

# CRUD Livros da Biblioteca

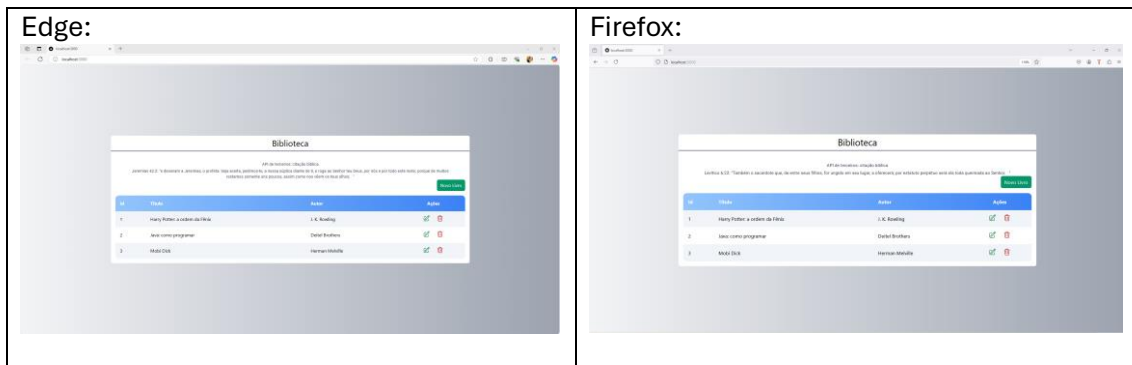
Igor Campos de Borba

## Resumo:

- Introdução: Cadastro (CRUD) de livros de uma biblioteca como prova de conceito.
- Objetivo: Cadastrar, mostrar, atualizar e deletar registros em uma tabela para gerenciamento dos livros.
- Tecnologias: ReactJs, NextJs, Design Materials UI, Tailwind, TypeScript, HTML e CSS.
- Como executar: na pasta do projeto, execute na linha de comando: `npm run dev`
- GitHub: <https://github.com/igorcamosdeborba/library-react>

## 1. Configurar o ambiente para criar e depurar aplicativos React

O aluno testou a aplicação em diferentes navegadores para garantir sua compatibilidade e funcionamento adequado?



## 1. Configurar o ambiente para criar e depurar aplicativos React

O aluno criou a aplicação utilizando o Create React App, conforme as melhores práticas recomendadas?

Software em camadas: domain para representar a entidade central da aplicação; pages para armazenar página; components para separar os componentes e chamadas a API de terceiros. Utilizou-se SOLID: princípio da responsabilidade única na separação dos componentes, princípio da segregação de interfaces ao utilizar-se interfaces como sendo um contrato entre os componentes e uso do TypeScript para tipagem, Princípio aberto para extensão e fechado para modificação ao modularizar o código.

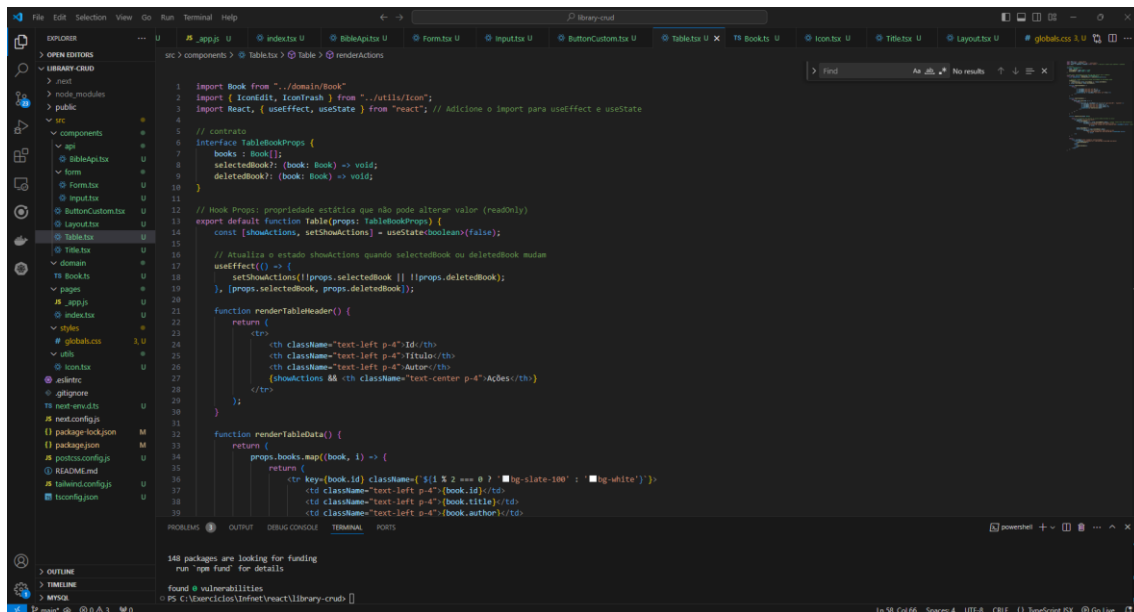
<https://github.com/igorcamosdeborba/library-react/blob/main/src/components/Table.tsx>

## 1. Configurar o ambiente para criar e depurar aplicativos React

O aluno escreveu o código da aplicação de forma que permita uma fácil identificação e correção de bugs, com comentários e uma organização lógica dos componentes?

Há comentários no código sobre o fluxo de processamento dos dados. Comunicação entre componentes por exemplo a passagem do objeto Book entre o componente pai Layout e o filho Table na página index via props. Há comentários sobre o contrato do componente funcional sobre o que ele controla, se é livros e quais as operações como

selectedBook, deletedBook e armazenamento do próprio Book. Uso do Hook useEffect para atualizar o estado dos livros selecionados ou deletados alterando o valor na tabela.



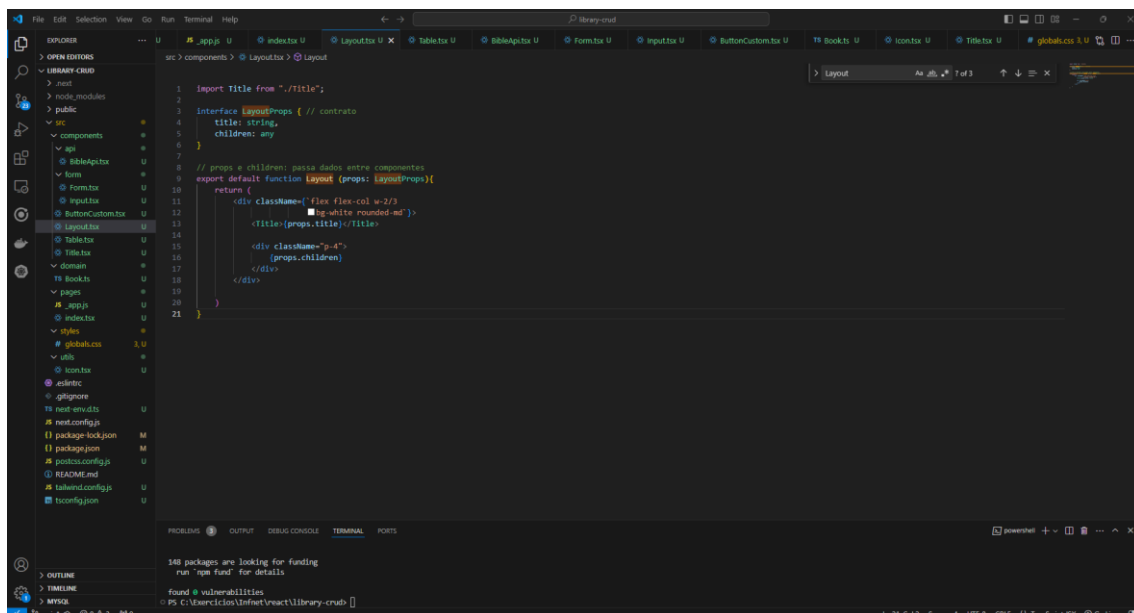
```
1 import Book from "../domain/Book"
2 import { IconEdit, IconTrash } from "../utils/icon"
3 import React, { useEffect, useState } from "react"; // Adicione o import para useEffect e useState
4
5 // contrato
6 interface TableBookProps {
7   books: Book[];
8   selectedBook?: (book: Book) => void;
9   deletedBook?: (book: Book) => void;
10 }
11
12 // Hook Props: propriedade estática que não pode alterar valor (readOnly)
13 export default function Table(props: TableBookProps) {
14   const [showActions, setShowActions] = useState(false);
15
16   // Atualiza o estado showActions quando selectedBook ou deletedBook mudam
17   useEffect(() => {
18     setShowActions(!props.selectedBook || !props.deletedBook);
19   }, [props.selectedBook, props.deletedBook]);
20
21   function renderTableHeader() {
22     return (
23       <tr>
24         <th className="text-left p-4">Id</th>
25         <th className="text-left p-4">Titulo</th>
26         <th className="text-left p-4">Autor</th>
27         <th className="text-left p-4">Ações</th>
28       </tr>
29     );
30   }
31
32   function renderTableData() {
33     return (
34       <tbody>
35         {props.books.map((book, i) => (
36           <tr key={book.id} className={i % 2 === 0 ? "bg-slate-100" : "bg-white"}>
37             <td className="text-left p-4">{book.id}</td>
38             <td className="text-left p-4">{book.title}</td>
39             <td className="text-left p-4">{book.author}</td>
```

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/Table.tsx>

## 1. Configurar o ambiente para criar e depurar aplicativos React

O aluno organizou a estrutura do projeto em módulos e componentes de forma que facilite a manutenção e a compreensão do código?

Há módulos que representam os componentes que constituem as funcionalidades como formulário (Form), tabela, esqueleto do Layout que contém o título e espaço para componentes filhos serem inseridos dinamicamente via props. Eu criei o componente Layout para intercambiar o componente tabela e o formulário no mesmo espaço para evitar a troca de páginas porque são relacionados ao mesmo domínio (book).



```
1 import Title from "../title";
2
3 interface LayoutProps { // contrato
4   title: string;
5   children: any;
6 }
7
8 // props e children: passa dados entre componentes
9 export default function Layout(props: LayoutProps) {
10   return (
11     <div className="flex flex-col w-2/3 bg-white rounded-md">
12       <Title>{props.title}</Title>
13       <div className="p-4">
14         {props.children}
15       </div>
16     </div>
17   );
18 }
```

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/Layout.tsx>

## **2. Explorar React, seus componentes, JSX e recursos do ES6**

O aluno utilizou JSX corretamente para renderizar componentes e elementos na aplicação?

Utilizei TypeScript (TSX) para garantir tipagem e recursos como manipulação de interfaces nos componentes a fim de garantir que o objeto Book esteja consistente e com atributos obrigatórios contidos no fluxo de dados.

## **2. Explorar React, seus componentes, JSX e recursos do ES6**

O aluno criou componentes funcionais e de classe em React, demonstrando compreensão das diferenças entre eles?

No domain usei classe já que representa o escopo central que representa Book. No restante dos componentes utilizei componentes funcionais já que compõem partes que constituem as funcionalidades do CRUD.

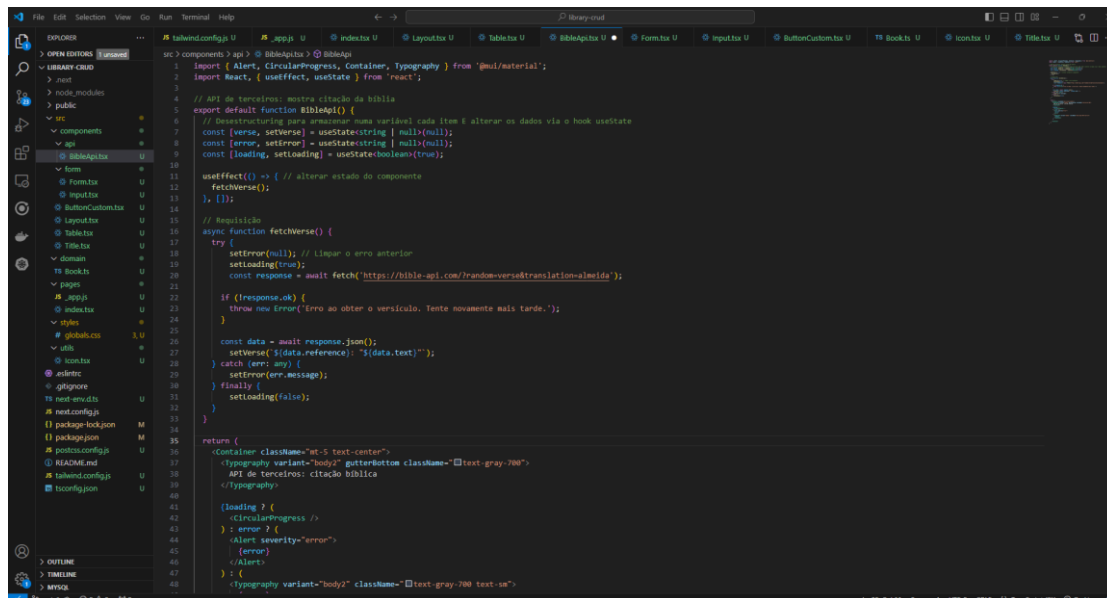
<https://github.com/igorcamosdeborba/library-react/blob/main/src/domain/Book.ts>

<https://github.com/igorcamosdeborba/library-react/blob/main/src/components/form/Form.tsx>

## 2. Explorar React, seus componentes, JSX e recursos do ES6

O aluno aplicou recursos do ES6, como arrow functions e destructuring, de maneira eficaz em seu código React?

Usei destructuring para armazenar numa variável cada item E alterar os dados via o hook useState. Também usei arrow function para alterar o estado do componente via hook useEffect. Na classe no domain usei atributos privados via símbolo #



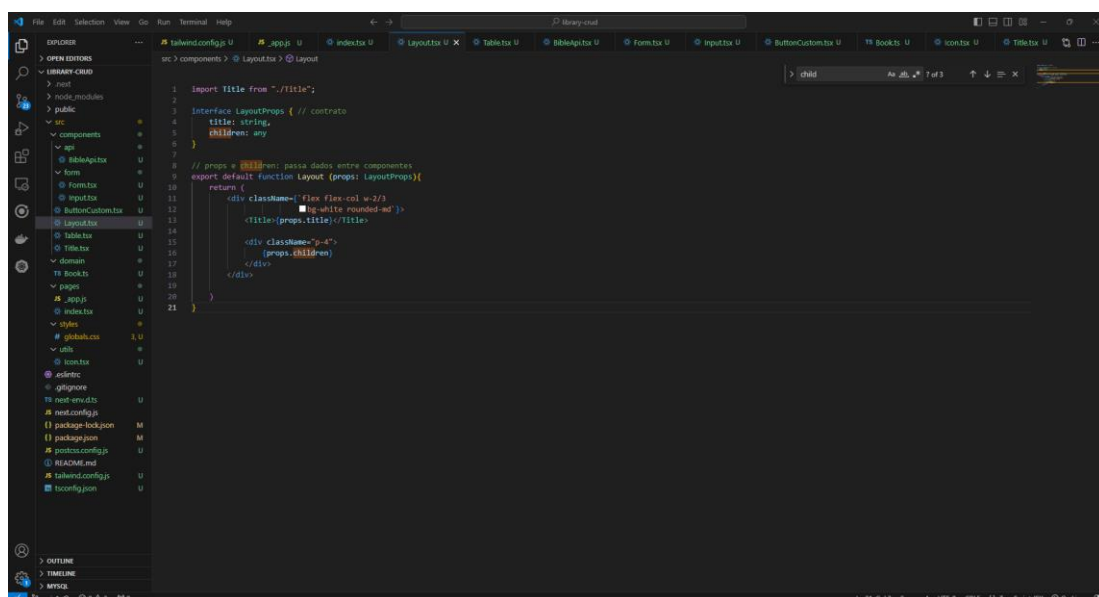
```
1 import { Alert, CircularProgress, Container, Typography } from '@mui/material';
2 import React, { useEffect, useState } from 'react';
3
4 // API de terceiros: mostra citação da biblia
5 export default function BibleApi() {
6   // Destructuring para armazenar numa variável cada item E alterar os dados via o hook useState
7   const [verse, setVerse] = useState<string | null>(null);
8   const [error, setError] = useState<string | null>(null);
9   const [loading, setLoading] = useState<boolean>(true);
10
11   useEffect(() => { // alterar estado do componente
12     fetchVerse();
13   }, []);
14
15   // Requisição
16   async function fetchVerse() {
17     try {
18       setError(null); // Limpar o erro anterior
19       setLoading(true);
20       const response = await fetch('https://bible-api.com/random-verse&translation=almeida');
21       if (!response.ok) {
22         throw new Error('Erro ao obter o versículo. Tente novamente mais tarde.');
```

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/api/BibleApi.tsx>

## 2. Explorar React, seus componentes, JSX e recursos do ES6

O aluno implementou a reutilização de componentes em sua aplicação, utilizando props para passar dados entre eles?

Sim. Criei o componente Layout que transmite via props o Book para componentes filhos como o ButtonCustom que é usado no formulário e na página inicial.



```
1 import Title from './Title';
2
3 interface LayoutProps { // contrato
4   title: string;
5   children: any;
6 }
7
8 // Props e children: para passar dados entre componentes
9 export default function Layout (props: LayoutProps){
10   return (
11     <div className="flex flex-col w-2/3" style={{background: 'white', border: '1px solid #ccc', padding: '10px'}}>
12       <Title>{props.title}</Title>
13
14       <div className="p-4">
15         <div>
16           {props.children}
17         </div>
18       </div>
19     </div>
20   );
21 }
```

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/Layout.tsx>

### 3. Gerenciar estados e componentes com Redux e estilos

O aluno implementou corretamente a estrutura do Redux, incluindo actions, reducers e store, para gerenciar o estado da aplicação?

Não implementado.

### 3. Gerenciar estados e componentes com Redux e estilos

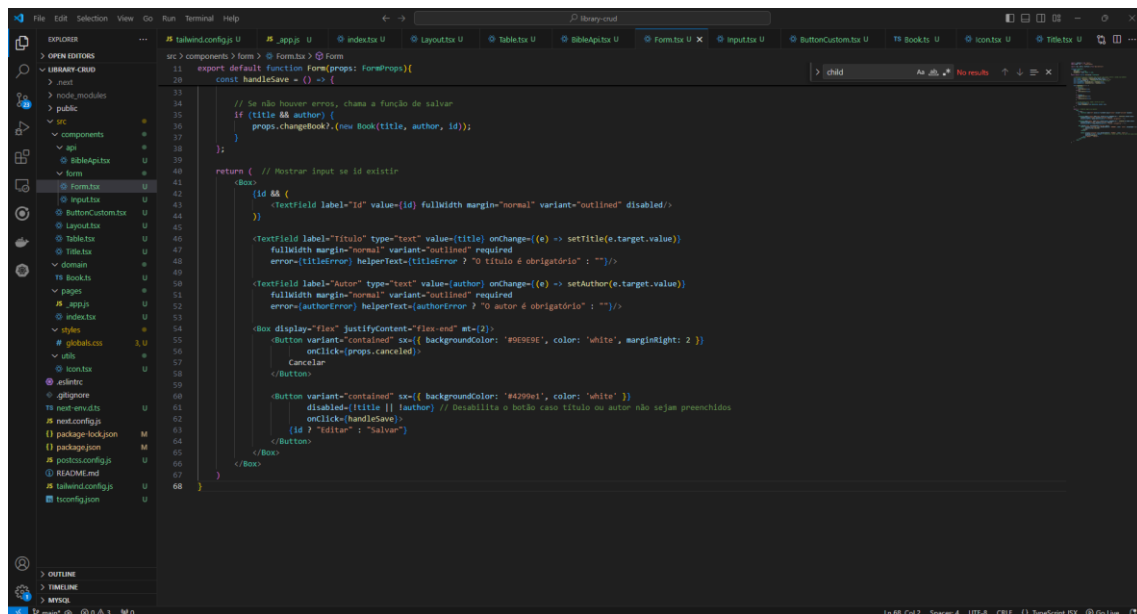
O aluno utilizou o useSelector e useDispatch de forma eficaz para acessar e modificar o estado global da aplicação?

Não implementado.

### 3. Gerenciar estados e componentes com Redux e estilos

O aluno aplicou estilos adequados aos componentes utilizando uma biblioteca de estilos (como Styled Components, Material UI ou CSS Modules) para garantir uma interface visual atraente?

Sim, usei Material UI no formulário. E personalizei o restante da aplicação com Tailwind.



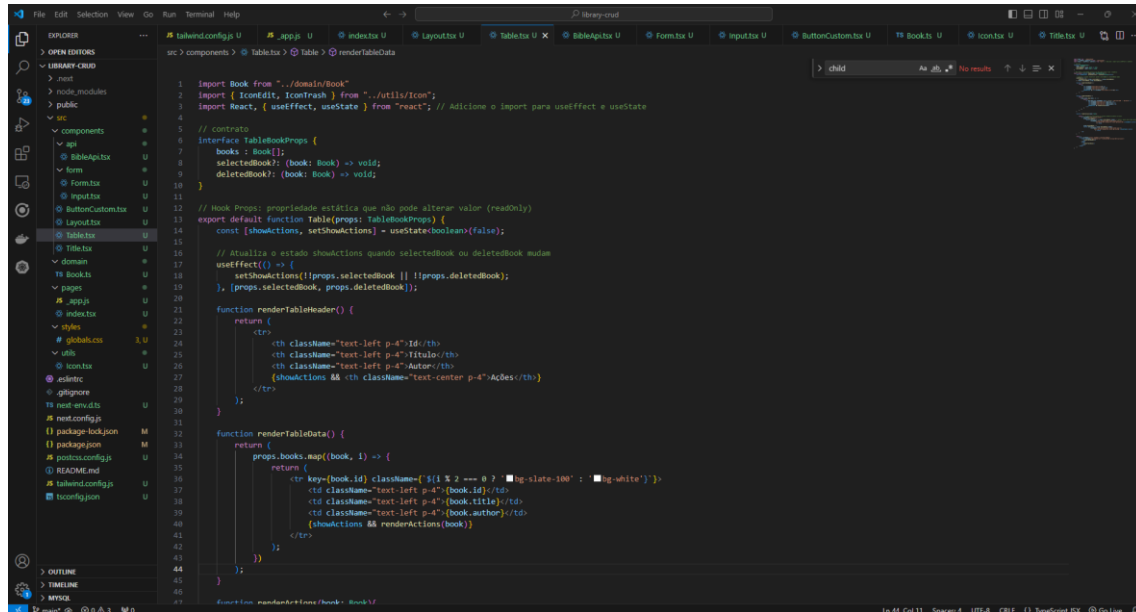
```
11 export default function Form(props: FormProps) {
12   const { handleSave } = () => {
13     // Se não houver erros, chama a função de salvar
14     if (!title || !author) {
15       props.handleChange(new Book(title, author, id));
16     }
17   };
18   return (
19     // Mostrar input se id existir
20     <div>
21       <input
22         type="text"
23         value={id}
24         className="fullWidth margin-normal variant-outlined disabled"
25       />
26       <input
27         type="text"
28         value={title}
29         onChange={e => setTitle(e.target.value)}
30         className="fullWidth margin-normal variant-outlined required"
31         error={titleError}
32         helperText={titleError ? "O título é obrigatório" : ""}
33       />
34       <input
35         type="text"
36         value={author}
37         onChange={e => setAuthor(e.target.value)}
38         className="fullWidth margin-normal variant-outlined required"
39         error={authorError}
40         helperText={authorError ? "O autor é obrigatório" : ""}
41       />
42       <div display="flex" justify-content="flex-end" mt={2}>
43         <button
44           variant="contained"
45           sx={{ backgroundColor: "#9E9E9E", color: "white", marginRight: 2 }}
46           onClick={props.cancel}
47         > Cancelar
48       </button>
49       <button
50         variant="contained"
51         sx={{ backgroundColor: "#42999E", color: "white" }}
52         disabled={!title || !author} // Desabilita o botão caso título ou autor não sejam preenchidos
53         onClick={handleSave}
54       > {id ? "Salvar" : "Salvar"}
55     </div>
56   </div>
57 );
58 }
```

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/form/Form.tsx>

### 3. Gerenciar estados e componentes com Redux e estilos

O aluno demonstrou compreensão do fluxo de dados em uma aplicação React com Redux, garantindo que as atualizações de estado fossem refletidas corretamente na interface?

As operações de CRUD estão funcionando corretamente via manipulação de estados com props, useState e useEffect.



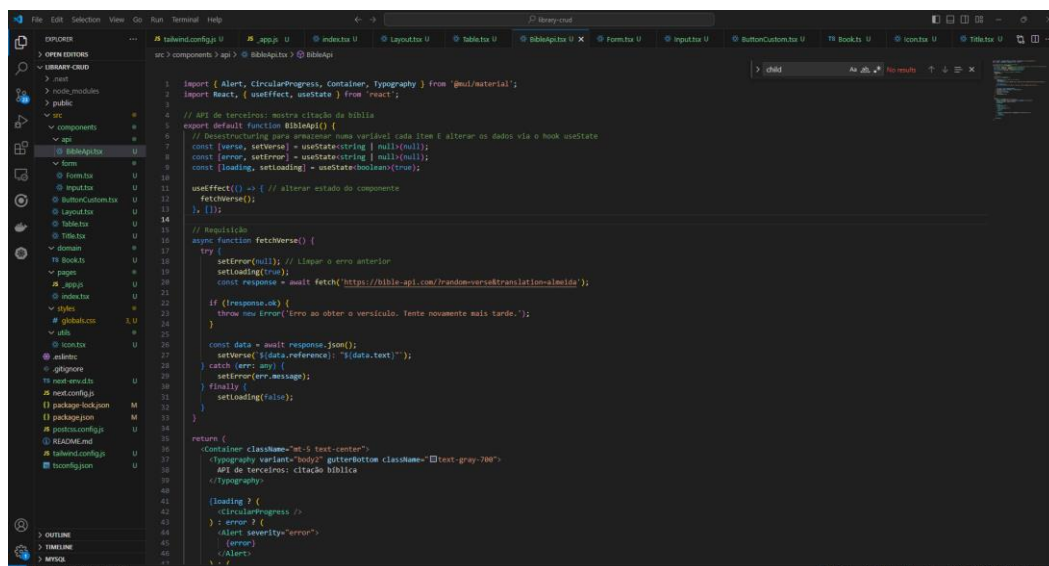
```
1 import Book from "../domain/Book"
2 import { IconEdit, IconTrash } from "../utils/Icons"
3 import React, { useEffect, useState } from "react" // Adicione o import para useEffect e useState
4
5 // contrato
6 interface TableBookProps {
7   books: Book[];
8   selectedBook?: (book: Book) => void;
9   deletedBook?: (book: Book) => void;
10 }
11
12 // Hook Props: propriedade estática que não pode alterar valor (readOnly)
13 export default function Table(props: TableBookProps) {
14   const [showActions, setShowActions] = useState(false);
15
16   // atualiza o estado showActions quando selectedBook ou deletedBook mudam
17   useEffect(() => {
18     setShowActions(!props.selectedBook || !props.deletedBook);
19   }, [props.selectedBook, props.deletedBook]);
20
21   function renderTableHeader() {
22     return (
23       <tr>
24         <th className="text-left p-4">Id</th>
25         <th className="text-left p-4">Titulo</th>
26         <th className="text-left p-4">Autor</th>
27         <th className="text-center p-4">Ações</th>
28       </tr>
29     );
30   }
31
32   function renderTableData() {
33     return (
34       <tbody>
35         {props.books.map((book, i) => {
36           return (
37             <tr key={book.id} className={i % 2 === 0 ? "bg-white" : "bg-gray-100"}>
38               <td className="text-left p-4">{book.id}</td>
39               <td className="text-left p-4">{book.title}</td>
40               <td className="text-left p-4">{book.author}</td>
41               <td>{showActions && renderActions(book)}</td>
42             </tr>
43           );
44         })}
45       </tbody>
46     );
47   }
48 }
```

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/Table.tsx>

### 4. Consumir APIs em aplicações React

O aluno utilizou corretamente a Fetch API ou Axios para realizar requisições GET e POST a uma API externa?

Sim, estou usando Fetch API que é nativa do JavaScript para fazer requisição a API de terceiros no endpoint: 'https://bible-api.com/?random=verse&translation=almeida'. O objetivo é mostrar mensagens bíblicas na tela do usuário.



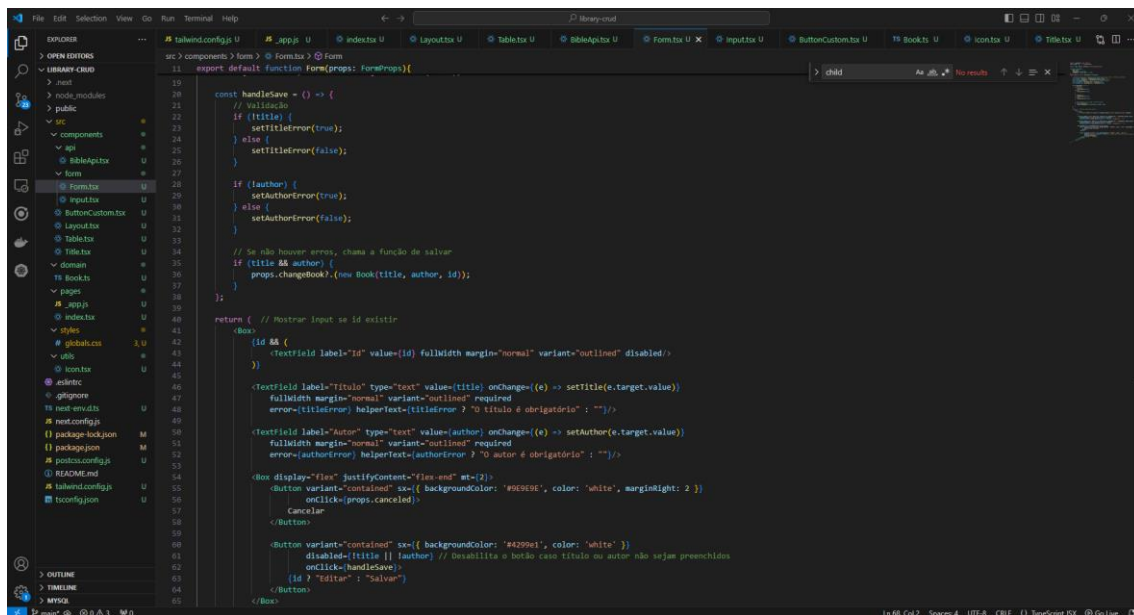
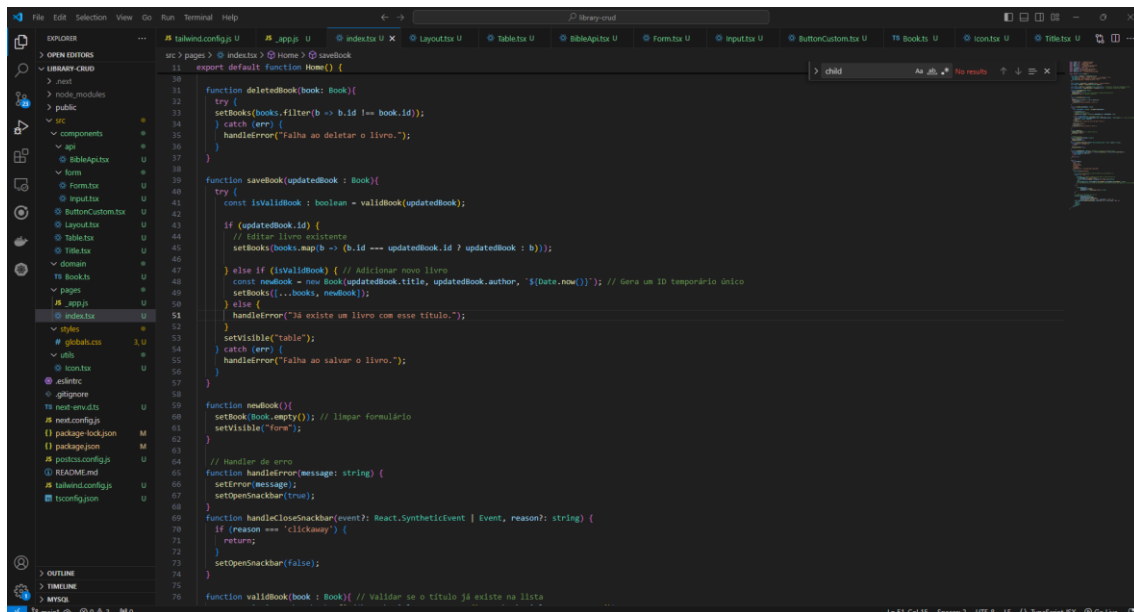
```
1 import { Alert, CircularProgress, Container, Typography } from '@mui/material';
2 import React, { useEffect, useState } from 'react';
3
4 // API de terceiros: mostra citação de bíblia
5 export default function BibleApi() {
6   // Desestruturação para armazenar uma variável cada item e alterar os dados via o hook useState
7   const [verse, setVerse] = useState<string | null>(null);
8   const [error, setError] = useState<string | null>(null);
9   const [loading, setLoading] = useState<boolean>(true);
10
11   // alterar estado do componente
12   useEffect(() => {
13     fetchVerse();
14   }, []);
15
16   // Requisição
17   async function fetchVerse() {
18     try {
19       setError(null); // Limpar o erro anterior
20       setLoading(true);
21       const response = await fetch('https://bible-api.com/random-verse&translation=almeida');
22
23       if (!response.ok) {
24         throw new Error('Erro ao obter o versículo. Tente novamente mais tarde.');
```

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/api/BibleApi.tsx>

## 4. Consumir APIs em aplicações React

O aluno implementou o tratamento de erros ao consumir a API, exibindo mensagens apropriadas em caso de falha nas requisições?

Há tratamentos de erros no CRUD com um handler modularizado e há a exibição de erros no consumo da API de terceiros. Também há validação com mensagens no formulário de modo que só habilite o botão de salvar se o usuário preencheu todos os campos obrigatórios.



<https://github.com/igorcamposdeborba/library-react/blob/main/src/pages/index.tsx>

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/api/BibleApi.tsx>

<https://github.com/igorcamposdeborba/library-react/blob/main/src/components/form/Form.tsx>

#### 4. Consumir APIs em aplicações React

O aluno utilizou React Query ou outra técnica de gerenciamento de estado para facilitar o cache e a atualização automática dos dados consumidos da API?

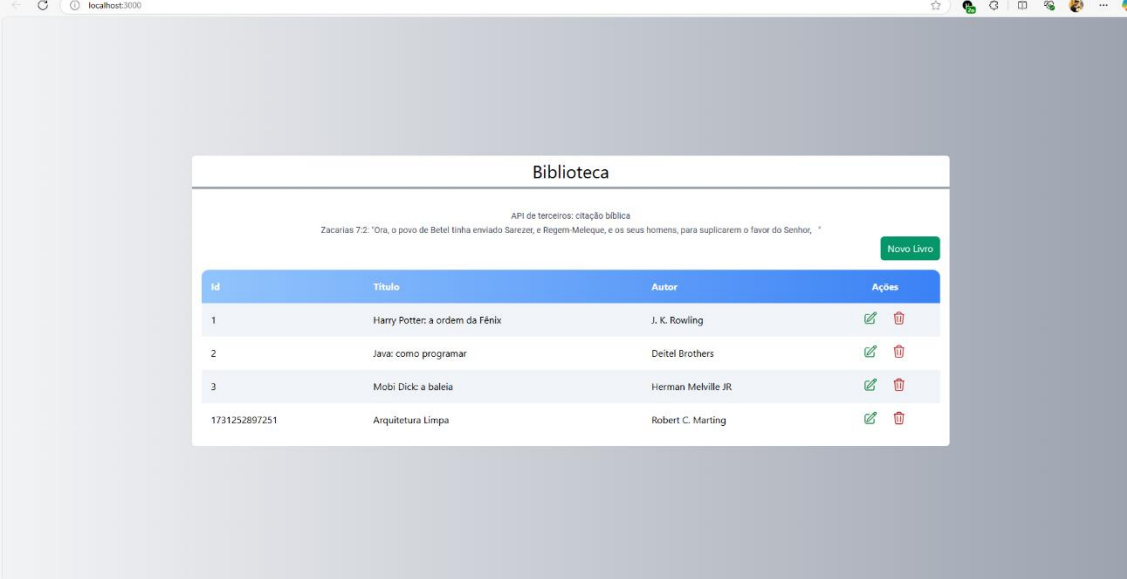
Sim, usei o API Routes do NextJs para gerenciar o estado global e sincronizar com o back-end, que gerencia as páginas ao pré-renderizar o HTML no build do projeto para gerar páginas estáticas. Essas funções permitem que as páginas sejam pré-renderizadas com dados atualizados, facilitando o gerenciamento de estado e evitando a necessidade de uma ferramenta externa como React Query.









#### 4. Consumir APIs em aplicações React

O aluno demonstrou habilidade em renderizar os dados obtidos da API na interface do usuário de forma dinâmica e eficiente?

Sim, as operações de CRUD estão funcionando, principalmente via uso de hooks como useState e useEffect para manipulação do ciclo de vida do componente, comunicação entre componentes via props e vinculação do fluxo de dados via children.

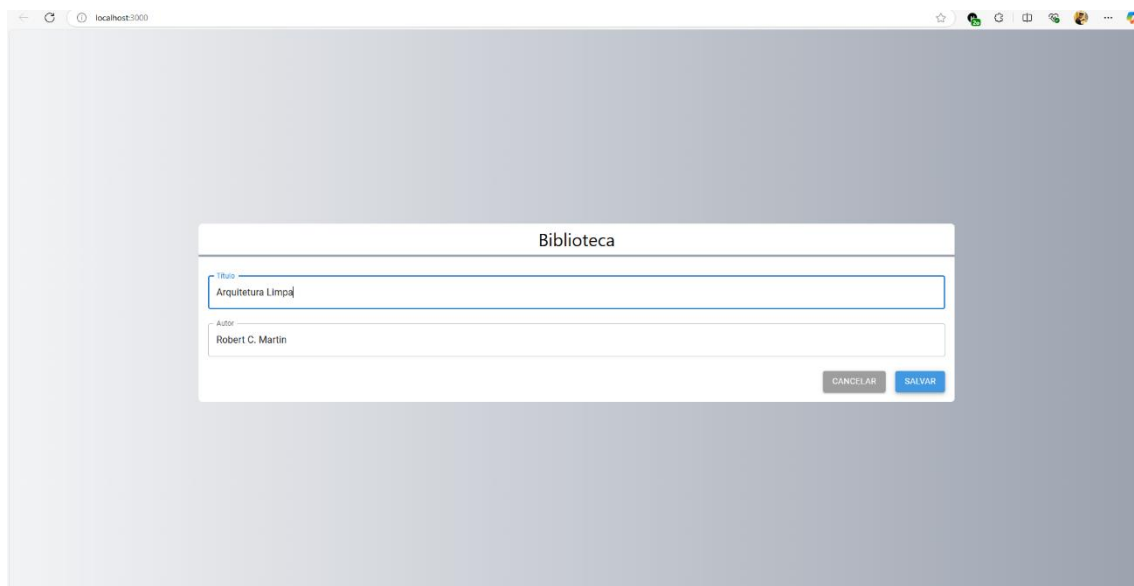
Tabela alterada e adicionado registro (baseado em timestamp):



Biblioteca			
API de terceiros: citação bíblica Zacarias 7:2: "Óra, o povo de Betel tinha enviado Sarezzer, e Riegem-Meleque, e os seus homens, para suplicarem o favor do Senhor, "			
Novo Livro			
Id	Título	Autor	Ações
1	Harry Potter: a ordem da Fênix	J. K. Rowling	 
2	Java: como programar	Deitel Brothers	 
3	Mobi Dick: a baleia	Herman Melville JR	 
1731252897251	Arquitetura Limpa	Robert C. Martin	 

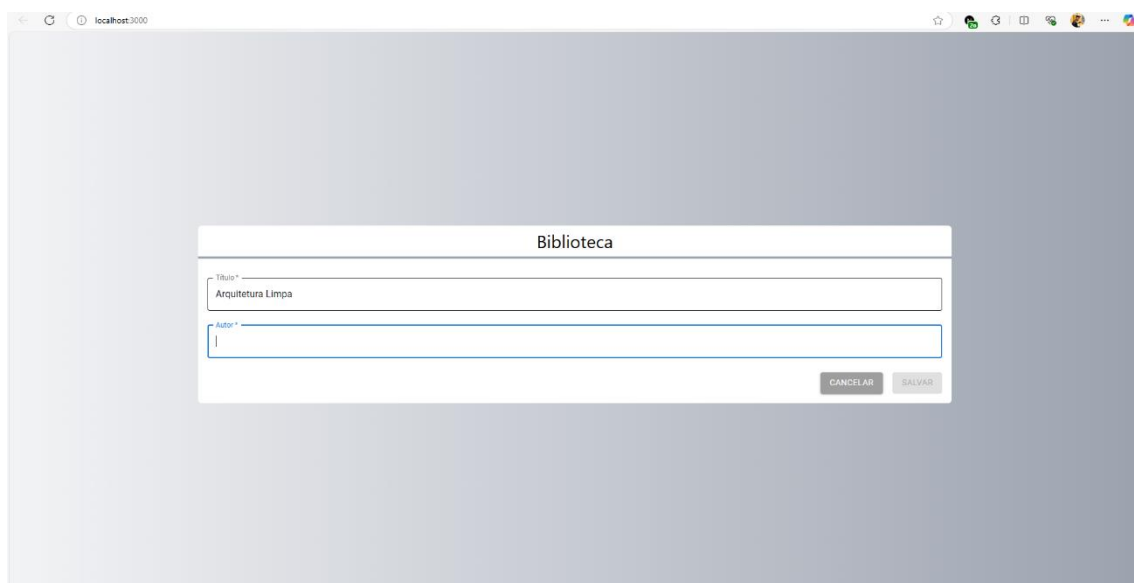


## Formulário com validação botão de salvar ativo



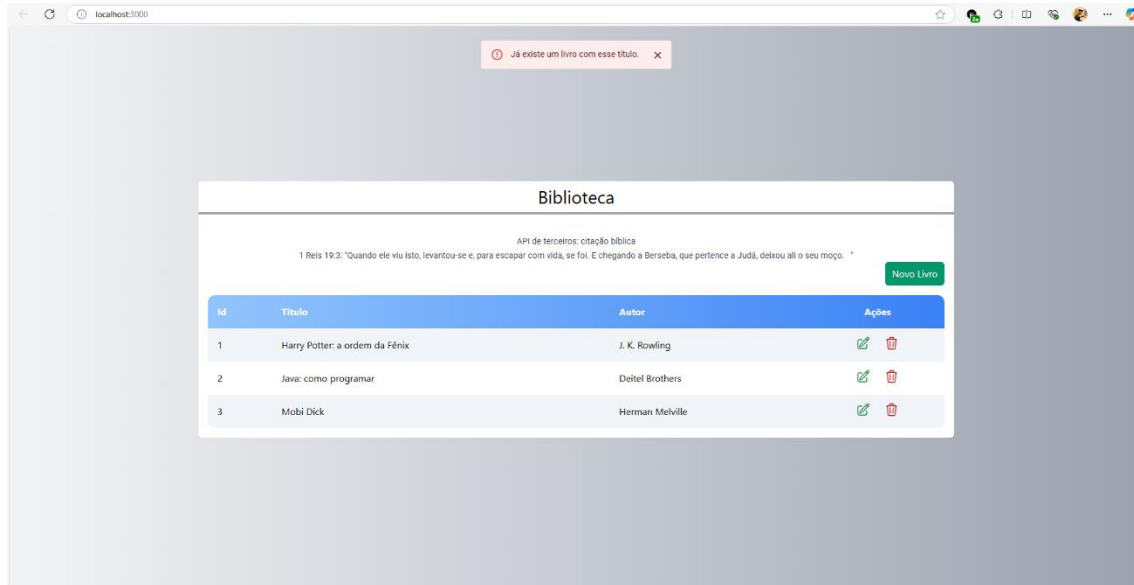
A screenshot of a web browser window displaying a form titled "Biblioteca". The form has two input fields: "Título" (Title) and "Autor" (Author). The "Título" field contains the text "Arquitetura Limpa" and the "Autor" field contains "Robert C. Martin". Below the fields are two buttons: "CANCELAR" (Cancel) and "SALVAR" (Save). The "SALVAR" button is highlighted in blue, indicating it is active. The browser's address bar shows "localhost:3000".

## Formulário com validação botão de salvar desativado quando campos obrigatório não estão preenchidos



A screenshot of a web browser window displaying the same "Biblioteca" form. In this state, the "Título" field is filled with "Arquitetura Limpa", but the "Autor" field is empty. The "SALVAR" button is now greyed out, indicating it is disabled due to the missing required information. The "CANCELAR" button remains active. The browser's address bar shows "localhost:3000".

## Validação de exception:



## Considerações finais

O trabalho foi um desafio para a manipulação do estado dos componentes e fluxo de dados. Contudo, o desafio agregou conhecimento ao usar com mais profundidade os recursos disponíveis.

## Créditos

Validação de formulários pelo canal do YouTube do Lucas Souza Dev:

<https://youtu.be/J9uZK-NyiUo?si=llMlx5pUg0uOQAhp>

Ciclo de vida dos Hooks e uso de endpoints pelo YouTube de Luiz Fernando Nunes:

<https://www.youtube.com/watch?v=yzz78l4ydQk&list=PLvGbhQluFefYF9WxDLnv3cxeGqnkE1WXs>