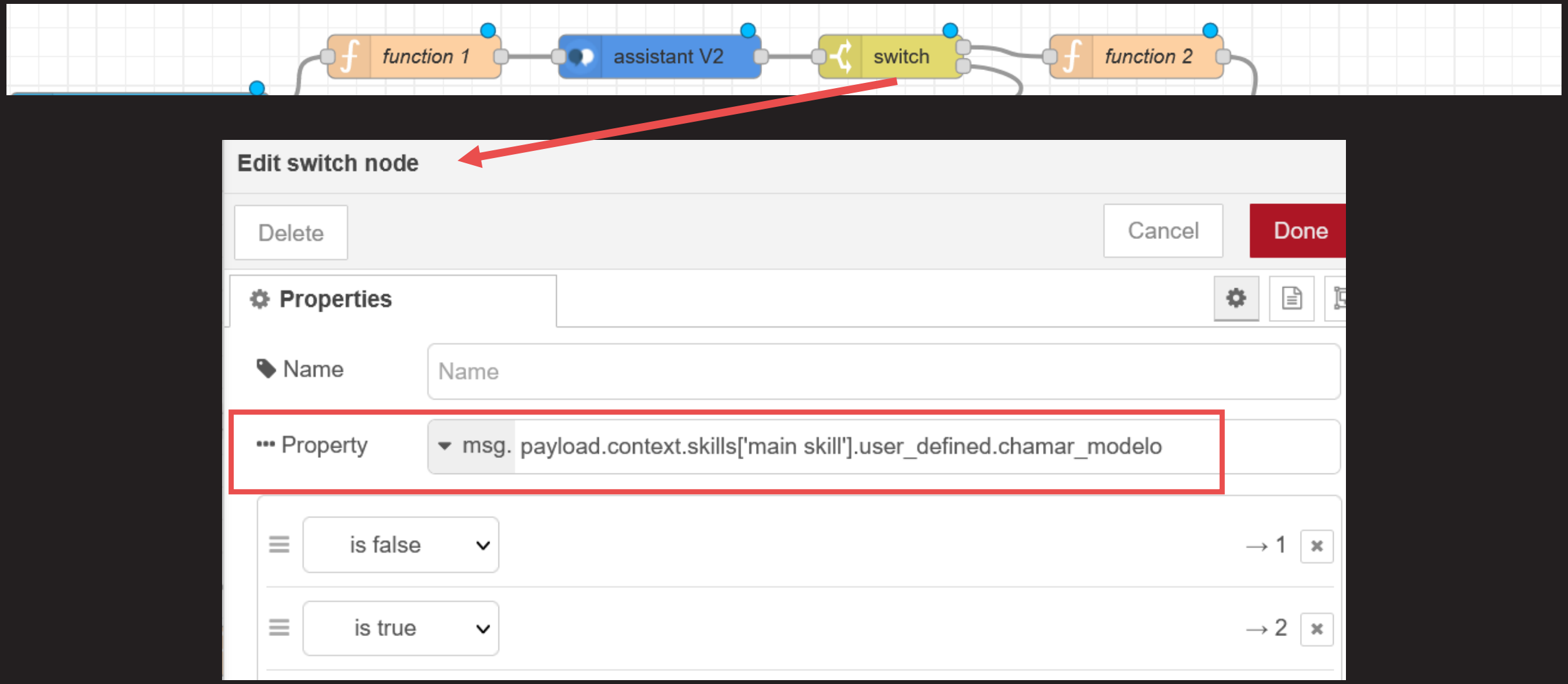
```
1  msg.params={
2      'session_id':msg.payload.chatId
3  }
4  msg.additional_context={
5      'chamar_modelo':false
6  }
7  msg.payload=msg.payload.content;
8  return msg;
```

Name function 2

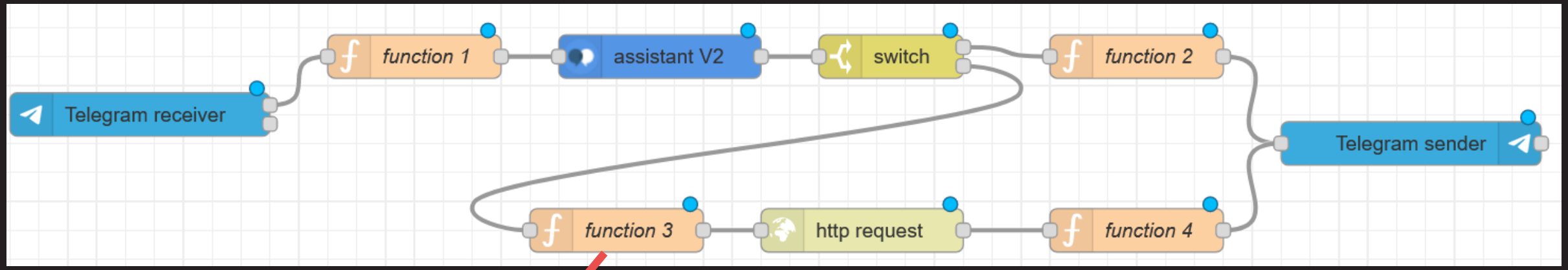
Setup | On Start | On Messa

```
1  msg.payload={
2      'content':msg.payload.output.generic[0].text,
3      'type':'message',
4      'chatId':msg.params.session_id
5  }
6  return msg;
```

Flask + Node-RED + WA



Flask + Node-RED + WA

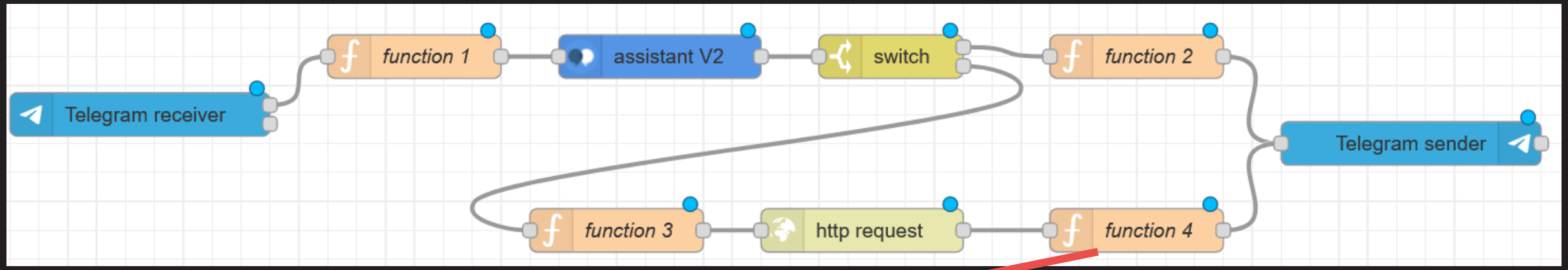


Name: function 3

Setup | On Start | **On Message** | On Stop

```
1  msg.payload = {
2      'comp_abd':msg.payload.context.skills['main skill'].user_defined.comp_abd,
3      'comp_ant':msg.payload.context.skills['main skill'].user_defined.comp_ant
4  }
5  return msg;
```

Flask + Node-RED + WA

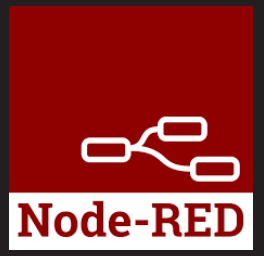


Name function 4

Setup | On Start | **On Message** | On

```
1  msg.payload={
2      'content': "Este inseto é um(a): " + msg.payload.previsao[0],
3      'chatId':msg.params.session_id,
4      'type':'message'
5  }
6  return msg;
```

Flask + Node-RED + WA



- O nosso bot no Watson Assistant será configurado como:

Intenções

Intents (2) ↑	Description
#classificador_insetos	O usuário quer usar o classificador de gafanhot...
#oqeh_entomologia	saber sobre entomologia

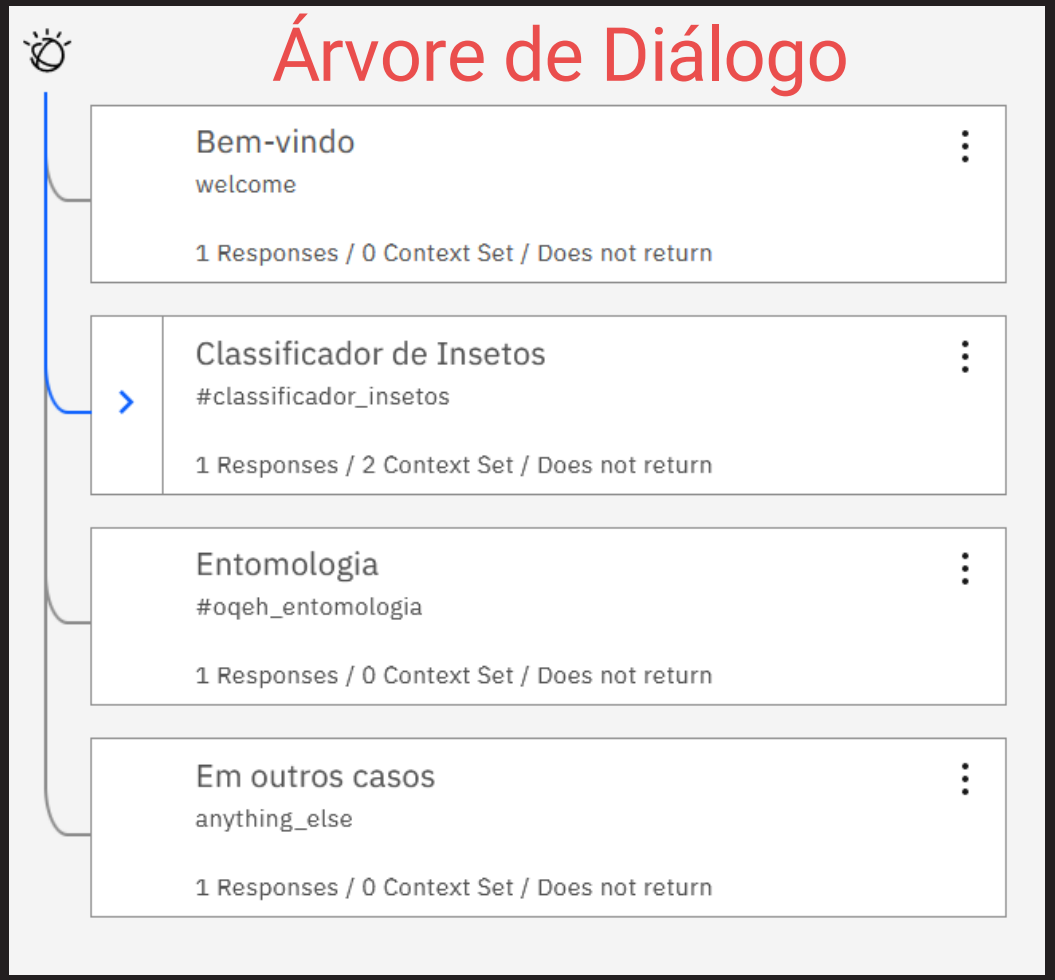
Entidades

@sys-number	Extracts numbers expressed as digits or written as numbers. (21)	On

Flask + Node-RED + WA

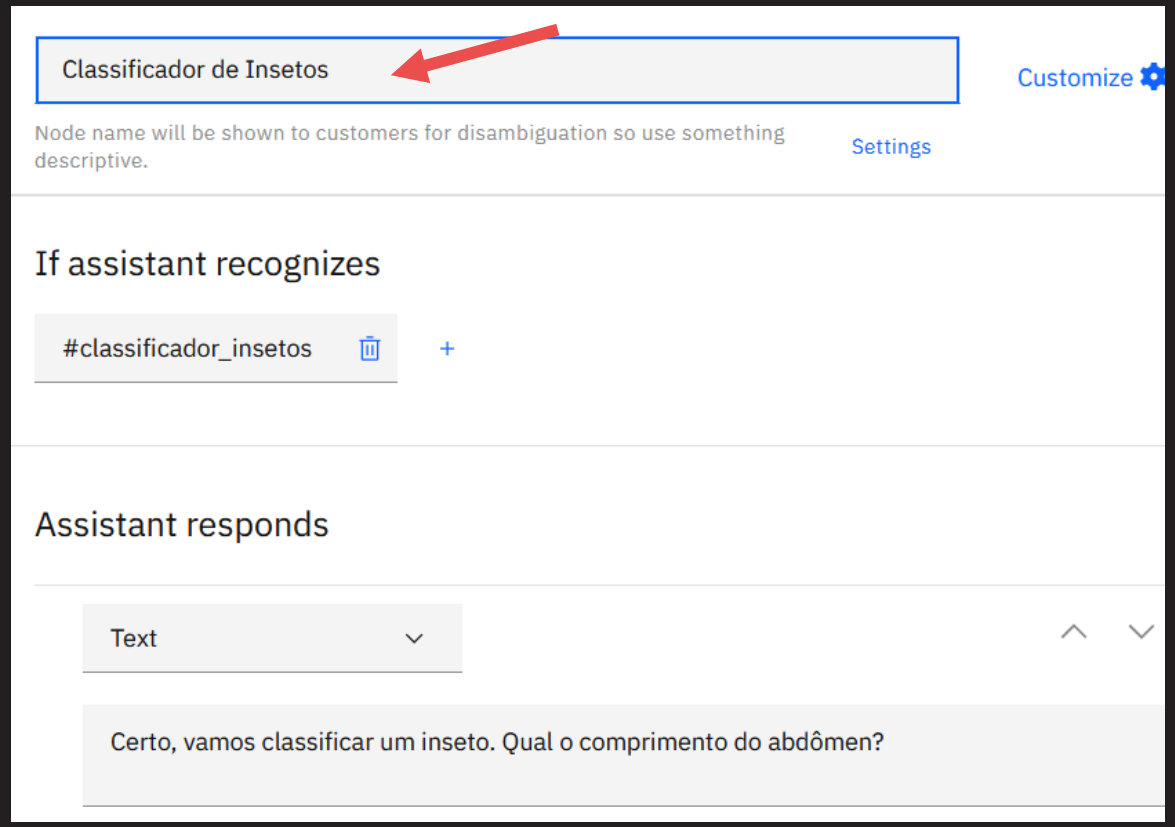
- O nosso bot no Watson Assistant será configurado como:

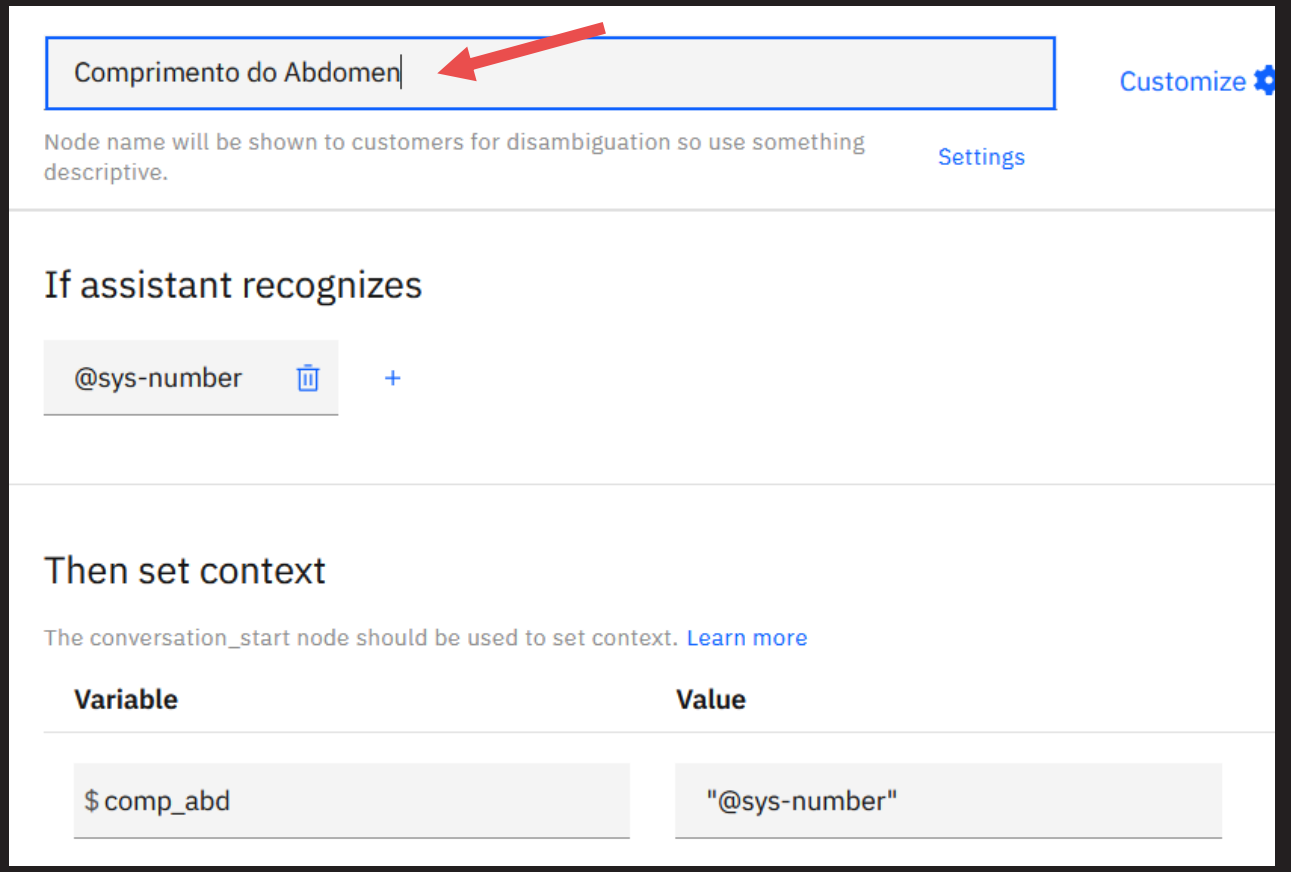
Árvore de Diálogo



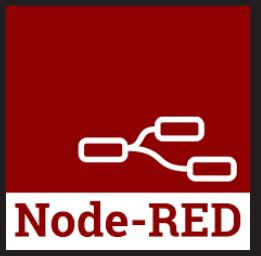
Flask + Node-RED + WA

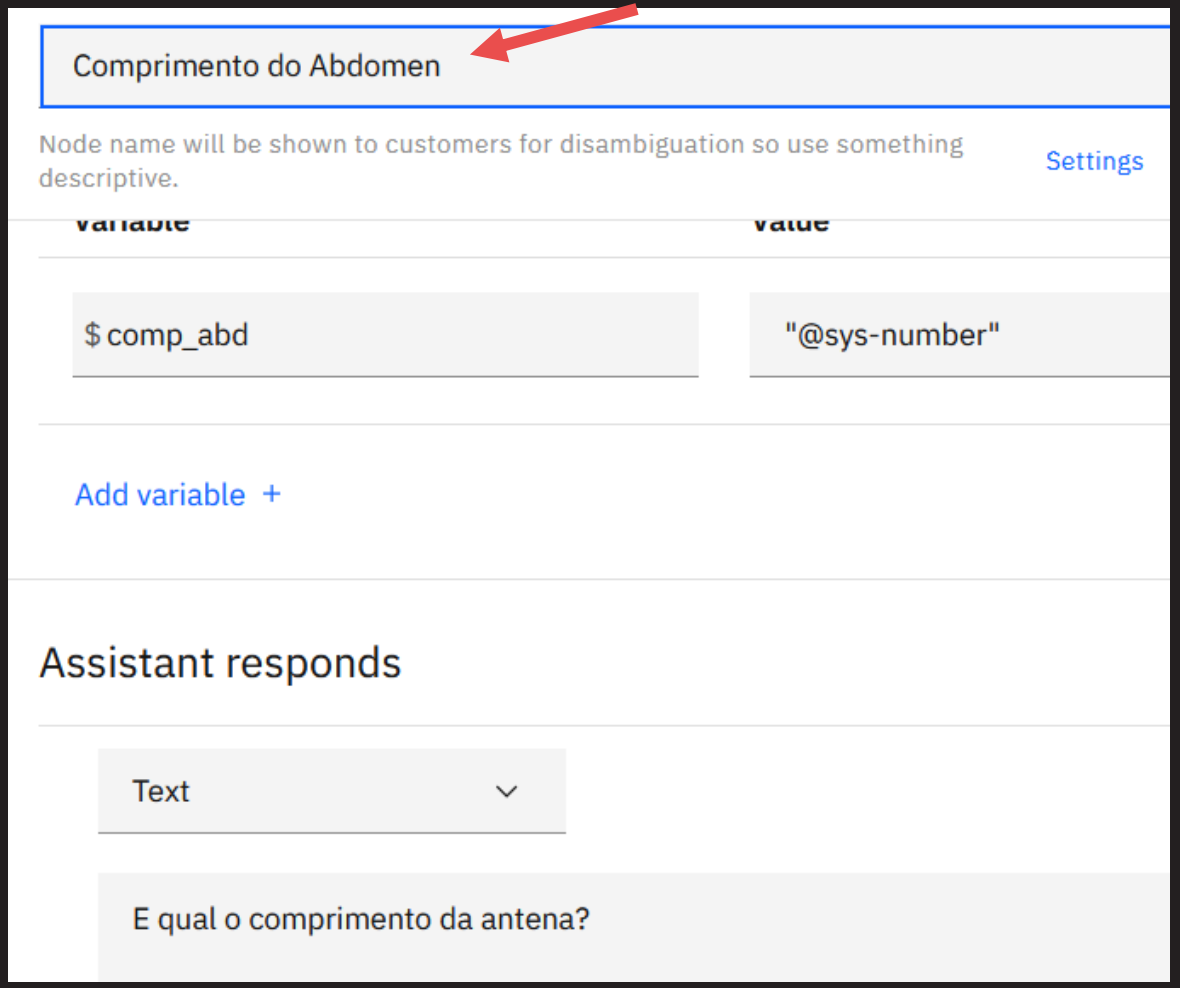
- Criando os nós de diálogo para capturar as variáveis de interesse:

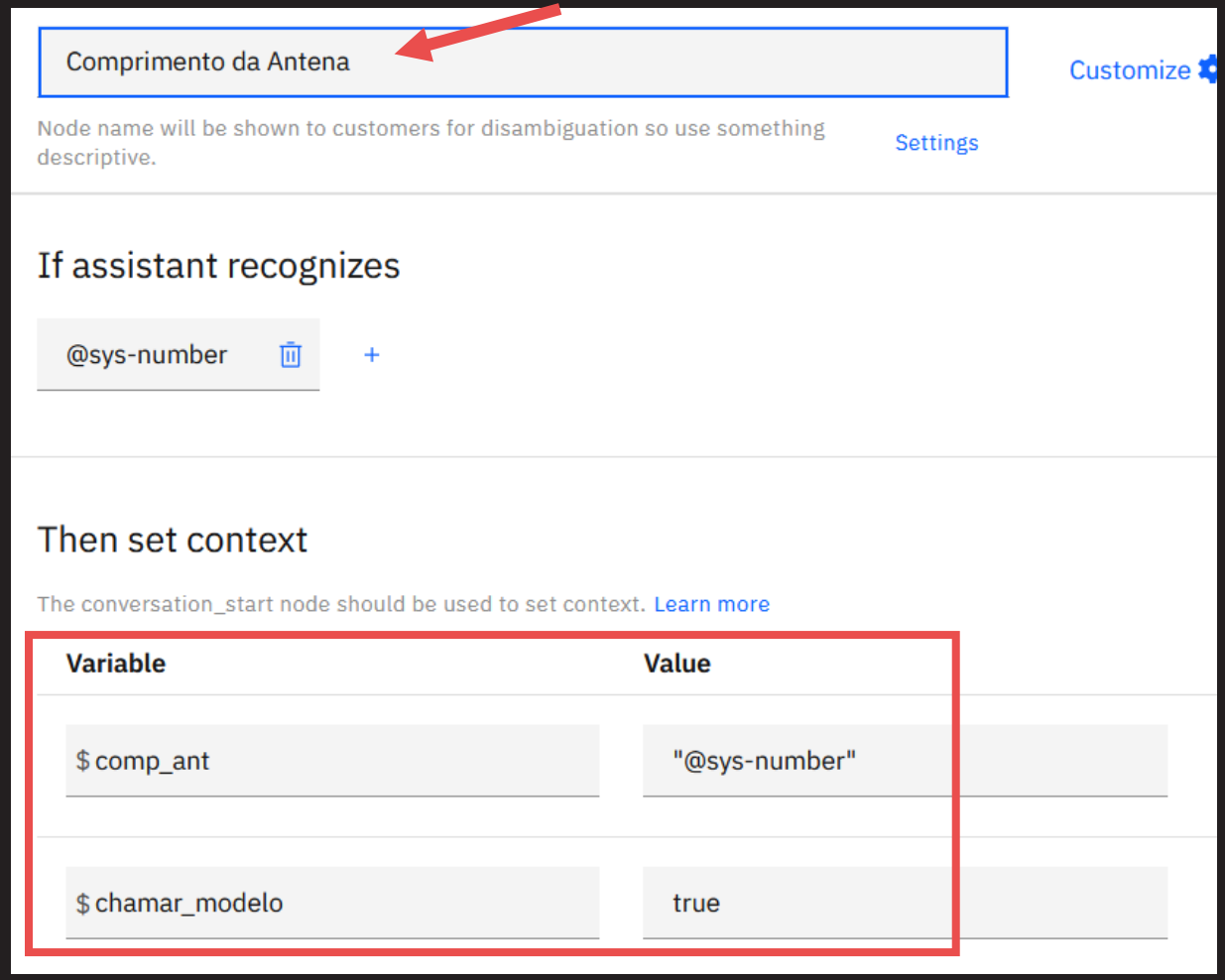




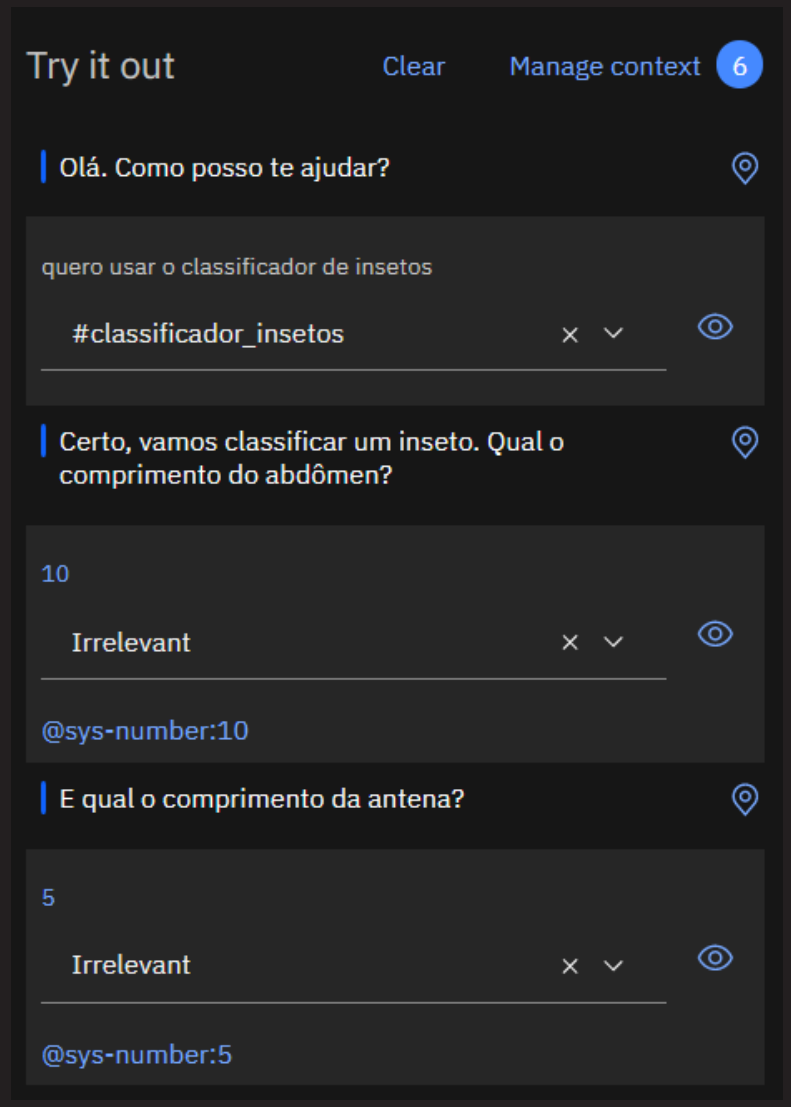
Flask + Node-RED + WA





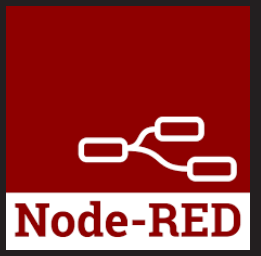


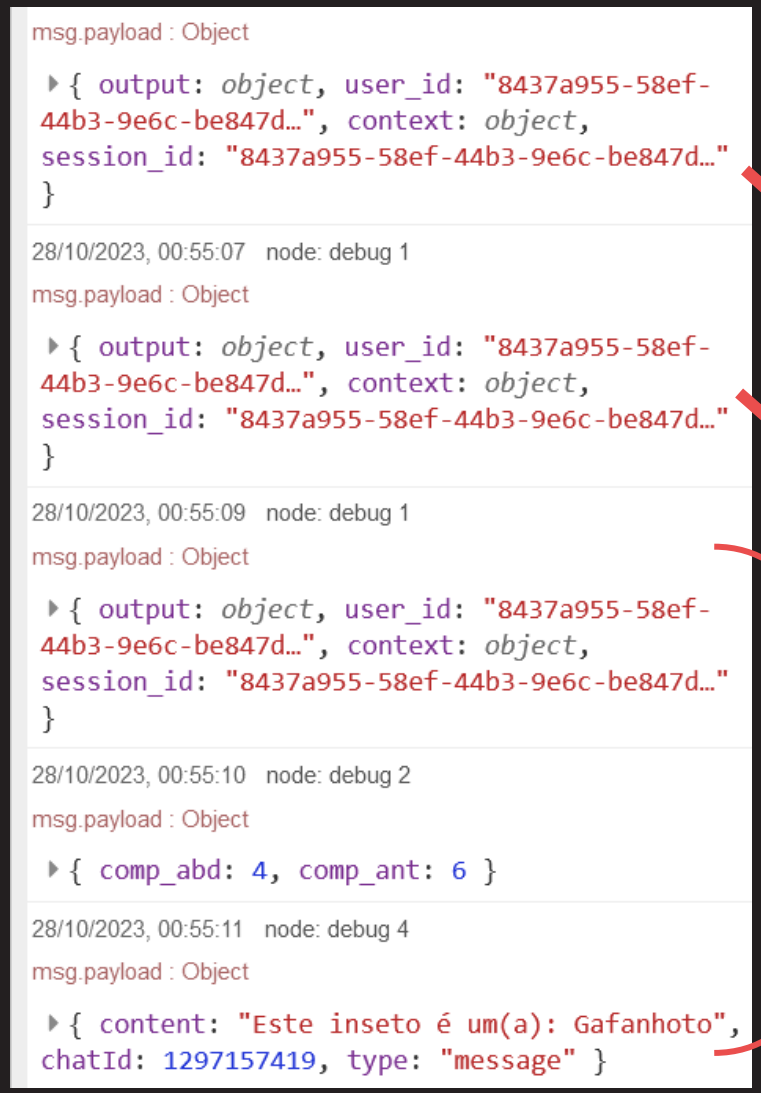
Flask + Node-RED + WA

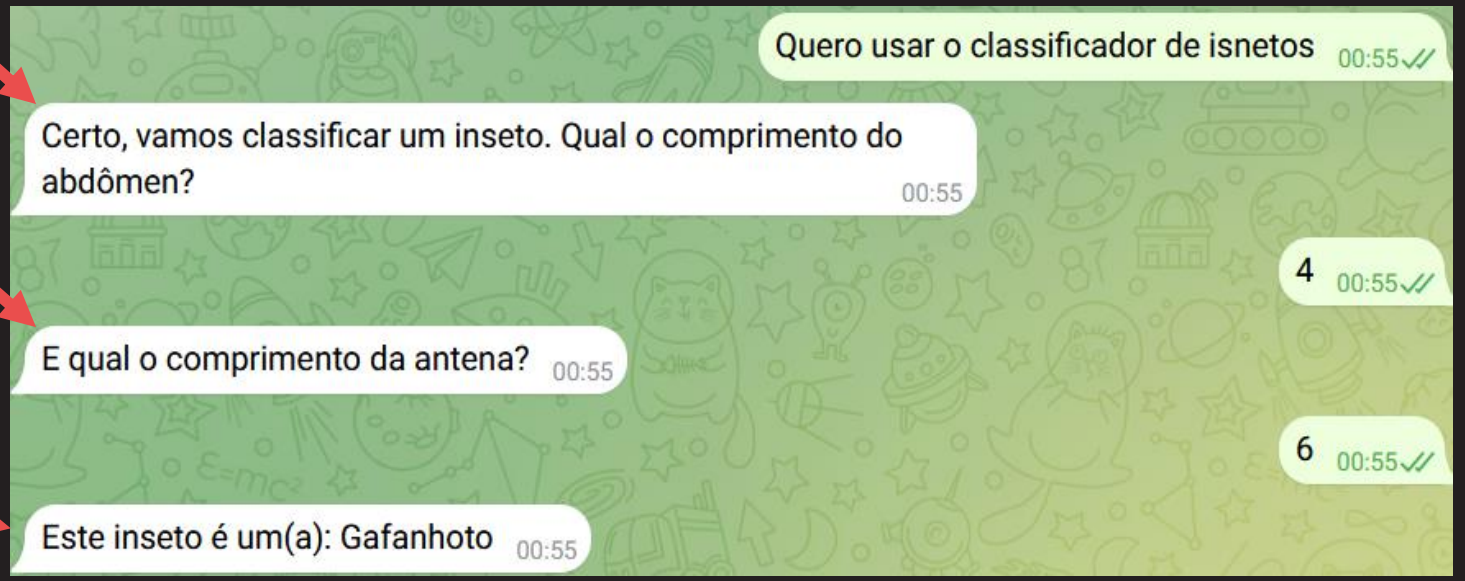


Perceba que o teste no try it do WA não veremos o resultado do modelo, já que este está sendo chamado pelo nosso servidor Node-RED:

Flask + Node-RED + WA







Agora é com você!

Teste seu conhecimento do assunto realizando exercícios extras

Exercício

1. Faça um bot de Telegram que gere um filtro de imagem. O bot deve receber uma imagem e retornar a imagem filtrada (use uma técnica de segmentação, como vista no exemplo 3 de agrupamento).
2. Faça um bot de Telegram que classifique imagens;

Dica: em ambos será necessário adaptar o servidor Flask e o fluxo de integração no Node-RED.