



IA GENERATIVA

GRUPO 8 - BRUNO TANABE
RAFAEL FORTES

Índice

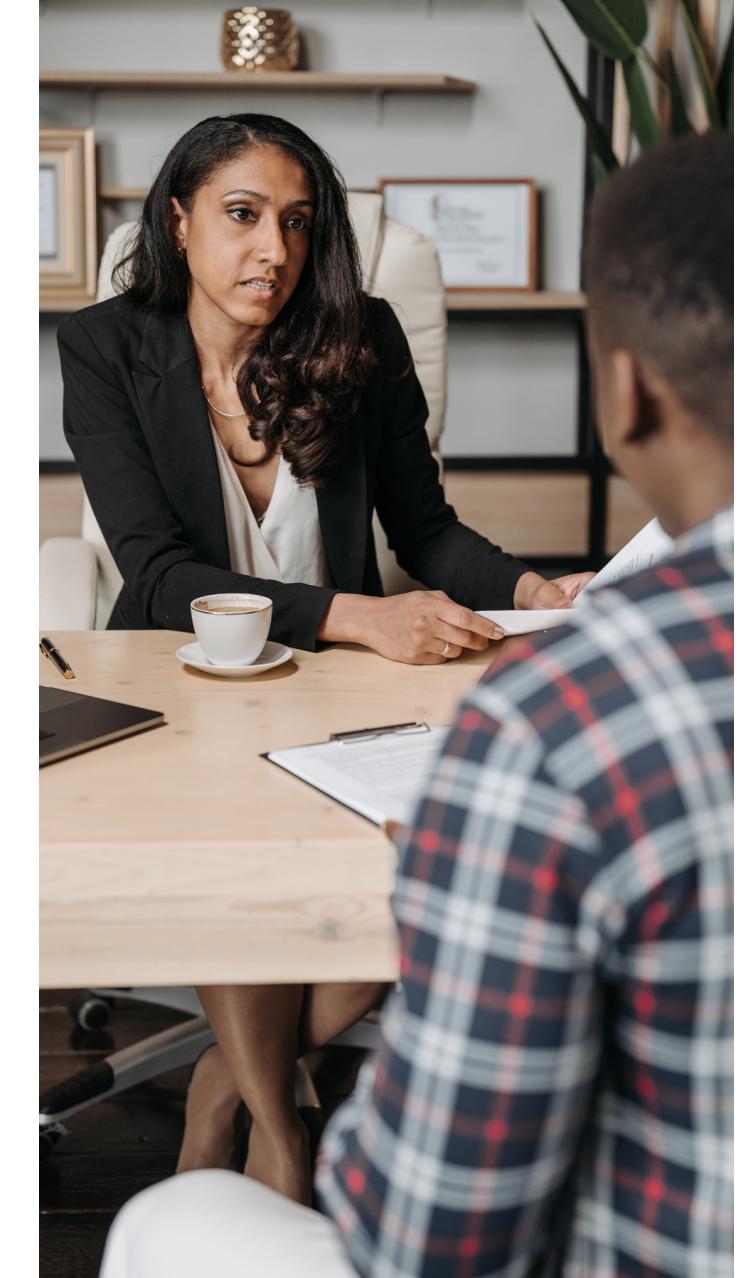
- A IMPORTÂNCIA DO SUPORTE
- O DESAFIO DE FAZER UM SUPORTE EFICIENTE E PRECISO
- NOSSA SOLUÇÃO
- DEMONSTRAÇÃO
- COMO A SOLUÇÃO FOI CONSTRUÍDA?
- TECNOLOGIAS UTILIZADAS
- NOSSO CÓDIGO
- WORD EMBEDDINGS
- PROBLEMAS E DESAFIOS NO DESENVOLVIMENTO
- IMPACTOS E VANTAGENS
- AGRADECIMENTO ESPECIAL
- DÚVIDAS?



TECH4HUMANS & CEU

A importância do suporte ao cliente

- O que isso significa?
- Por que é importante?
- Pesquisa da Opinium
- Nossa motivação



[VOLTAR PARA O ÍNDICE](#)



O desafio: Fazer um Suporte Eficiente e Preciso

- Crescentes demandas
- Diversidade de Problemas
- Nossa solução



Nossa Solução: IA no Atendimento

FAQ:

- Pergunta:** A empresa oferece descontos para pacotes combinados?
- Resposta:** Sim, oferecemos descontos especiais para clientes que optam por contratar mais de um serviço da nossa empresa.

Similaridade: 100.00%

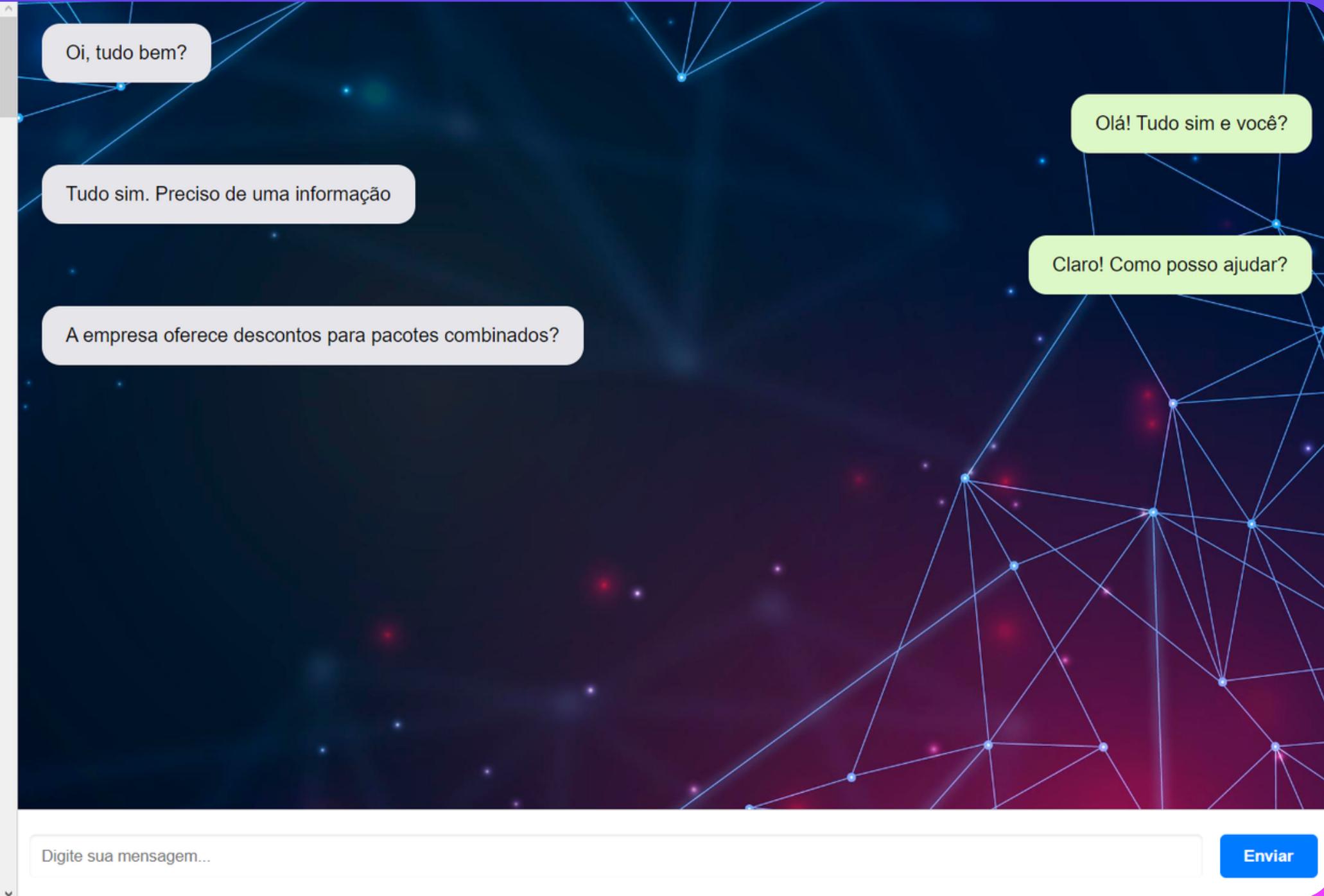
- Pergunta:** A empresa oferece planos empresariais?
- Resposta:** Sim, temos planos especiais para empresas, com benefícios e condições exclusivas para atender às necessidades corporativas.

Similaridade: 73.78%

- Pergunta:** A empresa oferece suporte técnico?
- Resposta:** Sim, nossa empresa oferece suporte técnico especializado para auxiliá-lo em questões relacionadas aos nossos serviços.

Similaridade: 71.85%

Respostas: A empresa oferece



Chat interface screenshot:

- User: Oi, tudo bem?
- AI: Olá! Tudo sim e você?
- User: Tudo sim. Preciso de uma informação
- AI: Claro! Como posso ajudar?
- User: A empresa oferece descontos para pacotes combinados?

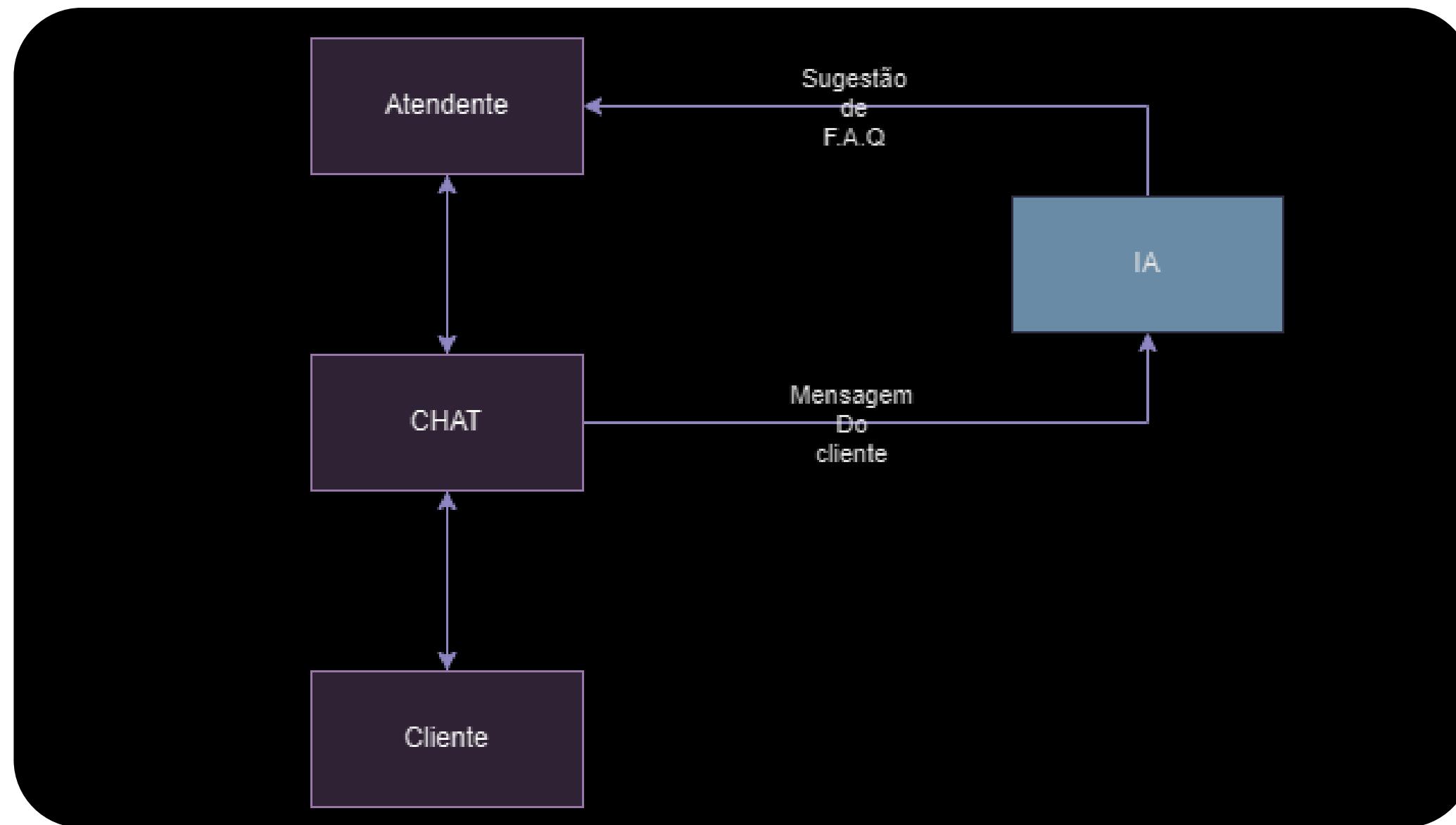
Message input field: Digite sua mensagem... Enviar

[VOLTAR PARA O ÍNDICE](#)

DEMONSTRAÇÃO DO FUNCIONAMENTO

[VOLTAR PARA O ÍNDICE](#)

Como a solução foi construída?



[VOLTAR PARA O ÍNDICE](#)

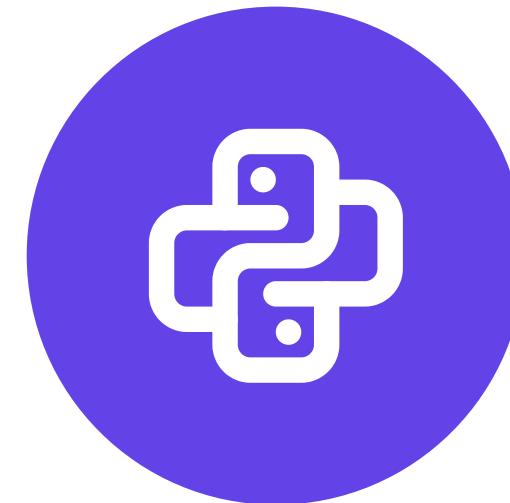
Tecnologias utilizadas no Front-End:



HTML



CSS



PYTHON

Código: HTML

```
● ● ●  
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4   <title>Chat de Atendimento</title>  
5   <!-- Incluindo o arquivo de estilo CSS -->  
6   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">  
7   <!-- Adicionando o link do favicon -->  
8   <link rel="icon" href="{{ url_for('static', filename='favicon.png') }}" type="image/png">  
9 </head>
```

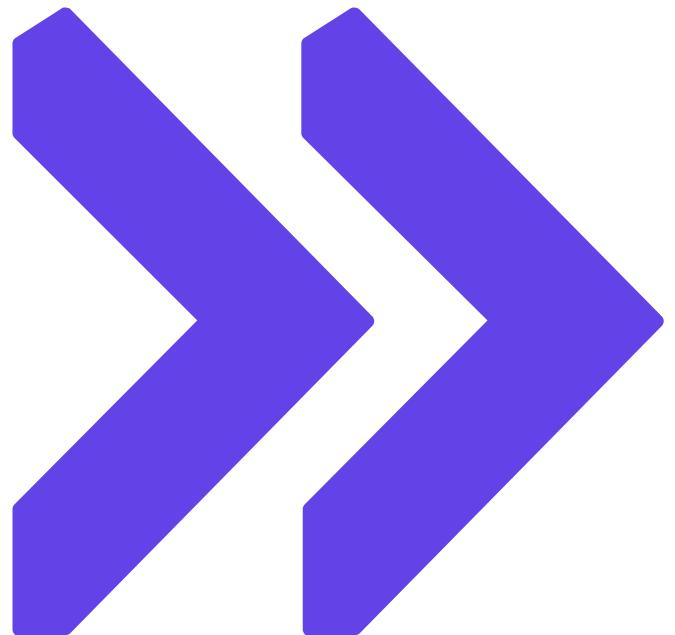


```
1 <body>
2     <!-- Container principal para o chat -->
3     <div class="chat-container">
4         <!-- Barra lateral do chat -->
5         <div class="chat-sidebar">
6             <!-- Lista de contatos -->
7             <div class="contact-list">
8                 <!-- Verifica se há respostas sugeridas -->
9                 {% if respostas_sugeridas %}
10                 <!-- Caso não haja respostas sugeridas -->
11                 <h2>FAQ:</h2>
12                 <ul>
13                     <!-- Loop através das respostas sugeridas -->
14                     {% for similaridade, pergunta, resposta in respostas_sugeridas %}
15                         <li>
16                             <!-- Exibe a pergunta -->
17                             <p class="pergunta"><strong>Pergunta:</strong> {{ pergunta }}</p>
18                             <!-- Exibe a resposta -->
19                             <p class="resposta"><strong>Resposta:</strong> {{ resposta }}</p>
20                             <!-- Exibe a similaridade -->
21                             <p class="similaridade"><strong>Similaridade:</strong> {{ "%.2f" % (similaridade * 100) }}%</p>
22                         </li>
23                     {% endfor %}
24                 </ul>
25                 {% else %}
26                     <!-- Caso não haja respostas sugeridas -->
27                     <h2>FAQ:</h2>
28                     <p>Nenhuma resposta sugerida no momento.</p>
29                 {% endif %}
30             </div>
31         </div>
```



```
1  <!-- Área principal do chat -->
2      <div class="chat-main">
3          <div class="chat-history">
4              <!-- Exibe a conversa entre o atendente e o usuário -->
5              {% for mensagem, remetente in conversa %}
6                  {% if remetente == 'usuario' %}
7                      <div class="message from-usuario">
8                          <div class="message-bubble usuario-bubble">
9                              {{ mensagem }}
10                         </div>
11                     </div>
12                     {% endif %}
13                     {% if remetente == 'atendente' %}
14                         <div class="message from-atendente">
15                             <div class="message-bubble atendente-bubble">
16                                 {{ mensagem }}
17                             </div>
18                         </div>
19                         {% endif %}
20                     {% endfor %}
21                 </div>
22             <!-- Área de entrada de texto -->
23             <div class="chat-input">
24                 <form method="post">
25                     <!-- Campo de entrada de texto -->
26                     <input type="text" id="mensagem" name="mensagem" placeholder="Digite sua mensagem..." required>
27                     <!-- Botão de envio -->
28                     <button type="submit">Enviar</button>
29                 </form>
30             </div>
31         </div>
32     </div>
33 </body>
34 </html>
```

Código: Python (Conexão)



[VOLTAR PARA O ÍNDICE](#)

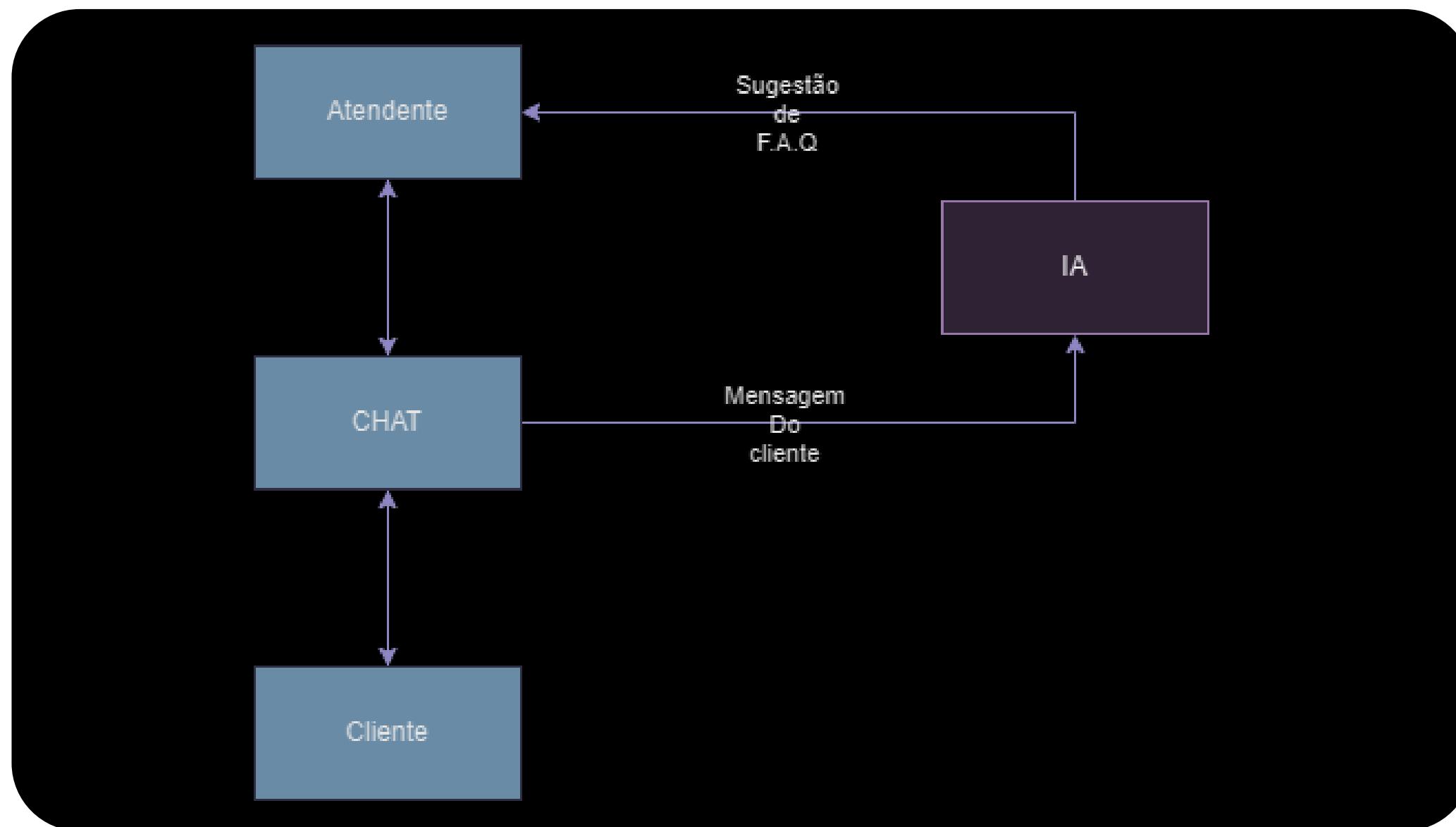
```
● ● ●

 1  from flask import Flask, render_template, request
 2  import requests
 3
 4  app = Flask(__name__)
 5
 6  # Função para obter respostas sugeridas com base na mensagem do cliente
 7  def obter_respostas_sugeridas(mensagem_do_cliente):
 8      url_da_api = f"http://127.0.0.1:8000/get_similarities/{mensagem_do_cliente}"
 9      try:
10          # Faz uma requisição à API local
11          resposta = requests.get(url_da_api)
12          if resposta.status_code == 200:
13              resposta_json = resposta.json()
14              similaridades = resposta_json["Similarity"]
15              respostas_sugeridas = []
16
17              # Filtra respostas sugeridas com similaridade maior que 0.5
18              for indice, similaridade in similaridades.items():
19                  pergunta = resposta_json["Question"][indice]
20                  resposta = resposta_json["Answer"][indice]
21                  if similaridade < 0:
22                      similaridade = 0
23                  respostas_sugeridas.append((similaridade, pergunta, resposta))
24
25              # Ordena a lista de respostas sugeridas com base nas similaridades (do maior para o menor)
26              respostas_sugeridas.sort(reverse=True)
27
28              return respostas_sugeridas
29
30      else:
31          return []
32
33  except requests.exceptions.RequestException:
34      return []
```

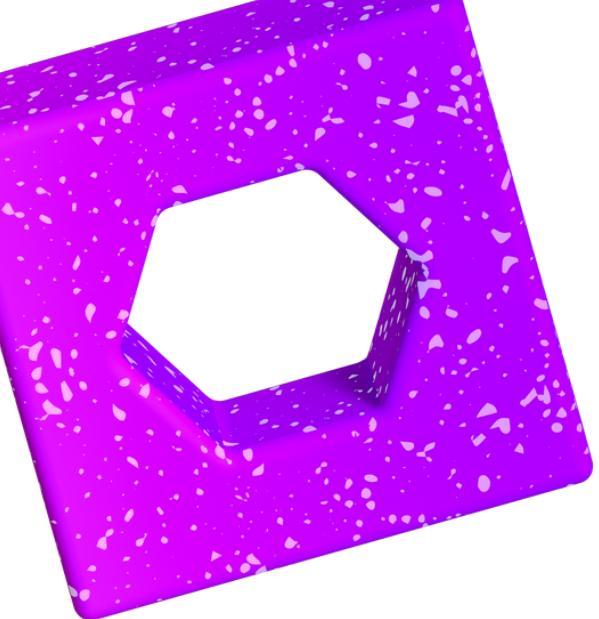
```
● ● ●

1 # Rota principal do aplicativo
2 @app.route('/', methods=['GET', 'POST'])
3 def index():
4     respostas_sugeridas = []
5     conversa = [
6         ("Olá! Tudo bem?", "usuario"),
7         ("Olá! Tudo sim e você?", "atendente"),
8         ("Tudo sim. Preciso de uma informação", "usuario"),
9         ("Claro! Como posso ajudar?", "atendente"),
10    ]
11
12    if request.method == 'POST':
13        # Obtém a mensagem do cliente do formulário
14        mensagem_do_cliente = request.form['mensagem']
15        # Chama a função para obter respostas sugeridas
16        respostas_sugeridas = obter_respostas_sugeridas(mensagem_do_cliente)
17        # Adiciona a mensagem do cliente à conversa
18        conversa.append((mensagem_do_cliente, "usuario"))
19
20    # Se a conversa estiver vazia, adiciona mensagens do atendente previamente criadas
21    if not conversa:
22        mensagens_atendente = [
23            ("Olá! Tudo sim e você?", "atendente"),
24            ("Claro! Como posso ajudar?", "atendente"),
25        ]
26        conversa.extend(mensagens_atendente)
27
28    # Renderiza o template HTML com as respostas sugeridas e a conversa
29    return render_template('index.html', respostas_sugeridas=respostas_sugeridas, conversa=conversa)
30
31 # Executa o aplicativo em modo de depuração
32 if __name__ == '__main__':
33     app.run(debug=True)
```

Como a solução foi construída?



[VOLTAR PARA O ÍNDICE](#)



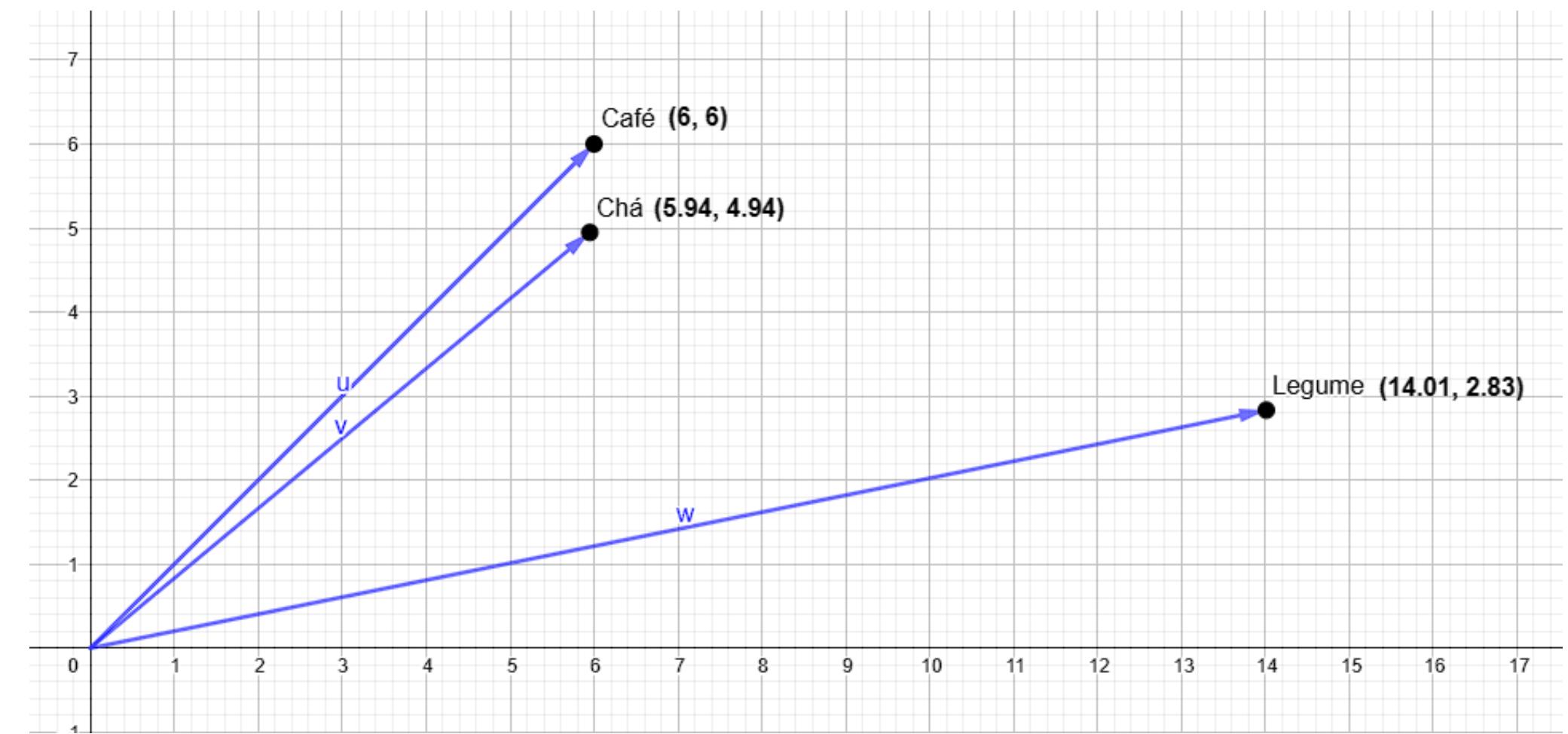
Tecnologias utilizadas na construção da IA:

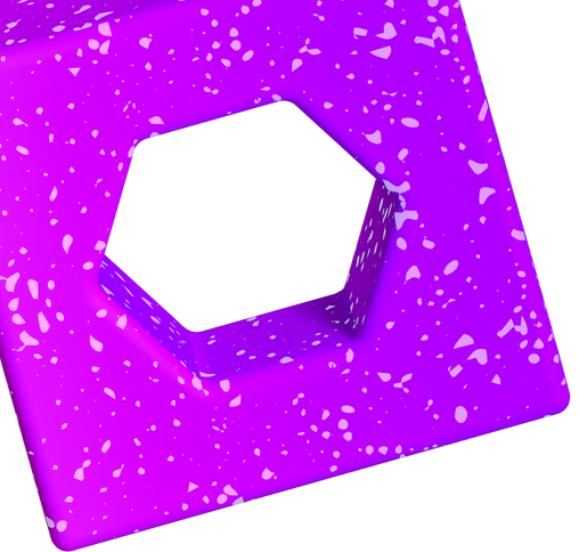
- Python
- Pandas
- Spacy
- SentenceTransformer
- FastAPI



Word Embeddings

- Representação vetorial de textos
- Possibilita a busca semântica
- BERTimbau - UFRGS





```
1 df = pd.read_csv("data/raw/FAQ_example.csv", sep=";")  
2 df.head(3)
```

	Question	Answer
0	Quais serviços a empresa de telecomunicações o...	Nossa empresa oferece serviços de telefonia fi...
1	Como posso contratar os serviços da empresa?	Para contratar nossos serviços, você pode entr...
2	Quais são os planos disponíveis para a telefon...	Temos diversos planos com diferentes franquias...

```
● ● ●
1 def get_sentence_embeddings(model, series):
2     for i, text in enumerate(series):
3         embed = model.encode(get_clean_text(text))
4
5         if i == 0:
6             text_embeddings = pd.DataFrame({f"{i}": embed})
7         else:
8             text_embeddings[f"{i}"] = embed
9
10    return text_embeddings
```

```
1 df_embedding = get_sentence_embeddings(embedding_model, df.Question)
2 df_embedding.head()
```

	0	1	2	3	4	5	6	7	8	9	...
0	-0.188625	0.400796	0.411759	0.109735	0.009841	0.034627	0.166421	0.056301	0.097398	-0.092919	...
1	0.669775	-0.513453	0.729795	0.126028	0.267572	0.481285	1.141499	0.142642	0.742798	0.549269	...
2	-0.381557	-0.054400	-0.294071	0.117846	-0.608610	-0.065050	-0.305382	0.236780	-0.003630	0.204660	...
3	0.454016	0.232534	0.188410	-0.255084	0.524474	0.415265	-0.067251	0.152116	0.100762	0.834601	...
4	0.066648	-0.130414	0.453748	-0.097489	-0.049003	-0.478443	0.460942	0.129103	0.424205	-0.115655	...

API

```
● ● ●  
1 from fastapi import FastAPI  
2  
3 import pandas as pd  
4 import numpy as np  
5 import spacy  
6 from sentence_transformers import SentenceTransformer, util, models  
7  
8 from features import functions
```

```
● ● ●  
1 word_embedding_model = models.Transformer("models/bert-large-portuguese-cased")  
2 pooling_model = models.Pooling(word_embedding_model.get_word_embedding_dimension())  
3 embed_model = SentenceTransformer(modules=[word_embedding_model, pooling_model])  
4  
5 nlp_model = spacy.load("pt_core_news_sm")  
6  
7 embeddings = pd.read_csv("data/Embeddings.csv", sep=";")  
8 faq = pd.read_csv("data/FAQ_example.csv", sep=";")
```

```
1 def get_clean_text(model, text:str):
2     text_tokens = model(text)
3
4     clean_text = []
5
6     for token in text_tokens:
7         if not token.is_stop and not token.is_punct:
8             clean_text.append(token.text)
9
10    clean_text = " ".join(clean_text)
11
12    return re.sub(r'[^a-zA-Zà-ÿ\s]', '', clean_text).lower()
```

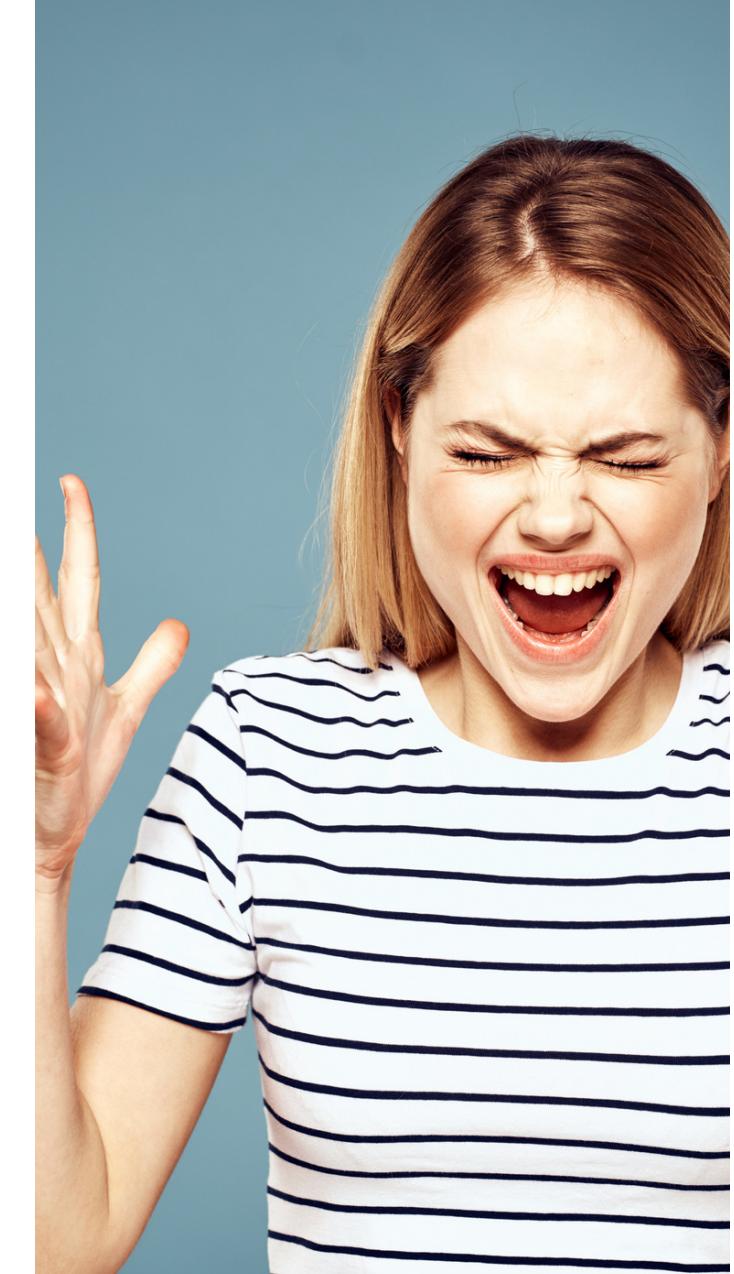
```
1 def get_similarity(model, embeddings, text):
2     text_embed = model.encode(text)
3     similarities = []
4
5     for column in embeddings.columns:
6         question_embedding = embeddings[column]
7         similarity = util.cos_sim(text_embed, question_embedding.values.astype(np.float32))
8         similarities.append(similarity)
9
10
11    return pd.Series(similarities).astype(float)
```

```
● ● ●  
1 app = FastAPI()  
2  
3  
4 @app.get("/")  
5 def home():  
6     return {"Status": "ok"}  
7  
8  
9 @app.get("/get_similarities/{text}")  
10 def get_similarities(text: str):  
11     text_preprocessed = functions.get_clean_text(nlp_model, text)  
12     similarities = functions.get_similarity(embed_model, embeddings, text_preprocessed)  
13  
14     response = faq.copy()  
15     response["Similarity"] = similarities.values  
16     response.sort_values(by="Similarity", ascending=False, inplace=True)  
17     response.index = [i for i in range(response.shape[0])]  
18  
19     return response
```



Problemas e desafios no desenvolvimento:

- Garantir a precisão das respostas sugeridas pela IA
- Avaliação do modelo





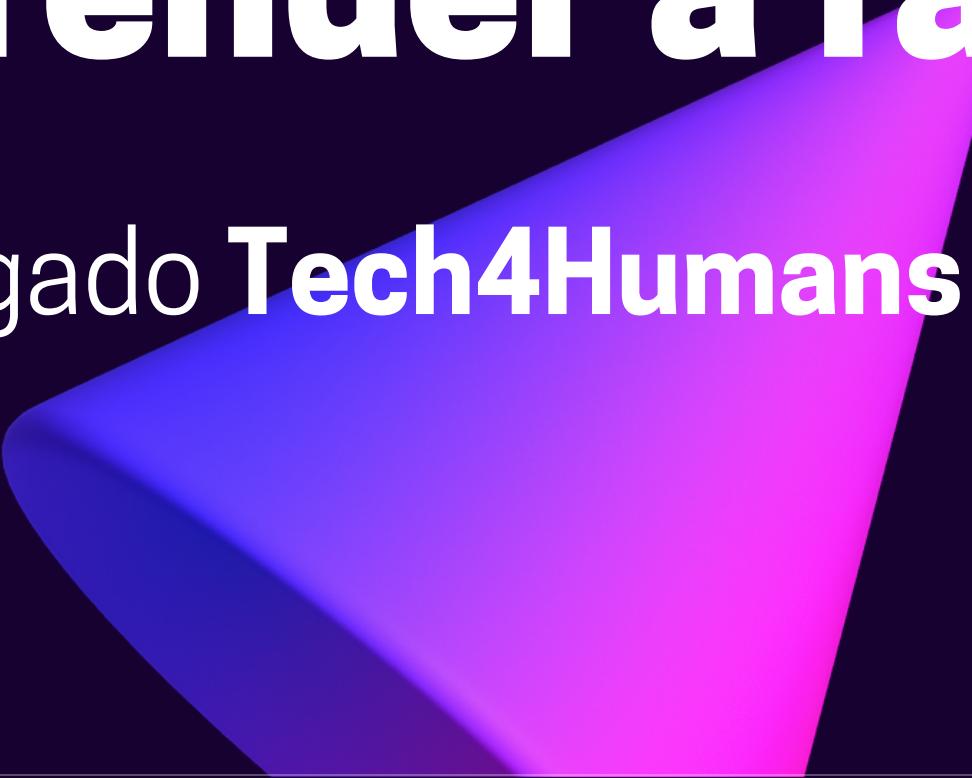
Impacto e Vantagens

- Aumento na **satisfação do atendente**
- Melhora na **experiência do cliente**
- **Diferenciação no Mercado**





**É fazendo que se aprende a
fazer aquilo que se deve
aprender a fazer.**



Obrigado **Tech4Humans & CEU!**

Dúvidas?

Fale com a gente!



Bruno Tanabe

LinkedIn: Bruno Tanabe
GitHub: @brunotanabe



Rafael Fortes

LinkedIn: Rafael Fortes
GitHub: @rafael-fortes