## **Melhorias para o Projeto de Inventário**

### **Novas Funcionalidades**

1. **Edição e Remoção de Itens no Inventário**
   * Adicionar a funcionalidade de editar um livro existente.
   * Melhorar a funcionalidade de remoção para exibir confirmações.
2. **Filtros**
   * Permitir buscar livros por título ou referência.
3. **Validações**
   * Garantir que todos os campos obrigatórios estão preenchidos.
   * Impedir valores inválidos para preço e quantidade.
4. **Estilização Avançada**
   * Melhorar a interface com foco na usabilidade e design.

### **Backend Atualizado**

1. **Adicionar Endpoint para Atualizar um Livro**
   * Modificar o controlador LivroController para incluir suporte à edição via PUT.

@PutMapping("/{id}")  
public Livro atualizar(@PathVariable Long id, @RequestBody Livro livro) {  
 livro.setId(id);  
 return livroService.salvarOuAtualizar(livro);  
}

1. **Filtro no Repositório**
   * Adicionar suporte para buscar livros por referência ou título.

@Repository  
public interface LivroRepository extends JpaRepository<Livro, Long> {  
 List<Livro> findByReferenciaContainingIgnoreCaseOrTituloContainingIgnoreCase(String referencia, String titulo);  
}

1. **Atualizar o Serviço para Suporte a Filtros**

public List<Livro> filtrarLivros(String filtro) {  
 return livroRepository.findByReferenciaContainingIgnoreCaseOrTituloContainingIgnoreCase(filtro, filtro);  
}

1. **Atualizar Controlador para Filtragem**
   * Adicionar um endpoint para buscar livros com filtro.

@GetMapping("/buscar")  
public List<Livro> filtrarLivros(@RequestParam String filtro) {  
 return livroService.filtrarLivros(filtro);  
}

### **Frontend Atualizado**

1. **HTML com Funcionalidades de Edição e Filtro**

<!DOCTYPE html>  
<html lang="pt">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Inventário de Livros</title>  
 <style>  
 body {  
 font-family: Arial, sans-serif;  
 margin: 20px;  
 text-align: center;  
 }  
 .form, .filter {  
 margin-bottom: 20px;  
 }  
 .form input, .filter input {  
 padding: 10px;  
 margin: 5px;  
 font-size: 16px;  
 }  
 .form button, .filter button {  
 padding: 10px 15px;  
 font-size: 16px;  
 background-color: #007bff;  
 color: white;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 }  
 .form button:hover, .filter button:hover {  
 background-color: #0056b3;  
 }  
 ul {  
 list-style-type: none;  
 padding: 0;  
 }  
 li {  
 margin: 10px 0;  
 font-size: 18px;  
 }  
 .edit-btn, .remove-btn {  
 background-color: #28a745;  
 color: white;  
 border: none;  
 border-radius: 5px;  
 padding: 5px 10px;  
 cursor: pointer;  
 }  
 .remove-btn {  
 background-color: #dc3545;  
 }  
 .edit-btn:hover {  
 background-color: #218838;  
 }  
 .remove-btn:hover {  
 background-color: #b02a37;  
 }  
 .error {  
 color: red;  
 font-weight: bold;  
 margin: 10px 0;  
 }  
 </style>  
</head>  
<body>  
 <h1>Inventário de Livros</h1>  
 <div class="filter">  
 <input type="text" id="filtro" placeholder="Filtrar por Referência ou Título">  
 <button onclick="filtrarLivros()">Filtrar</button>  
 </div>  
 <div class="form">  
 <input type="text" id="referencia" placeholder="Referência">  
 <input type="text" id="titulo" placeholder="Título">  
 <input type="text" id="descricao" placeholder="Descrição">  
 <input type="number" id="preco" placeholder="Preço">  
 <input type="number" id="quantidade" placeholder="Quantidade">  
 <button id="adicionarBtn" onclick="adicionarLivro()">Adicionar Livro</button>  
 <button id="editarBtn" style="display: none;" onclick="editarLivro()">Guardar Alterações</button>  
 </div>  
 <p id="error" class="error"></p>  
 <ul id="livros-lista"></ul>  
  
 <script>  
 const API\_URL = "http://localhost:8080/api/livros";  
 let livroEditando = null;  
  
 async function listarLivros() {  
 const response = await fetch(API\_URL);  
 const livros = await response.json();  
 renderizarLivros(livros);  
 }  
  
 async function filtrarLivros() {  
 const filtro = document.getElementById("filtro").value;  
 const response = await fetch(`${API\_URL}/buscar?filtro=${filtro}`);  
 const livros = await response.json();  
 renderizarLivros(livros);  
 }  
  
 function renderizarLivros(livros) {  
 const lista = document.getElementById("livros-lista");  
 lista.innerHTML = "";  
 livros.forEach((livro) => {  
 const li = document.createElement("li");  
 li.innerHTML = `  
 <strong>${livro.referencia}</strong>: ${livro.titulo}   
 (${livro.descricao}) - Preço: €${livro.preco.toFixed(2)}   
 | Quantidade: ${livro.quantidade}  
 <button class="edit-btn" onclick="prepararEdicao(${livro.id}, '${livro.referencia}', '${livro.titulo}', '${livro.descricao}', ${livro.preco}, ${livro.quantidade})">Editar</button>  
 <button class="remove-btn" onclick="removerLivro(${livro.id})">Remover</button>  
 `;  
 lista.appendChild(li);  
 });  
 }  
  
 async function adicionarLivro() {  
 const livro = obterDadosFormulario();  
 if (!livro) return;  
 await fetch(API\_URL, {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify(livro),  
 });  
 limparFormulario();  
 listarLivros();  
 }  
  
 async function editarLivro() {  
 const livro = obterDadosFormulario();  
 if (!livro || !livroEditando) return;  
 await fetch(`${API\_URL}/${livroEditando}`, {  
 method: "PUT",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify(livro),  
 });  
 limparFormulario();  
 listarLivros();  
 livroEditando = null;  
 document.getElementById("adicionarBtn").style.display = "inline-block";  
 document.getElementById("editarBtn").style.display = "none";  
 }  
  
 async function removerLivro(id) {  
 await fetch(`${API\_URL}/${id}`, { method: "DELETE" });  
 listarLivros();  
 }  
  
 function prepararEdicao(id, referencia, titulo, descricao, preco, quantidade) {  
 livroEditando = id;  
 document.getElementById("referencia").value = referencia;  
 document.getElementById("titulo").value = titulo;  
 document.getElementById("descricao").value = descricao;  
 document.getElementById("preco").value = preco;  
 document.getElementById("quantidade").value = quantidade;  
 document.getElementById("adicionarBtn").style.display = "none";  
 document.getElementById("editarBtn").style.display = "inline-block";  
 }  
  
 function obterDadosFormulario() {  
 const referencia = document.getElementById("referencia").value;  
 const titulo = document.getElementById("titulo").value;  
 const descricao = document.getElementById("descricao").value;  
 const preco = parseFloat(document.getElementById("preco").value);  
 const quantidade = parseInt(document.getElementById("quantidade").value);  
  
 if (!referencia || !titulo || !descricao || isNaN(preco) || isNaN(quantidade)) {  
 mostrarErro("Todos os campos são obrigatórios!");  
 return null;  
 }  
  
 return { referencia, titulo, descricao, preco, quantidade };  
 }  
  
 function limparFormulario() {  
 document.getElementById("referencia").value = "";  
 document.getElementById("titulo").value = "";  
 document.getElementById("descricao").value = "";  
 document.getElementById("preco").value = "";  
 document.getElementById("quantidade").value = "";  
 }  
  
 function mostrarErro(mensagem) {  
 const errorElement = document.getElementById("error");  
 errorElement.textContent = mensagem;  
 setTimeout(() => (errorElement.textContent = ""), 3000);  
 }  
  
 listarLivros();  
 </script>  
</body>  
</html>

### **Passo-a-Passo para Testar**

1. **Iniciar o backend**:
   * mvn spring-boot:run.
2. **Abrir o index.html**:
   * Preencha o formulário e adicione, edite e filtre livros.
3. **Valide os Campos**:
   * Certifique-se de que todos os campos são obrigatórios e os valores inválidos não são aceitos.