### Passo-a-Passo para Criar a Interface React.js e Consumir a API de Livros

Este guia assume que os formandos não têm React instalado. Seguem os passos desde a instalação até a implementação da interface completa.

### 0. Use o código da Solução do Exercício 9.3 como ponto de partida

Importante: Começe por Criar uma Nova pasta e Colocar lá a Solução do Exercício 9.3 (API de Livros que testámos na altura com ajuda de Postman). No sistema de ficheiros deve ter uma pasta com o código do exercício 9.3.

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

Descrição gerada automaticamente

### 1. Configuração Inicial do Ambiente

1. **Instalar Node.js e npm**:
   * Aceda ao site oficial [Node.js](https://nodejs.org) e descarregue a versão LTS.
   * Após a instalação, verifique as versões:
   * node -v  
     npm -v
2. **Instalar o Create React App**:
   * No terminal, instale o create-react-app globalmente:
   * npm install -g create-react-app

### 2. Criar o Projeto React

1. **Criar o projeto**:
   * No terminal, execute:
   * npx create-react-app react-frontend  
     cd react-frontend
2. **Instalar o Axios**:
   * Axios é uma biblioteca para realizar requisições HTTP. Instale-o com:
   * npm install axios

### 3. Estrutura do Projeto

Após a criação do projeto, organize os ficheiros da seguinte forma:

react-frontend/  
├── public/  
│ └── index.html  
├── src/  
│ ├── components/  
│ │ └── LivroList.js  
│ ├── App.js  
│ ├── index.js  
│ ├── App.css  
│ └── components/LivroList.css  
├── package.json  
└── README.md

### 4. Implementar o Componente LivroList

1. Crie um ficheiro chamado LivroList.js dentro de src/components/.
2. Copie o seguinte código para o ficheiro:

import React, { useState, useEffect } from 'react';  
import axios from 'axios';  
import './LivroList.css';  
  
const LivroList = () => {  
 const [livros, setLivros] = useState([]);  
 const [titulo, setTitulo] = useState('');  
 const [autor, setAutor] = useState('');  
 const [loading, setLoading] = useState(false);  
 const [erro, setErro] = useState('');  
  
 useEffect(() => {  
 fetchLivros();  
 }, []);  
  
 const fetchLivros = async () => {  
 setLoading(true);  
 setErro('');  
 try {  
 const response = await axios.get('http://localhost:8080/api/livros');  
 setLivros(response.data);  
 } catch (error) {  
 setErro('Erro ao buscar livros. Tente novamente mais tarde.');  
 console.error(error);  
 } finally {  
 setLoading(false);  
 }  
 };  
  
 const adicionarLivro = async () => {  
 setErro('');  
 if (!titulo || !autor) {  
 setErro('Por favor, preencha todos os campos!');  
 return;  
 }  
 try {  
 const novoLivro = { titulo, autor };  
 await axios.post('http://localhost:8080/api/livros', novoLivro);  
 fetchLivros();  
 setTitulo('');  
 setAutor('');  
 } catch (error) {  
 setErro('Erro ao adicionar livro. Tente novamente.');  
 console.error(error);  
 }  
 };  
  
 const removerLivro = async (id) => {  
 setErro('');  
 try {  
 await axios.delete(`http://localhost:8080/api/livros/${id}`);  
 fetchLivros();  
 } catch (error) {  
 setErro('Erro ao remover livro. Tente novamente.');  
 console.error(error);  
 }  
 };  
  
 return (  
 <div className="livro-list">  
 <h1>Gestão de Livros</h1>  
 {erro && <p className="erro">{erro}</p>}  
  
 <div className="livro-form">  
 <input  
 type="text"  
 placeholder="Título"  
 value={titulo}  
 onChange={(e) => setTitulo(e.target.value)}  
 />  
 <input  
 type="text"  
 placeholder="Autor"  
 value={autor}  
 onChange={(e) => setAutor(e.target.value)}  
 />  
 <button onClick={adicionarLivro}>Adicionar Livro</button>  
 </div>  
  
 {loading ? (  
 <p>Carregando...</p>  
 ) : (  
 <ul>  
 {livros.map((livro) => (  
 <li key={livro.id}>  
 <strong>{livro.titulo}</strong> - {livro.autor}  
 <button  
 className="remover-btn"  
 onClick={() => removerLivro(livro.id)}  
 >  
 Remover  
 </button>  
 </li>  
 ))}  
 </ul>  
 )}  
 </div>  
 );  
};  
  
export default LivroList;

### 5. Adicionar Estilos ao LivroList

1. Crie o ficheiro LivroList.css dentro de src/components/.
2. Cole o seguinte código:

.livro-list {  
 font-family: Arial, sans-serif;  
 text-align: center;  
 margin: 20px;  
}  
  
.livro-form {  
 margin-bottom: 20px;  
}  
  
.livro-form input {  
 margin: 5px;  
 padding: 10px;  
 font-size: 16px;  
}  
  
.livro-form button {  
 padding: 10px 15px;  
 font-size: 16px;  
 background-color: #007bff;  
 color: white;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
}  
  
.livro-form button:hover {  
 background-color: #0056b3;  
}  
  
ul {  
 list-style-type: none;  
 padding: 0;  
}  
  
li {  
 margin: 10px 0;  
 font-size: 18px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
}  
  
.remover-btn {  
 background-color: #dc3545;  
 color: white;  
 border: none;  
 border-radius: 5px;  
 padding: 5px 10px;  
 cursor: pointer;  
}  
  
.remover-btn:hover {  
 background-color: #b02a37;  
}  
  
.erro {  
 color: red;  
 font-weight: bold;  
 margin: 10px 0;  
}

### 6. Integrar o Componente no App.js

1. Abra o ficheiro App.js e substitua o conteúdo por:

import React from 'react';  
import LivroList from './components/LivroList';  
  
function App() {  
 return (  
 <div className="App">  
 <LivroList />  
 </div>  
 );  
}  
  
export default App;

### 7. Configurar CORS no Backend

Certifique-se de que a classe CorsConfig no Spring Boot está configurada:

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("/\*\*").allowedOrigins("http://localhost:3000");  
 }  
 };  
 }  
}

### 8. Testar o Projeto

1. **Iniciar o Backend**:
   * No diretório do Spring Boot, execute:
   * mvn spring-boot:run
   * Certifique-se de que a API está disponível em http://localhost:8080.
2. **Iniciar o Frontend**:
   * No diretório react-frontend, execute:
   * npm start
   * A interface estará disponível em http://localhost:3000.

### Explicação Pedagógica

1. **React Hooks**:
   * useState: Gerencia o estado do componente (livros, título, autor, erros).
   * useEffect: Carrega a lista de livros ao iniciar o componente.
2. **Axios**:
   * Facilita as requisições HTTP para o backend.
3. **Validação e Mensagens de Erro**:
   * Garante que o utilizador receba feedback ao realizar ações incorretas.

Com este guia, os formandos conseguem implementar uma interface React funcional e conectada ao backend Spring Boot! 😊