Eixo Temático: Fábrica de Software: Desenvolvimento Web, Desktop e Mobile

USO DO FRAMEWORK FLASK PARA DESENVOLVIMENTO DE CHATBOTS EM PYTHON

Sthefany T. S. Novaes [1]; Erick B. Nascimeneto[2] e Mirthys Marinho[3].

INTRODUÇÃO

A crescente presença de *chatbots* em interfaces digitais reflete a demanda por soluções mais ágeis, escaláveis e interativas no relacionamento entre sistemas e usuários. Seja em atendimentos automatizados, sistemas educacionais ou plataformas de serviço. Essas interfaces conversacionais têm se mostrado eficientes em otimizar o tempo de resposta e personalizar a experiência do usuário. No entanto, os desafios persistem, especialmente relacionados à escolha de ferramentas que aliem simplicidade, flexibilidade e eficiência, bem como à integração adequada com outras tecnologias *Web* para construção de interfaces acessíveis e funcionais (DOBBALA; LINGOLU, 2024).

Nesse cenário, o *Flask*, um Microframework escrito para a linguagem *Python*, apresenta-se como uma alternativa viável e robusta para o desenvolvimento de aplicações *Web* que envolvem *chatbots*. Por conta de sua estrutura simplificada e flexível, com gerenciamento eficiente de rotas, manipulação de dados e integração com APIs, *Flask* permite que desenvolvedores criem soluções personalizadas sem a complexidade e rigidez de *frameworks* que consomem muitos recursos (GRINBERG, 2018). Dessa forma, é possível estruturar a lógica de interação do sistema de modo fluido, mantendo a aplicação leve e de fácil manutenção.

Portanto, o presente artigo traz uma abordagem conceitual sobre o *Flask Framework* e sua aplicação em sistemas interativos, apresentando os elementos técnicos para integração de dados e lógica de comunicação com o usuário final. São abordadas as possibilidades de

[1] Discente do Curso de Bacharelado em Sistemas de Informação (UNIRIOS) - sthefany.novaes13@gmail.com
[2] Mestre em Ciências da Computação (UFS), Docente do Bacharelado em Sistemas de Infromação (UniRios) - erick.nascimento@unirios.edu.br
[3] Mestre em Desenvolvimento de Processos Ambientais (UNICAP), Docente do Bacharelado em Sistemas de Informação (UniRios) - e-mail: mirthys.melo@unirios.edu.br

construção de interfaces com foco na experiência do usuário, termo em inglês conhecido com *User Experience* (UX), reforçando a importância do *Front-end* no desenvolvimento de soluções conversacionais eficazes e acessíveis, ou seja, um sistema de *software* que interage com o usuário por meio de linguagem natural, geralmente através de texto ou voz, simulando uma conversa humana (MENDES, 2022).

OBJETIVO

O presente estudo tem por objetivo analisar a aplicabilidade do *Framework Flask* no desenvolvimento de *chatbots* em *Python* voltados para aplicações *Web*, considerando os seus aspectos técnicos, a sua arquitetura e a integração com tecnologias complementares. Busca-se, ainda, examinar as principais funcionalidades do *framework*, destacando a relevância do gerenciamento de rotas e da manipulação de dados para a construção de uma lógica conversacional funcional e escalável, bem como investigar a contribuição de tecnologias de *front-end*, como HTML, CSS e *JavaScript*, para o aprimoramento da interface e da experiência do usuário.

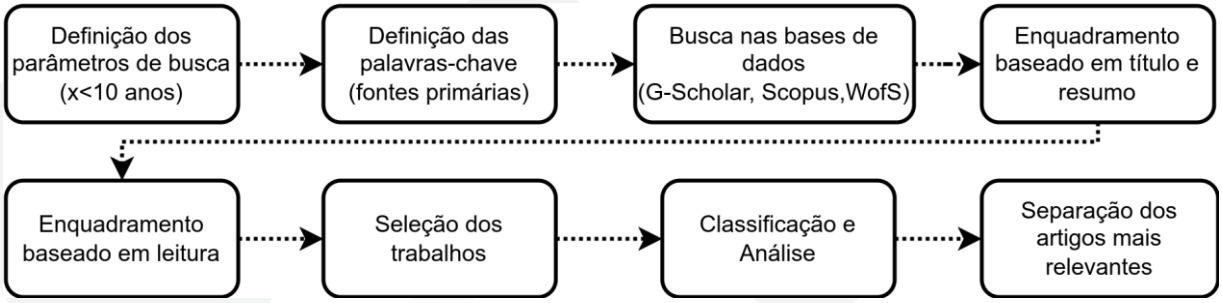
METODOLOGIA

Para esta pesquisa, foram desenvolvidos um conjunto de macro-atividades dividindo os componentes da metodologia em tarefas posteriores para auxiliar a visualização e agilizar o processo investigativo. Logo, quanto à abordagem essa pesquisa se caracteriza como descritiva-exploratória, porque busca analisar um determinado conceito aprofundando-se sobre o mesmo. Para isso, foram utilizados procedimentos descritos em tutorias mãos-a-obra em sites de projeto, além de consultas bibliográficas, tais como: *Flask framework cookbook* (2015), de Shalabh Aggarwal; *Curso Intensivo de Python: uma introdução prática e baseada em projetos à programação* (2016), de Eric Matthes, para entermos os pontos-chave do projeto de concepção do sistema a ser desenvolvido como experimento, fazendo uma análise qualitativa do sistema.

Portanto, de acordo com Gil (2017, p.41), a pesquisa descritiva-exploratória tem como objetivo descrever e explorar um determinado fenômeno, buscando aprofundar o conhecimento sobre ele.

Já a pesquisa qualitativa se concentra na coleta e análise de dados não numéricos, como entrevistas, observações e documentos codificados, para compreender as nuances e os significados por trás dos dados. Dessa forma, para obtenção dos dados fonte que compõe esse trabalho, o seguinte fluxo foi executado como mostra a Figura 1:

Figura 1: Fluxograma das etapas do processo de seleção dos artigos.



Fonte: Autores, 2025

RESULTADOS E DISCUSSÕES

Nesta seção, apresentamos a implementação do *Chatbot Web*, combinando o *Framework Flask* no *backend,* utilizando tecnologias modernas de *front-end*, como HTML, *Tailwind* CSS e *JavaScript*, para a construção de uma interface interativa, responsiva e funcional. O objetivo principal foi validar a viabilidade técnica e a eficiência da solução proposta para oferecer uma experiência conversacional fluida e adaptada a aplicações *web*. A Figura 2, apresenta o sistema de *backend*, intitulado “app.py”. Na linha 1, são importadas as bibliotecas necessárias para utilização do *Flask*, bem como, renderizador páginas HTML e conversor de formatos JSON.

Figura 2: Código de aplicação *backend* utilizando o Framework Flask Python.

app.py

```python
from flask import Flask, render_template, request, jsonify
import google.generativeai as genai

app = Flask(__name__)

# Configure sua chave API do Google Generative AI
genai.configure(api_key="SUA_CHAVE_AQUI")
model = genai.GenerativeModel('gemini-pro')

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/send", methods=["POST"])
def send():
    user_input = request.json.get("mensagem", "")
    try:
        prompt = f"""
Você é um assistente virtual educado e objetivo. Responda à seguinte mensagem em português:

Mensagem: "{user_input}"
"""
        response = model.generate_content(prompt)
        resposta_limpa = response.text.strip() if response.text else "Não consegui responder."
        return jsonify({"resposta": resposta_limpa})
    except Exception as e:
        return jsonify({"resposta": f"Erro: {str(e)}"}), 500

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

Fonte: Autores, 2025

A aplicação conta com duas rotas principais: a raiz ("/"), na linha 10, responsável por servir a página inicial do *chatbot*, e a rota "/send", declarada na linha 14, configurada para receber requisições do tipo POST contendo as mensagens do usuário. Ao receber uma mensagem, o servidor processa o conteúdo por meio da função “generate_content()”, na linha 23, que neste protótipo, simula o funcionamento de um modelo de linguagem natural, retornando uma resposta pré-definida com base no texto recebido.

Esse recurso é possível, devido a declaração da variável “model”, na linha 8. Essa declaração model = google.generativeai(“gemini-pro”) é um comando que cria e configura uma instância do modelo de inteligência artificial Gemini Pro para ser usada no sistema, onde, “google.generativeai” é o nome da biblioteca ou módulo que fornece acesso à API de IA Generativa do Google; e “gemini-pro” é o argumento passado para a função.

Essa abordagem demonstra como o *Flask* pode facilmente ser integrado a mecanismos mais sofisticados de Processamento de Linguagem Natural (PLN) ou APIs externas para Inteligência Artificial, garantindo escalabilidade futura, permitindo implementar *endpoints* RESTful com poucas linhas de código, otimizando o desenvolvimento e facilitando a manutenção.

A interface do usuário (UI) da tela do *chatbot* foi desenvolvida com HTML semântico para garantir acessibilidade e uma boa estruturação do conteúdo. Para a estilização, foi utilizada a biblioteca Tailwind CSS via CDN, que oferece um conjunto de classes utilitárias. Isso possibilitou a criação rápida de um layout responsivo e moderno, sem a necessidade de um arquivo CSS extenso e customizado. Veja a Figura 3.

Figura 3: Código de aplicação *frontend* utilizando o HTML com Tailwind CSS.

index.html

templates > index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8" />
    <title>Chatbot com Flask</title>
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet" />
</head>
<body class="bg-gray-100 flex items-center justify-center h-screen">
    <div class="bg-white rounded-lg shadow-lg w-full max-w-md p-6 flex flex-col">
        <h1 class="text-2xl font-bold mb-4 text-center">Chatbot Flask</h1>
        <div id="chat" class="flex-grow overflow-auto mb-4 border p-4 rounded-lg bg-gray-50"></div>
        <form id="chat-form" class="flex">
            <input id="message-input" type="text" placeholder="Digite sua mensagem..." class="flex-grow border rounded-l px-4 py-2 focus:outline-none" required />
            <button type="submit" class="bg-blue-600 text-white px-4 rounded-r hover:bg-blue-700 transition">Enviar</button>
        </form>
    </div>

    <script>
        const chat = document.getElementById('chat');
        const form = document.getElementById('chat-form');
        const input = document.getElementById('message-input');

        form.addEventListener('submit', async (e) => {
            e.preventDefault();
            const userMessage = input.value.trim();
            if (!userMessage) return;

            appendMessage('Você', userMessage);
            input.value = '';

            try {
                const response = await axios.post('/send', { message: userMessage });
                appendMessage('Bot', response.data.response);
                chat.scrollTop = chat.scrollHeight;
            } catch (error) {
                appendMessage('Bot', 'Erro ao se comunicar com o servidor.');
            }
        });

        function appendMessage(sender, text) {
            const messageElement = document.createElement('div');
            messageElement.classList.add('mb-2');
            messageElement.innerHTML = `<strong>${sender}:</strong> ${text}`;
            chat.appendChild(messageElement);
        }
    </script>
</body>
</html>
```

Fonte: Autores, 2025

Para proporcionar uma experiência fluida e dinâmica, o *frontend* do chatbot utiliza "*JavaScript*" com a biblioteca "*Axios*" para realizar requisições assíncronas ao *backend*. Essa tecnologia é crucial numa aplicação conversacional, pois permite o envio e recebimento de mensagens em tempo real, sem recarregar a página. A interface ainda exibe as mensagens de forma sequencial e inclui rolagem automática, garantindo que o usuário sempre visualize a interação mais recente.

Por fim, com a integração entre *frontend* e *backend* é feita por meio de uma API REST simples, permite a troca rápida e eficiente de informações. O uso de Axios para comunicação assíncrona possibilita que o *chatbot* responda às mensagens sem interromper a navegação do usuário, mantendo o fluxo natural da conversa. Além disso, a adoção do *Tailwind* CSS simplifica a estilização, promovendo um *layout* responsivo que se adapta bem a diferentes tamanhos de tela, tornando sistema acessível tanto em *Desktops* quanto em *Smartphones* ou *Tablets*.

CONSIDERAÇÕES FINAIS

Este trabalho demonstrou a viabilidade do uso do *framework Flask* no desenvolvimento de *chatbots* para aplicações *web*, unindo simplicidade estrutural e eficiência técnica. A aplicação foi construída com tecnologias acessíveis e consolidadas, como *Python*, HTML, CSS, *JavaScript* e *Tailwind* CSS , mostrando como a integração entre *backend* e *frontend* pode gerar soluções conversacionais funcionais e visualmente eficazes.

O *Flask* provou ser uma base sólida para lidar com rotas, requisições HTTP e integração com modelos de linguagem, possibilitando o desenvolvimento ágil de APIs voltadas à interação com o usuário. A separação entre lógica de servidor e camada de apresentação reforçou a organização do projeto e boas práticas de desenvolvimento web.

No *frontend*, o uso do *Tailwind* CSS facilitou a criação de uma interface limpa, responsiva e centrada no usuário, sem exigir *frameworks JavaScript* complexos. A clareza na comunicação entre as camadas e a simplicidade do código evidenciam que é possível desenvolver chatbots eficientes mesmo em cenários de baixa complexidade técnica.

Conclui-se, portanto, que a integração adotada oferece um caminho viável e replicável para desenvolvedores interessados em construir agentes conversacionais leves, de fácil

manutenção e usabilidade, com potencial tanto para fins didáticos quanto para aplicações práticas.

PALAVRAS-CHAVE

Flask. Chatbots. Python. Web. Aplicações conversacionais.

REFERÊNCIAS

ALMEIDA, Renata. **Frameworks CSS: tendências e aplicações no desenvolvimento web.** São Paulo: Novatec, 2023.

CAMPANHA SETEMBRO VERDE. **Qual a diferença entre Flask e Django?** Setembro Verde, 2024. Disponível em: <https://campanhasetembroverde.com.br/qual-a-diferenca-entre-flask-e-django>. Acesso em: 23 maio 2025.

COSTA, Marcelo. **JavaScript e comunicação assíncrona: APIs REST e AJAX.** Rio de Janeiro: Ciência Moderna, 2021.

CREDITED. **Flask (microframework) – Desenvolvimento web.** Credited, 2024. Disponível em: <https://credited.com.br/glossario/flask-microframework-desenvolvimento-web>. Acesso em: 23 maio 2025.

DOBBALA, Manoj Kumar; LINGOLU, Mani Shankar Srinivas. **Conversational AI and Chatbots: Enhancing User Experience on Websites**. *American Journal of Computer Science and Technology*, v. 7, n. 3, p. 37-47, 2024.

FERREIRA, Carlos. **Desenvolvimento de Chatbots com Python: conceitos e práticas.** São Paulo: Novatec, 2021.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 7. ed. São Paulo: Atlas, 2022.

GRINBERG, Miguel. **Flask Web Development**. 2. ed. Sebastopol: O'Reilly Media, 2018.

JANDERSON, Siqueira. **Construindo Chatbots com IA: como utilizar APIs de Processamento de Linguagem Natural.** Dev.to, 2024. Disponível em: <https://dev.to/jandersonsiqueira/construindo-chatbots-com-ia-como-utilizar-apis-de-processamento-de-linguagem-natural-bfp>. Acesso em: 23 maio 2025.

LEADSTER. **Chatbot: O que é, história e futuro da ferramenta.** 2025. Disponível em: <https://leadster.com.br/chatbot/>. Acesso em: 23 maio 2025.

MENDES, E.; DIAS, M.; PEREIRA, J. C**onversational Agents: Goals, Technologies, Vision and Challenges**. In: Frontiers in Conversational Agent Research, s.l.: s.n., 2022.

MENDONÇA, André. **Inteligência Artificial aplicada a Chatbots.** Rio de Janeiro: Ciência Moderna, 2024.

MONUTTI, Diego. **O que é Flask no Python: Desenvolvimento Web Ágil e Flexível.** Hashtag Treinamentos, 2024. Disponível em: <https://www.hashtagtreinamentos.com/o-que-e-flask-python>. Acesso em: 23 maio 2025.

ORACLE. **O que é um chatbot?.** 2025. Disponível em: <https://www.oracle.com/br/chatbots/what-is-a-chatbot/>. Acesso em: 23 maio 2025.
PEREIRA, Daniela. **Desenvolvimento Front-end: conceitos e práticas.** Curitiba: Appris, 2022.

SANTOS, Maria. **Arquitetura Modular em Desenvolvimento Web.** Curitiba: Appris, 2023.
SILVA, João; OLIVEIRA, Ana. **Processamento de Linguagem Natural: fundamentos e aplicações.** Brasília: Editora UnB, 2022.

SOUZA, Ana; LIMA, Felipe. **Design centrado no usuário para interfaces digitais.** Brasília: Editora UnB, 2023.

STRYKER, Cole; HOLDSWORTH, Jim. **Processamento de Linguagem Natural (PLN).** IBM Think, 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/natural-language-processing>. Acesso em: 23 maio 2025.