Laboratório 20: Conectando o Chat (UI)

Objetivo do Laboratório

Habilitar o CORS na nossa API Spring Boot e construir um HTML muito simples com JavaScript para consumir a rota RAG criada no laboratório passado, proporcionando a experiência completa de uso (*Fator Uau!*).

Atividade Prática

1. Liberando o CORS no Java

Abra o projeto do Laboratório 19 na sua IDE. Vá até a classe `ChatController.java` e adicione a anotação `@CrossOrigin` acima da classe para permitir que chamadas do navegador sejam aceitas.

```
import org.springframework.web.bind.annotation.CrossOrigin;
// ... (outros imports)

@RestController
@CrossOrigin(origins = "*") // ATENÇÃO: Libera de qualquer origem (Apenas para fins di
public class ChatController {
    // ... código restante
}
```

Re-execute o projeto Spring Boot.

2. O Código Front-end

Crie uma nova pasta em qualquer lugar do seu computador (fora do projeto Java) e crie um arquivo chamado `index.html`:

```
<!DOCTYPE html>
```

```
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Chat com IA da Empresa</title>
    <style>
        body { font-family: sans-serif; padding: 20px; max-width: 600px; margin: auto
        #chat-box { border: 1px solid #ccc; padding: 10px; min-height: 200px; margin-b
        .user { color: blue; font-weight: bold; }
        .ai { color: green; margin-bottom: 15px; }
    </style>
</head>
<body>

    <h2>Chatbot RAG Corporativo</h2>
    <div id="chat-box"></div>
    <input type="text" id="pergunta" placeholder="Digite sua pergunta..." style="widt
    <button onclick="enviarPergunta()">Enviar</button>

    <script>
        async function enviarPergunta() {
            const input = document.getElementById("pergunta");
            const chatBox = document.getElementById("chat-box");
            const pergunta = input.value;

            if(!pergunta) return;

            // Imprime a pergunta do usuário na tela
            chatBox.innerHTML += `<div class="user">Você: ${pergunta}</div>`;
            chatBox.innerHTML += `<div class="ai" id="loading">Pensando...</div>`;
            input.value = ""; // Limpa a caixa

            try {
                // Requisição FETCH para o nosso Back-end Spring Boot
                const url = `http://localhost:8080/api/rag?msg=${encodeURIComponent(pe
                const resposta = await fetch(url);
                const textoGerado = await resposta.text();

                // Remove o loading e insere a resposta final
                document.getElementById("loading").remove();
                chatBox.innerHTML += `<div class="ai">IA: ${textoGerado}</div>`;
            } catch (erro) {
                document.getElementById("loading").remove();
                chatBox.innerHTML += `<div class="ai" style="color:red;">Erro ao cone
            }
        }
    </script>
```

```
</body>
</html>
```

3. Execução e Entrega

1 Com o servidor Spring rodando, abra o `index.html` diretamente no seu navegador (com um duplo-clique).

2 Faça a pergunta que configuramos no *VectorStore* (ex: sobre o vale-refeição) e aguarde a resposta visual.

3 Comemore a integração *Full-Stack* finalizada!

4 Envie o código modificado do Java (com a anotação do CORS) e o arquivo `index.html` para o GitLab.

Critério de Avaliação

Entrega consolidada do HTML operando o `fetch` e o `ChatController` Java aceitando a requisição HTTP.