REDES DE COMPUTADORES

M.A.X

(M)etereologista (A)utomatizado X

GUSTAVO RODRIGUES

ÍGOR CAPELETTI

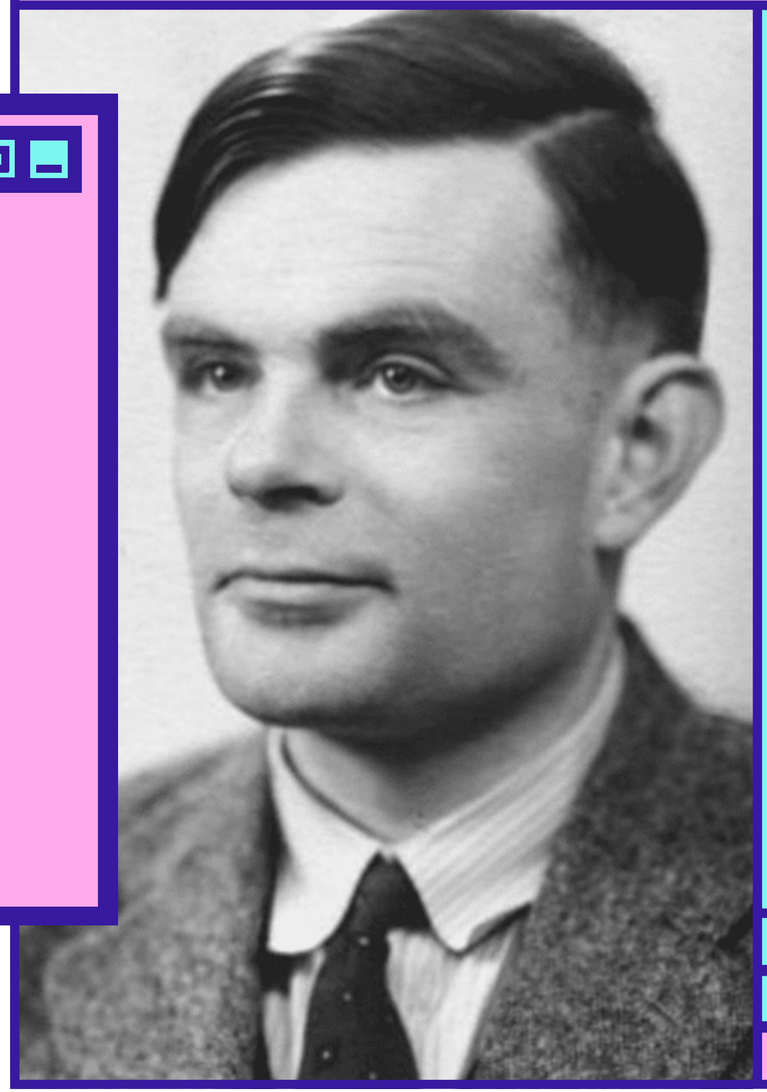
VISÃO GERAL DA APRESENTAÇÃO

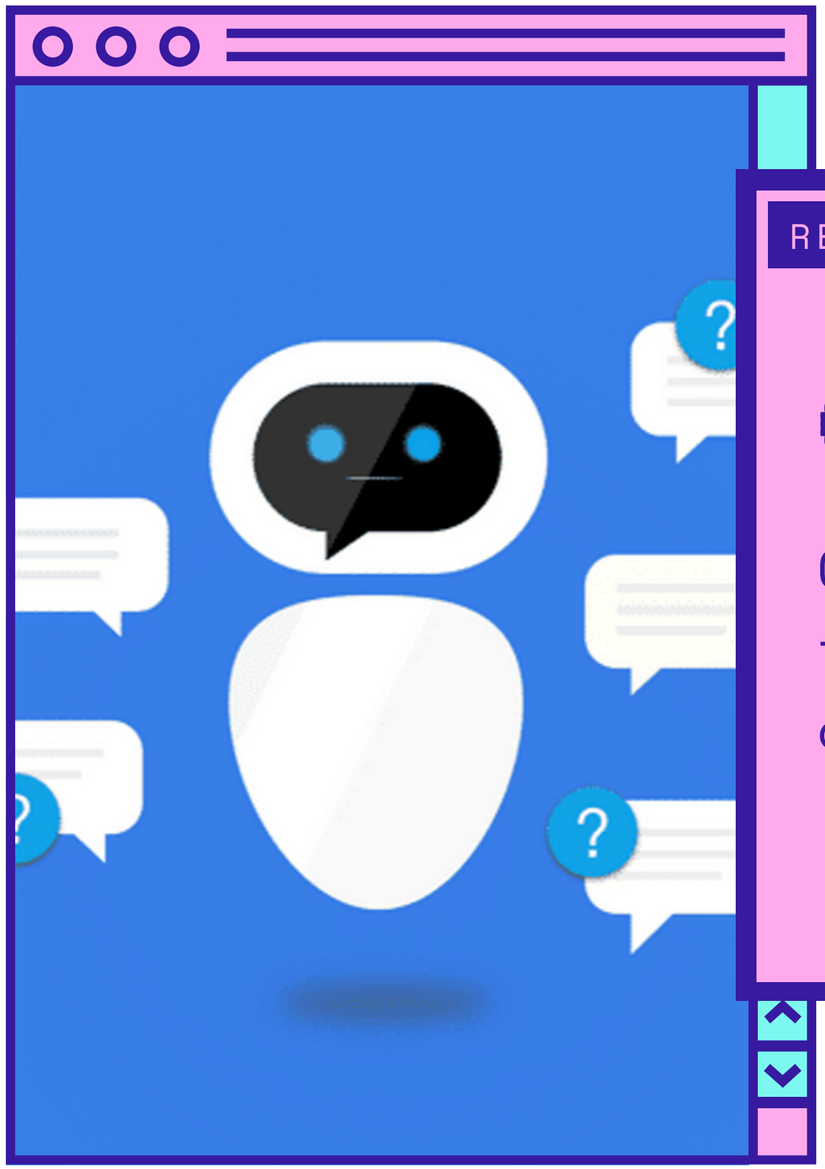
SUMÁRIO

- Contexto histórico
  - Teste de turing
  - Chatbots
- M.A.X
  - Arquitetura
  - Comandos

TESTE DE TURING

Para uma máquina passar no Teste de Turing, ela deve exibir um comportamento inteligente indistinguível do comportamento de um ser humano.





REDES DE COMPUTADORES

CHATTERBOT

Chatbot é um programa de computador que tenta simular um ser humano na conversação com as pessoas.

REDES DE COMPUTADORES

Cortana

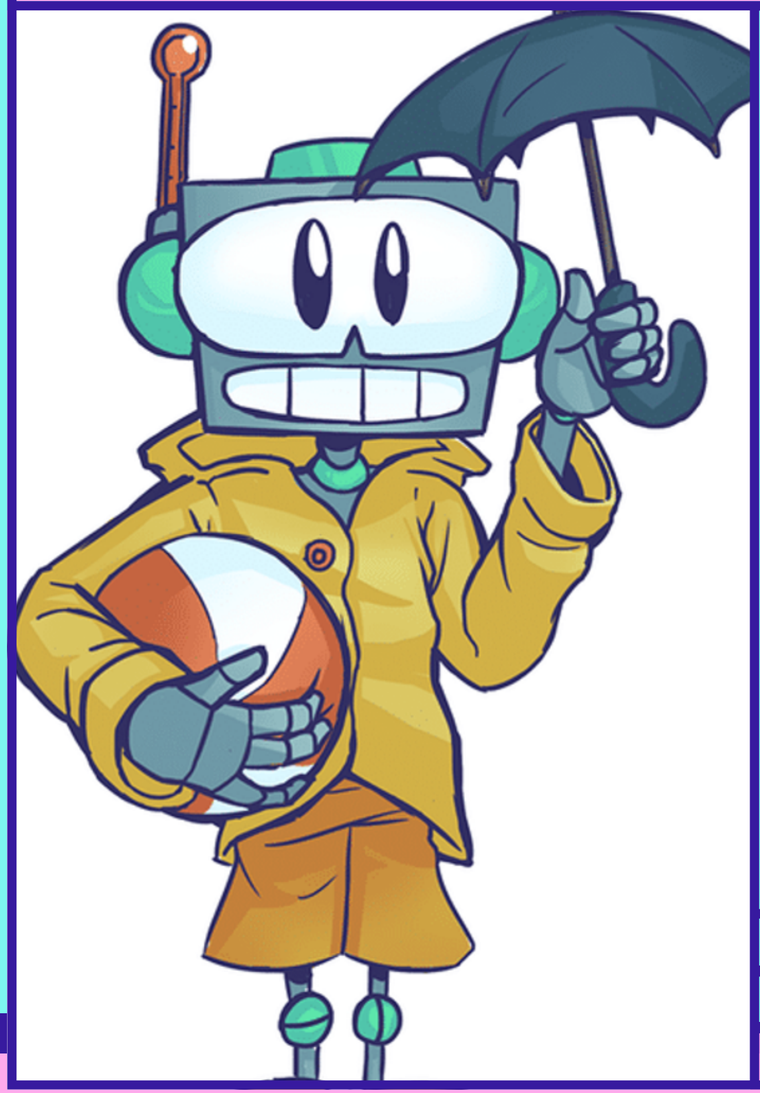
REDES DE COMPUTADORES

amazon alexa

REDES DE COMPUTADORES

M.A.X

chatbot metereologista



OpenWeather

Yandex

AccuWeather

TECNOLOGIAS UTILIZADAS

REDES DE COMPUTADORES

ARQUITETURA

REDES DE COMPUTADORES

\devs

\datahora

\help

\weather

\weatherweek

COMANDOS

REDES DE COMPUTADORES

```
[INFO] Conectado ao servidor 127.0.0.1, porta: 8000, ACK recebido!

 ____    ____       _      ____ ____
|_   \  /   _|     / \    |_  _||_  _|
  |   \/   |      / _ \     \ \  / /
  | |\  /| |     / ___ \     > `' <
 _| |_\/_| |_  _/ /   \ \_  _/ /'`\ \_
|_____||_____||____| |____||____||____|
[HELP] Digite \help para listar os comandos disponiveis do M.A.X

[HELP] Aperte 'q' e de enter para sair

$ \datahora
```

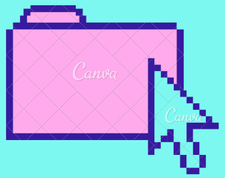
```
[PROCESSANDO ...]

 ____    ____       _      ____ ____
|_   \  /   _|     / \    |_  _||_  _|
  |   \/   |      / _ \     \ \  / /
  | |\  /| |     / ___ \     > `' <
 _| |_\/_| |_  _/ /   \ \_  _/ /'`\ \_
|_____||_____||____| |____||____||____|
[TIME] Data e hora atual: 2019-10-05 01:48:30.303121
--------------------------
```

```
[LOG] Escutando 127.0.0.1:8000...

[LOG] Conectado com cliente 127.0.0.1:43538
[LOG] ACK enviada para cliente!
-------- Cliente 127.0.0.1:43538 ---- Solicitação 1 ----
[INFO] command: DATAHORA
[INFO] Resposta processada e enviada para cliente 127.0.0.1:43538
```

COMANDOS



REDES DE COMPUTADORES

```
[INFO] Conectado ao servidor 127.0.0.1, porta: 8000, ACK recebido!
 ____    ____       _       ____ ____
|_   \  /   _|     / \     |_  _||_  _|
  |   \/   |      / _ \      \ \  / /
  | |\  /| |     / ___ \      > `' <
 _| |_\/_| |_  _/ /   \ \_  _/ /'`\ \_
|_____||_____||____| |____||____||____|
[HELP] Digite \help para listar os comandos disponiveis do M.A.X

[HELP] Aperte 'q' e de enter para sair

$ \weather Alegrete
```

```
[PROCESSANDO ...]
 ____    ____       _       ____ ____
|_   \  /   _|     / \     |_  _||_  _|
  |   \/   |      / _ \      \ \  / /
  | |\  /| |     / ___ \      > `' <
 _| |_\/_| |_  _/ /   \ \_  _/ /'`\ \_
|_____||_____||____| |____||____||____|
[FORECAST] Cidade Alegrete
Status atual: céu nublado nuvens
Previsão de chuva: 0mm
Temperatura atual: 16°C
Umidade em 94%
Velocidade do vento em 20 km/h
------------------------
$
```

```
[LOG] Escutando 127.0.0.1:8000...

[LOG] Conectado com cliente 127.0.0.1:43626
[LOG] ACK enviada para cliente!
-------- Cliente 127.0.0.1:43626 ---- Solicitação 1 ----
comando: 'WEATHER' argumento: 'Alegrete'
[LOG] Pesquisando previsão...
[LOG] Traduzindo...
[INFO] Resposta processada e enviada para cliente 127.0.0.1:43626
```

COMANDOS

REDES DE COMPUTADORES

```
def makeConnection(host, port):
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.connect((host, port))  # conecta com servidor TCP
    data = server.recv(4096).decode()  # recebe resposta de conexão do serv
    if str(data) == "ACK":
        print("[INFO] Conectado ao servidor " + host + ", porta: " + str(po
        clientHandler(server)
        server.close()
        print("\n[INFO] Conexão encerrada com servidor!")
    else:
        print("[ERROR] A conexao com o server " + host + ", porta: " + str(
        server.close()
```

CONEXÃO <CLIENTE>

REDES DE COMPUTADORES

```python
def runServer(ip, port):
    """
    funcao responsavel pela iniclializacao do servidor e de criar conexoes com cada
    :param ip: ip em qual rodara o servidor "localhost"
    :param port: porta em qual o servidor no max rodara
    :return:
    """
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((ip, port))
    server.listen(0)


    print("[LOG] Escutando " + str(ip) + ":" + str(port) + "...\n")  # inicia servi

    while True:
        client, endereco = server.accept()  # aceita conexão dos clientes
        threading.Thread(target=makeConnection, args=(client, endereco)).start()
```

CONEXÃO <SERVIDOR>

REDES DE COMPUTADORES

```
def weather(argument):
    if argument != "":
        print("[LOG] Pesquisando previsão...")
        sockOpenWeather = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # abre conexao com Openweather
        sockOpenWeather.connect(("api.openweathermap.org", 80))  # conecta com a api
        request = ( 'GET /data/2.5/weather?q=' + argument +
                    '&appid=778c214add5c6263ad53043ba7bf546d HTTP/1.1\r\nHost: api.openweathermap.org\r\n\r\n')
        sockOpenWeather.sendall(request.encode())  # envia request GET http para a api
        requisicao = sockOpenWeather.recv(1024).decode()
        sockOpenWeather.close()  # fecha conexao

        status = tempo['weather'][0]['description']     # recebe descrição do tempo em inglês dentro do dict 'tempo'
        qChuva = 0
        if tempo['weather'][0]['main'] == "Rain" and 'rain' in tempo:   # verifica se existe palavra no dict 'tempo' e bu
            qChuva = list(tempo['rain'].values())[0]    #quantidade de chuva previsto

        print("[LOG] Traduzindo...")
        params = dict(key=API_T, text=status, lang='en-pt')     #parametros para requisição com API de tradução
        res = requests.get(urlt, params=params)     #busca com uma API a tradução da previsão do tempo, retornando um jso
        jsonT = res.json()      # transforma para dict o json recebido da API
        status = str(jsonT['text'][0])  #pega string traduzida da API

        answer = "[FORECAST] Cidade " + argument + "\nStatus atual: " + status + "\nPrevisão de chuva: " + str(
            qChuva) + "mm\nTemperatura atual: " + str(round((float(tempo['main']['temp'])) - 273.15)) +
            "°C\nUmidade em " + str(tempo['main']['humidity']) + "%\nVelocidade do vento em " +
            str(round((float(tempo['wind']['speed'])) * 3.6)) + " km/h"
    else:
        answer = "[ERROR] Comando \weather necessita de uma localizacao

    return answer
```

CONEXÃO
<API OPENWEATHHER>

REDES DE COMPUTADORES

```
def weatherWeek(argument):
    if argument != "":
        print("[LOG] Cidade " + argument + " recebida do cliente e ")
        city = urllib.parse.quote(str(argument))  # transforma string da cidade para formato URL p/ API reconhecer
        print("[LOG] Pesquisando previsão...")

        sockAccuWeather = socket.socket(socket.AF_INET,socket.SOCK_STREAM)  # abre conexao TCP com o servidor do AccuWeather
        sockAccuWeather.connect(("dataservice.accuweather.com", 80))  # conecta com a API AccuWeather
        request = ('GET /locations/v1/cities/search?apikey=' + API +
                   '&q=' + city + '&details=true HTTP/1.1\r\nHost: dataservice.accuweather.com\r\n\r\n')

        sockAccuWeather.sendall(request.encode('utf-8'))  # envia request GET http para a api
        requisicao = sockAccuWeather.recv(2048).decode()  # recebe string com resposta do servidor
        sockAccuWeather.close()  # fecha conexao com API AccuWeather

        sockAccuWeather = socket.socket(socket.AF_INET,socket.SOCK_STREAM)  # abre conexao  com o servidor do AccuWeather
        sockAccuWeather.connect(("dataservice.accuweather.com", 80))  # conecta com a api

        request = ('GET /forecasts/v1/daily/5day/' + location_key + '?apikey=' + API +
                   '&details=true HTTP/1.1\r\nHost: dataservice.accuweather.com\r\n\r\n')

        sockAccuWeather.sendall(request.encode('utf-8'))  # envia request GET http para a api
        requisicao2 = sockAccuWeather.recv(2048).decode('utf-8')  # recebe string com resposta do servidor
        requisicao2 = requisicao2 + sockAccuWeather.recv(2048).decode('utf-8')  # acrescenta resposta anterior com esta nova
                                    ...
        requisicao2 = requisicao2 + sockAccuWeather.recv(2048).decode('utf-8')  # acrescenta resposta anterior com esta nova
        sockAccuWeather.close()  # fecha conexao com API AccuWeather

        for key1 in tempo['DailyForecasts']:
            status = key1['Day']['LongPhrase']
            respStatus.insert(cont, str(status))
            cont = cont + 1
            status = key1['Night']['LongPhrase']
            respStatus.insert(cont, str(status))
            cont = cont + 1

        params = dict(key=API_T, text=str(respStatus),lang='en-pt')  # envia list anterior em
        res = requests.get(urlt, params=params)  # recebe json da API com as palavras já trad
        jsonT = res.json()  # transforma arquivo json recebido em dict
        status = str(jsonT['text'][0])
```

CONEXÃO
<API ACCUWEATHER>

REDES DE COMPUTADORES

GUSTAVO RODRIGUES

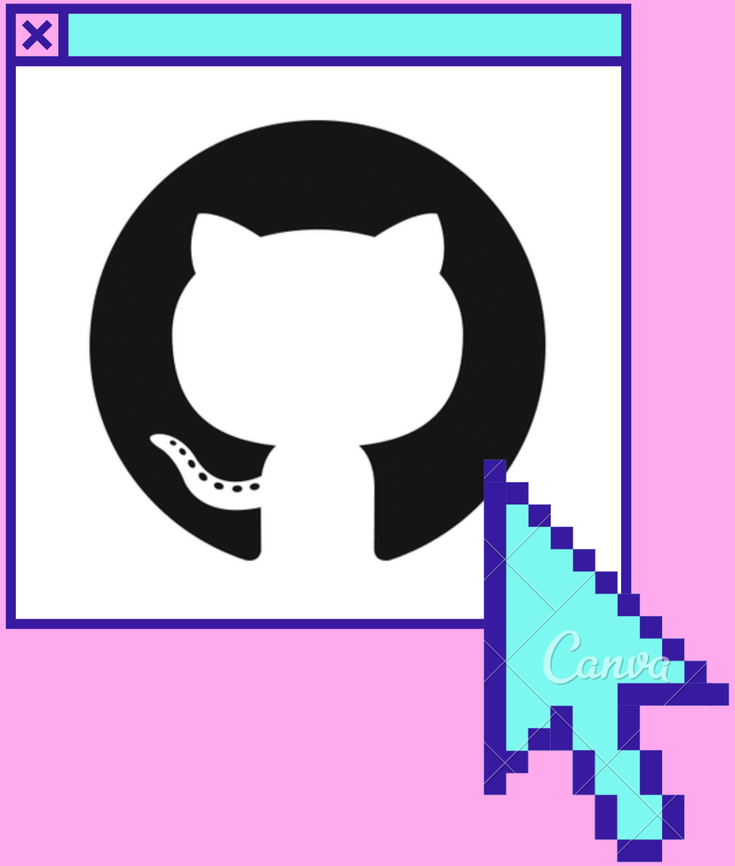
Programador

gustavorodrigues.aluni@unipampa.edu.br

IGOR CAPELETTI

Programador

igor-capeletti@hotmail.com



ACESSE NO GITHUB

SCAN ME

github.com/gustavocrod/maxbot