### Exercício 9.4 - Tutorial Passo-a-Passo

### (Uso de HTML/CSS/JS para a API de Livros)

### **Parte 1: Configuração do CORS[[1]](#footnote-1) no Backend**

O objetivo desta secção é configurar o Spring Boot para permitir que um ficheiro HTML (frontend) possa fazer requisições à API REST do backend.

Começe por fazer uma cópia da pasta da solução do Exercício 9.3 (API de Livros, que na altura testámos com Postman, por não termos nenhum frontend).

#### 1. Adicionar a Configuração de CORS no Backend

No projeto Spring Boot, crie uma nova classe chamada CorsConfig.java dentro do pacote principal, por exemplo: src/main/java/com/example/livros

No ficheiro de solução do Exercício 9.3, acrescente a classe CorsConfig.java

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

#### 

Insira o seguinte código na classe CorsConfig:

* package com.example.demo;  
    
  import org.springframework.context.annotation.Bean;  
  import org.springframework.context.annotation.Configuration;  
  import org.springframework.web.servlet.config.annotation.CorsRegistry;  
  import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
    
  @Configuration  
  public class CorsConfig {  
    
   @Bean  
   public WebMvcConfigurer corsConfigurer() {  
   return new WebMvcConfigurer() {  
   @Override  
   public void addCorsMappings(CorsRegistry registry) {  
   registry.addMapping("/\*\*") // Permitir todas as rotas da API  
   .allowedOrigins("\*") // Permitir requisições de qualquer origem  
   .allowedMethods("GET", "POST", "DELETE", "PUT", "OPTIONS") // Métodos permitidos  
   .allowedHeaders("\*") // Todos os cabeçalhos permitidos  
   .allowCredentials(false); // Não permite autenticação baseada em cookies  
   }  
   };  
   }  
  }

#### 

#### 2. Inicie (ou Pare e Reinicie) o Backend

**Reinicie a aplicação Spring Boot**

**Depois, certifique-se de que o backend está configurado corretamente.**

O backend deve estar a correr em <http://localhost:8080>.

### **Parte 2: Criar o Frontend em HTML Simples**

Agora vamos criar um ficheiro HTML para testar o backend.

#### 

#### Criar o Ficheiro index.html

Crie um ficheiro chamado index.html.

Insira o seguinte código no ficheiro:

* <!DOCTYPE html>  
  <html lang="en">  
  <head>  
   <meta charset="UTF-8">  
   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
   <title>Gestão de Livros</title>  
   <style>  
   body {  
   font-family: Arial, sans-serif;  
   margin: 20px;  
   text-align: center;  
   }  
   .form {  
   margin-bottom: 20px;  
   }  
   .form input {  
   padding: 10px;  
   margin: 5px;  
   font-size: 16px;  
   }  
   .form button {  
   padding: 10px 15px;  
   font-size: 16px;  
   background-color: #007bff;  
   color: white;  
   border: none;  
   border-radius: 5px;  
   cursor: pointer;  
   }  
   .form button:hover {  
   background-color: #0056b3;  
   }  
   ul {  
   list-style-type: none;  
   padding: 0;  
   }  
   li {  
   margin: 10px 0;  
   font-size: 18px;  
   }  
   .remove-btn {  
   background-color: #dc3545;  
   color: white;  
   border: none;  
   border-radius: 5px;  
   padding: 5px 10px;  
   cursor: pointer;  
   }  
   .remove-btn:hover {  
   background-color: #b02a37;  
   }  
   .error {  
   color: red;  
   font-weight: bold;  
   margin: 10px 0;  
   }  
   </style>  
  </head>  
  <body>  
   <h1>Gestão de Livros</h1>  
   <div class="form">  
   <input type="text" id="titulo" placeholder="Título">  
   <input type="text" id="autor" placeholder="Autor">  
   <button onclick="adicionarLivro()">Adicionar Livro</button>  
   </div>  
   <p id="error" class="error"></p>  
   <ul id="livros-lista"></ul>  
    
   <script>  
   const API\_URL = "http://localhost:8080/api/livros";  
    
   async function listarLivros() {  
   try {  
   const response = await fetch(API\_URL);  
   const livros = await response.json();  
   const lista = document.getElementById("livros-lista");  
   lista.innerHTML = ""; // Limpa a lista antes de atualizar  
   livros.forEach((livro) => {  
   const li = document.createElement("li");  
   li.innerHTML = `  
   <strong>${livro.titulo}</strong> - ${livro.autor}  
   <button class="remove-btn" onclick="removerLivro(${livro.id})">Remover</button>  
   `;  
   lista.appendChild(li);  
   });  
   } catch (error) {  
   mostrarErro("Erro ao buscar livros. Tente novamente.");  
   console.error(error);  
   }  
   }  
    
   async function adicionarLivro() {  
   const titulo = document.getElementById("titulo").value;  
   const autor = document.getElementById("autor").value;  
    
   if (!titulo || !autor) {  
   mostrarErro("Por favor, preencha todos os campos!");  
   return;  
   }  
    
   try {  
   await fetch(API\_URL, {  
   method: "POST",  
   headers: {  
   "Content-Type": "application/json",  
   },  
   body: JSON.stringify({ titulo, autor }),  
   });  
   document.getElementById("titulo").value = "";  
   document.getElementById("autor").value = "";  
   listarLivros(); // Atualiza a lista  
   } catch (error) {  
   mostrarErro("Erro ao adicionar livro. Tente novamente.");  
   console.error(error);  
   }  
   }  
    
   async function removerLivro(id) {  
   try {  
   await fetch(`${API\_URL}/${id}`, { method: "DELETE" });  
   listarLivros(); // Atualiza a lista  
   } catch (error) {  
   mostrarErro("Erro ao remover livro. Tente novamente.");  
   console.error(error);  
   }  
   }  
    
   function mostrarErro(mensagem) {  
   const errorElement = document.getElementById("error");  
   errorElement.textContent = mensagem;  
   setTimeout(() => {  
   errorElement.textContent = "";  
   }, 3000); // Remove o erro após 3 segundos  
   }  
    
   listarLivros();  
   </script>  
  </body>  
  </html>

### **Parte 3: Testar o Projeto**

1. **Abrir o Ficheiro HTML**:
   * Clique duas vezes no ficheiro index.html ou arraste-o para um navegador (e.g., Chrome ou Edge).
2. **Adicionar Livros**:
   * Insira um título e um autor nos campos e clique no botão "Adicionar Livro".
   * O livro deverá aparecer na lista.
3. **Remover Livros**:
   * Clique no botão "Remover" ao lado de um livro.
   * O livro deverá ser removido.
4. **Erros Comuns**:
   * **CORS bloqueado**:
     + Certifique-se de que o backend tem a configuração de CORS corretamente aplicada.
   * **Backend não está a correr**:
     + Inicie o backend com o comando mvn spring-boot:run.

1. O CORS (*Cross-Origin Resource Sharing*) é um mecanismo de segurança dos navegadores que bloqueia requisições feitas de um domínio para outro, protegendo contra acessos não autorizados. Durante o desenvolvimento de aplicações web distribuídas, é necessário configurá-lo no backend (Spring Boot) para permitir que o frontend (ex.: HTML em http://localhost:3000) aceda à API em outro domínio (ex.: http://localhost:8080). Sem esta configuração, o navegador rejeitará as requisições por questões de segurança. [↑](#footnote-ref-1)