Gerador de Conselhos

Autor: Thiago Dias Mazzoni **Data:** 21 de Agosto de 2025

1. Apresentação da Aplicação

A aplicação desenvolvida, intitulada Gerador de Conselhos, é um sistema web simples que demonstra a arquitetura cliente-servidor. O objetivo principal do projeto é ilustrar a comunicação e a interação entre duas linguagens de programação distintas, uma operando no lado do cliente (front-end) e outra no lado do servidor (back-end).

A funcionalidade para o usuário final é direta: uma página web apresenta um botão. Ao clicar neste botão, um conselho ou frase aleatória é solicitado ao servidor e exibido dinamicamente na tela, substituindo a mensagem anterior. Este projeto, embora simples, encapsula os conceitos fundamentais da comunicação web moderna.

2. Arquitetura e Divisão de Linguagens

A aplicação segue o modelo cliente-servidor, onde as responsabilidades são separadas. Cada camada da aplicação foi implementada com uma linguagem de programação específica, adequada às suas funções.

2.1. Front-end (Lado do Cliente)

- **Linguagem Principal: JavaScript**
- **Tecnologias de Apoio:** HTML5 e CSS3

  Responsável para estruturar o conteúdo da página web (HTML), estilizar a apresentação visual (CSS), capturar as interações do usuário, como o clique no botão (JavaScript), iniciar a comunicação com o back-end para solicitar dados (JavaScript), receber os dados do servidor e atualizar a página dinamicamente sem a necessidade de recarregá-la (JavaScript).

2.2. Back-end (Lado do Servidor)

- **Linguagem Principal: Python**
- **Tecnologia de Apoio:** Flask (um micro-framework web)

  Responsável por gerenciar a lógica de negócio da aplicação, manter o "banco de dados" da aplicação (neste caso, uma lista de conselhos em memória), expor um "ponto de acesso"

(endpoint) para que o front-end possa fazer requisições, processar as requisições recebidas, selecionar um conselho aleatório e enviá-lo de volta como resposta.

3. Método de Interface entre as Linguagens

A comunicação entre o front-end (JavaScript) e o back-end (Python) não ocorre diretamente, pois eles operam em ambientes distintos (navegador e servidor). A interface entre eles é realizada através de uma **API (Application Programming Interface)** que opera sobre o protocolo **HTTP**, o protocolo padrão da web.

O método específico utilizado pode ser detalhado nos seguintes passos e conceitos:

1. **API RESTful:** O back-end expõe uma API que segue os princípios do estilo arquitetural REST. Foi criado um **endpoint** específico: a URL http://127.0.0.1:5000/api/conselho. Um endpoint é um endereço específico onde o front-end pode fazer uma requisição para obter um recurso (neste caso, um conselho).
2. **Requisição HTTP com fetch:** Quando o usuário clica no botão, o código JavaScript utiliza a função nativa fetch() para criar e enviar uma **requisição HTTP** do tipo GET para o endpoint definido no back-end. A requisição GET é o método padrão para solicitar dados de um servidor.
3. **Formato de Dados JSON:** Para a troca de informações, foi utilizado o formato **JSON (JavaScript Object Notation)**. O JSON é um padrão de texto leve, legível por humanos e facilmente interpretável por praticamente todas as linguagens de programação, incluindo JavaScript e Python. O servidor Python, ao selecionar um conselho, o "empacota" em um objeto JSON (ex: {“conselho”: “Estude Python todos os dias”}) e nisso este pacote JSON é então enviado como corpo da **resposta HTTP**.
4. **Processamento da Resposta:** O front-end recebe a resposta do servidor. O código JavaScript então decodifica o pacote JSON para extrair o texto do conselho e, por fim, atualiza o conteúdo de um elemento <p> na página HTML, exibindo o resultado para o usuário.
5. **CORS (Cross-Origin Resource Sharing):** Um detalhe técnico crucial para permitir essa comunicação em ambiente de desenvolvimento é o CORS. Por padrão, navegadores bloqueiam requisições de uma origem (ex: https://127.0.0.1:5500 do front-end) para outra (ex: http://127.0.0.1:5000 do back-end). No back-end Python, a biblioteca Flask-CORS foi utilizada para instruir o servidor a aceitar essas requisições, viabilizando a interface entre as duas partes da aplicação.