### 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

**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, escrevendo na linha de comandos:

* + node -v  
    npm -v

(deve obter números de versões ao escrever esses comandos)

**Instalar o Create React App:**

Faça shift + botão da direita do rato para abrir a linha de comandos no local em que se encontra.

No terminal, instale o create-react-app globalmente, com o comando:

npm install -g create-react-app

**Eventual Erro dado pelo PowerShell (linha de comandos do Windows):**

Se estiver no Windows e a usar o Powershell e der erro, do tipo:

“npm (blá, blá…) scripts is disabled on this system. (blá, blá, …),

corra o Powershell **como administrador** e escreva este comando:

Set-ExecutionPolicy -Scope LocalMachine -ExecutionPolicy RemoteSigned

Depois deste commando, feche o PowerShell, para deixar o modo de administrador, e volte a repetir o passo anterior, por favor (mas só se tiver dado erro, não tente instalar duas vezes).

### 2. Criar o Projeto React

1. **Criar o projeto**:

No terminal, execute:

* + npx create-react-app react-frontend
  + cd react-frontend

Nota: no caso do formador, o processo de instalação assinalou um erro, que foi preciso resolver mais tarde: à frente explica-se como.

**Instalar a biblioteca Axios**:

A Axios é uma biblioteca para realizar requisições HTTP. Instale-o com:

npm install axios

### 3. Estrutura do Projeto

O comando npx “create-react-app” cria logo um projeto pronto a correr (ou deveria: no caso do formador, foi necessário fazer uma pequena instalação extra, que será explicada mais à frente). Apenas terá de se preocupar em criar a pasta “components”, e colocar lá dentro os ficheiros “LivroList.css” (para estilos) e “LivroList.js” (o código React). Depois terá de substituir o código que foi criado por defeito no ficheiro “App.js” pelo pequeno fragmento de código que pode copiar e colar deste manual.

Uma imagem com texto, captura de ecrã, software, número

Descrição gerada automaticamente

### 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

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

### **Configuração do CORS 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. Já tínhamos feito isto para o Exercício 9.4, é só repetir.

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>.

### 8. Testar o Frontend do Projeto

**Iniciar o Frontend**:

No diretório react-frontend, execute:

npm start

O processo leva algum tempo. No caso do formador, o processo deu erro na primeira tentativa. Foi preciso fazer:

npm install web-vitals

(uma das bibliotecas que deram erro na instalação inicial, mesmo tendo feito “--force").

e depois repetir npm start

A interface estará disponível em <http://localhost:3000>.

Depois de alguma espera (se não tiver erros, deve visualizar algo como o que se mostra na imagem seguinte.

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

Descrição gerada automaticamente

### Resumo de Conceitos:

1. **React Hooks**:
   * useState: Gere 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.