React

Site: Geração Tech
Curso: Formação em Desenvolvedor Web - Online
Livro: React

Impresso por: JOÃO VITOR DE MELO FREITAS
Data: quinta-feira, 18 jul. 2024, 22:44

Índice

1. Consumindo API e Exibindo Dados com PrimeReact

Passo 1: Estrutura Inicial

Vamos começar criando a estrutura básica do nosso projeto.

Estrutura HTML

Vamos criar uma seção (`section`) com um título (`h1`) e uma lista (`ul`) para exibir nossos produtos.

App.jsx

jsx

Copiar código

```
import React from 'react'; import 'primeflex/primeflex.css'; const App = () => { return ( <section className="p-p-4"> <h1>Lista de Produtos</h1> <ul className="p-grid p-nogutter"> <li className="p-col-12 p-md-3 p-mb-2"> <div className="p-card"> Produto 1 </div> </li> <li className="p-col-12 p-md-3 p-mb-2"> <div className="p-card"> Produto 2 </div> </li> <li className="p-col-12 p-md-3 p-mb-2"> <div className="p-card"> Produto 3 </div> </li> <li className="p-col-12 p-md-3 p-mb-2"> <div className="p-card"> Produto 4 </div> </li> </ul> </section> ); }; export default App;
```

Passo 2: Estilização com PrimeFlex

Vamos utilizar classes do PrimeFlex para estilizar nossa aplicação.

Classes Utilizadas

- **p-p-4**: Aplica padding ao redor do elemento.
- **p-grid**: Define um container de grid.
- **p-nogutter**: Remove espaços entre colunas.
- **p-col-12**: Define a largura como 100% em telas pequenas.
- **p-md-3**: Define a largura como 3 colunas (25%) em telas médias.
- **p-mb-2**: Aplica margem inferior ao elemento.

Passo 3: Consumir API com Fetch

Vamos agora consumir a API `Fake Store` e exibir os produtos dinamicamente.

Atualização do App.jsx

jsx

Copiar código

```
import React, { useEffect, useState } from 'react'; import 'primeflex/primeflex.css'; const App = () => { const [produtos, setProdutos] = useState([]); useEffect(() => { fetch('https://fakestoreapi.com/products') .then(response => response.json()) .then(data => setProdutos(data)) .catch(error => console.error('Erro ao buscar produtos:', error)); }, []); return ( <section className="p-p-4"> <h1>Lista de Produtos</h1> <ul className="p-grid p-nogutter"> {produtos.map(produto => ( <li key={produto.id} className="p-col-12 p-md-3 p-mb-2"> <div className="p-card"> <img src={produto.image} alt={produto.title} className="p-card-image" style={{ width: '100%' }} /> <div className="p-card-content"> <h2 className="p-card-title"> {produto.title}</h2> <p className="p-card-text">{produto.description}</p> <p className="p-card-price">${produto.price}</p> </div> </div> </li> ))} </ul> </section> ); }; export default App;
```

Explicação do Código

- **useState**: Utilizamos o `useState` para gerenciar o estado dos produtos.
- **useEffect**: Utilizamos o `useEffect` para fazer a requisição à API `Fake Store` quando o componente é montado.
- **fetch**: Fazemos uma requisição `fetch` para a API `Fake Store` e armazenamos os dados dos produtos no estado `produtos`.
- **map**: Utilizamos o método `map` para iterar sobre os produtos e renderizá-los dinamicamente na lista (`ul`).

Passo 4: Estilização Adicional

Vamos adicionar algumas classes de estilização aos nossos cards para melhorar a aparência.

Atualização do App.jsx

jsx

Copiar código

```
import React, { useEffect, useState } from 'react'; import 'primeflex/primeflex.css'; import 'primereact/resources/primereact.min.css'; import 'primereact/resources/themes/saga-blue/theme.css'; const App = () => { const [produtos, setProdutos] = useState([]); useEffect(() => { fetch('https://fakestoreapi.com/products') .then(response => response.json()) .then(data => setProdutos(data)) .catch(error => console.error('Erro ao buscar produtos:', error)); }, []); return ( <section className="p-p-4"> <h1>Lista de Produtos</h1> <ul className="p-grid p-nogutter"> {produtos.map(produto => ( <li key={produto.id} className="p-col-12 p-md-3 p-mb-2"> <div className="p-card p-shadow-5 p-p-3"> <img src={produto.image} alt={produto.title} className="p-card-image" style={{ width: '100%' }} /> <div className="p-card-content"> <h2 className="p-card-title p-text-truncate">{produto.title}</h2> <p className="p-card-text p-text-truncate" style={{ maxHeight: '3rem' }}>{produto.description}</p> <p className="p-card-price">${produto.price}</p> </div> </div> </li> ))} </ul> </section> ); }; export default App;
```

Classes Adicionadas

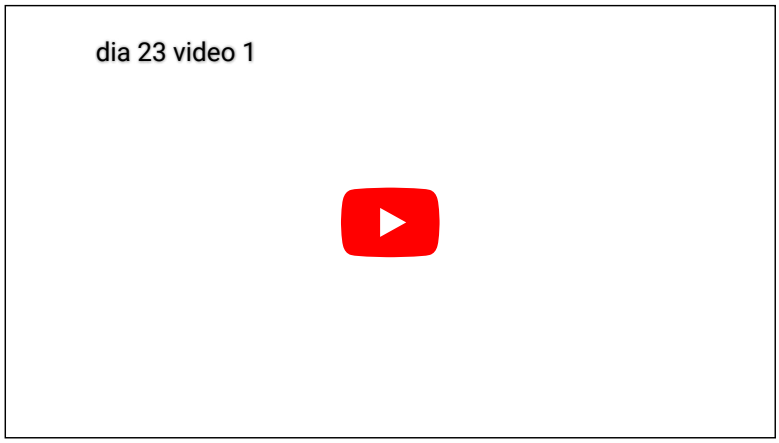
- **p-shadow-5**: Aplica sombra ao card.
- **p-p-3**: Aplica padding ao redor do card.
- **p-text-truncate**: Trunca o texto se for muito longo.
- **p-card-content**: Define o conteúdo do card.
- **p-card-title**: Define o título do card.
- **p-card-text**: Define o texto do card.
- **p-card-price**: Define o preço do card.

Conclusão

Nesta aula, aprendemos a consumir uma API e exibir os dados utilizando React e PrimeReact. Fizemos uma listagem de produtos e estilizamos os cards para melhorar a aparência da aplicação.

Até a próxima aula!

1.1. Video Aula



2. Estrutura do Card de Produto com PrimeReact

Introdução

Olá, pessoal! Tudo bem com vocês? Hoje vamos continuar construindo nossa lista de produtos, focando na criação e estilização do card de produto. Vamos estruturar o card, definir como ele será exibido e estilizar utilizando PrimeFlex.

Passo 1: Estrutura do Card de Produto

Vamos criar a estrutura básica do card de produto com as seguintes informações:

- Imagem do produto
- Nome do produto
- Categoria
- Preço
- Avaliação

App.jsx

jsx

Copiar código

```
import React, { useEffect, useState } from 'react'; import 'primeflex/primeflex.css'; import 'primereact/resources/primereact.min.css'; import 'primereact/resources/themes/saga-blue/theme.css'; const App = () => { const [produtos, setProdutos] = useState([]); useEffect(() => { fetch('https://fakestoreapi.com/products') .then(response => response.json()) .then(data => setProdutos(data)) .catch(error => console.error('Erro ao buscar produtos:', error)); }, []); return ( <section className="p-p-4"> <h1>Lista de Produtos</h1> <ul className="p-grid p-nogutter"> {produtos.map(produto => ( <li key={produto.id} className="p-col-12 p-md-3 p-mb-2"> <div className="p-card p-shadow-4 p-p-3"> <img src={produto.image} alt={produto.title} className="p-card-image" style={{ width: '100%', height: '200px', objectFit: 'contain' }} /> <div className="p-card-content"> <h3 className="p-card-title p-mb-2">{produto.title}</h3> <h6 className="p-card-category p-text-uppercase p-mb-2">{produto.category}</h6> <h2 className="p-card-price">${produto.price}</h2> <div className="p-card-rating p-bg-primary p-py-1 p-px-2 p-border-round-md p-text-bold"> {produto.rating.rate} </div> </div> </div> </li> ))} </ul> </section> ); }; export default App;
```

Passo 2: Estilização dos Cards

Vamos adicionar classes de estilização aos nossos cards para melhorar a aparência da aplicação.

Classes Utilizadas

- **p-shadow-4**: Aplica uma sombra ao card.
- **p-p-3**: Aplica padding ao redor do card.
- **p-card-image**: Define o estilo da imagem do card.
- **p-card-content**: Define o conteúdo do card.
- **p-card-title**: Define o título do card.
- **p-card-category**: Define a categoria do card.
- **p-card-price**: Define o preço do card.
- **p-card-rating**: Define a avaliação do card.

Explicação do Código

- **useState**: Utilizamos o `useState` para gerenciar o estado dos produtos.
- **useEffect**: Utilizamos o `useEffect` para fazer a requisição à API `Fake Store` quando o componente é montado.
- **fetch**: Fazemos uma requisição `fetch` para a API `Fake Store` e armazenamos os dados dos produtos no estado `produtos`.
- **map**: Utilizamos o método `map` para iterar sobre os produtos e renderizá-los dinamicamente na lista (`ul`).

Passo 3: Replicar Estrutura para Todos os Produtos

Agora que temos a estrutura do card pronta, vamos popular a lista com todos os produtos recebidos da API.

Atualização do App.jsx

jsx

Copiar código

```
import React, { useEffect, useState } from 'react'; import 'primeflex/primeflex.css'; import
'primereact/resources/primereact.min.css'; import 'primereact/resources/themes/saga-blue/theme.css'; const App = () => {
const [produtos, setProdutos] = useState([]); useEffect(() => { fetch('https://fakestoreapi.com/products') .then(response =>
response.json()) .then(data => setProdutos(data)) .catch(error => console.error('Erro ao buscar produtos:', error)); }, []);
return ( <section className="p-p-4"> <h1>Lista de Produtos</h1> <ul className="p-grid p-nogutter"> {produtos.map(produto =>
( <li key={produto.id} className="p-col-12 p-md-3 p-mb-2"> <div className="p-card p-shadow-4 p-p-3"> <img src=
{produto.image} alt={produto.title} className="p-card-image" style={{ width: '100%', height: '200px', objectFit: 'contain'
}} /> <div className="p-card-content"> <h3 className="p-card-title p-mb-2">{produto.title}</h3> <h6 className="p-card-
category p-text-uppercase p-mb-2">{produto.category}</h6> <h2 className="p-card-price">${produto.price}</h2> <div
className="p-card-rating p-bg-primary p-py-1 p-px-2 p-border-round-md p-text-bold"> {produto.rating.rate} </div> </div>
</div> </li> ))} </ul> </section> ); }; export default App;
```

Conclusão

Nesta aula, aprendemos a estruturar e estilizar um card de produto utilizando React e PrimeReact. Criamos a estrutura básica do card, adicionamos estilização e populamos a lista de produtos com dados recebidos de uma API.

Até a próxima aula!

2.1. Vídeo Aula

dia 23 video 2

3. Consumindo API Fake Store com Axios

Introdução

Olá, pessoal! Tudo bem com vocês? Hoje vamos continuar nossa aplicação React, desta vez integrando uma API real para popular nossa lista de produtos. Utilizaremos a biblioteca Axios para facilitar nossas requisições HTTP. Vamos começar!

Passo 1: Instalando Axios

Primeiro, vamos instalar a biblioteca Axios. Abra o terminal e execute o seguinte comando:

bash
Copiar código

```
npm install axios
```

Passo 2: Configurando a API com Axios

Vamos criar um serviço para centralizar a configuração da base URL da nossa API.

Estrutura de Arquivos

Crie uma pasta `services` e um arquivo `index.js` dentro dela.

services/index.js

javascript
Copiar código

```
import axios from 'axios'; export const api = axios.create({ baseURL: 'https://fakestoreapi.com', });
```

Passo 3: Fazendo a Requisição para a API

Agora, vamos fazer a requisição para a API Fake Store utilizando o Axios dentro do nosso componente React.

App.jsx

jsx
Copiar código

```
import React, { useEffect, useState } from 'react'; import 'primeflex/primeflex.css'; import
'primereact/resources/primereact.min.css'; import 'primereact/resources/themes/saga-blue/theme.css'; import { api } from
'./services'; const App = () => { const [produtos, setProdutos] = useState([]); useEffect(() => { const buscarProdutos =
async () => { try { const resposta = await api.get('/products'); setProdutos(resposta.data); } catch (erro) {
console.error('Erro ao buscar produtos:', erro); } }; buscarProdutos(); }, []); return ( <section className="p-p-4">
<h1>Lista de Produtos</h1> <ul className="p-grid p-nogutter"> {produtos.map(produto => ( <li key={produto.id} className="p-
col-12 p-md-3 p-mb-2"> <div className="p-card p-shadow-4 p-p-3"> <img src={produto.image} alt={produto.title} className="p-
card-image" style={{ width: '100%', height: '200px', objectFit: 'contain' }} /> <div className="p-card-content"> <h3
className="p-card-title p-mb-2">{produto.title}</h3> <h6 className="p-card-category p-text-uppercase p-mb-2">
{produto.category}</h6> <h2 className="p-card-price">${produto.price}</h2> <div className="p-card-rating p-bg-primary p-py-1
p-px-2 p-border-round-md p-text-bold"> {produto.rating.rate} </div> </div> </div> </li> ))} </ul> </section> ); }; export
default App;
```

Explicação do Código

1. **Configuração do Axios:** No arquivo `services/index.js`, configuramos o Axios com a base URL da API Fake Store.
2. **useEffect:** Utilizamos o `useEffect` para fazer a requisição à API quando o componente é montado.
3. **Função assíncrona:** Criamos a função `buscarProdutos` que faz a requisição à API usando `api.get('/products')`. Se a requisição for bem-sucedida, armazenamos os dados no estado `produtos` usando `setProdutos(resposta.data)`.
4. **Renderização dos Produtos:** Iteramos sobre o estado `produtos` usando o método `map` e renderizamos cada produto em um card.

Passo 4: Estilização do Card de Produto

Já temos a estrutura básica do card. Vamos revisar a estilização para garantir que tudo está correto.

Estilização do Card

jsx

Copiar código

```
<li key={produto.id} className="p-col-12 p-md-3 p-mb-2"> <div className="p-card p-shadow-4 p-p-3"> <img src={produto.image} alt={produto.title} className="p-card-image" style={{ width: '100%', height: '200px', objectFit: 'contain' }} /> <div className="p-card-content"> <h3 className="p-card-title p-mb-2">{produto.title}</h3> <h6 className="p-card-category p-text-uppercase p-mb-2">{produto.category}</h6> <h2 className="p-card-price">${produto.price}</h2> <div className="p-card-rating p-bg-primary p-py-1 p-px-2 p-border-round-md p-text-bold"> {produto.rating.rate} </div> </div> </div> </li>
```

Conclusão

Nesta aula, aprendemos a integrar uma API real usando Axios, configurar um serviço de API e popular nossa lista de produtos no React. Agora, nossa aplicação está pronta para exibir produtos reais.

Espero que vocês tenham gostado e vejo vocês na próxima aula! Até lá!

3.1. Vídeo Aulas

dia 23 video 3

4. Integração com API

Resumo do Progresso

Até agora, criamos a estrutura da nossa aplicação React, fizemos a chamada para a API Fake Store usando Axios, e renderizamos uma lista de produtos na nossa interface. Agora, vamos continuar ajustando a renderização dos produtos e refinando a nossa interface.

Melhorando a Renderização dos Produtos

Vamos melhorar a exibição dos produtos, garantindo que o layout fique mais organizado e as informações sejam exibidas corretamente.

Estrutura de Arquivos

- **App.jsx**
- **services/index.js**

Ajustando o Layout

Vamos garantir que cada produto seja exibido corretamente no layout.

App.jsx

jsx

Copiar código

```
import React, { useEffect, useState } from 'react'; import 'primeflex/primeflex.css'; import 'primereact/resources/primereact.min.css'; import 'primereact/resources/themes/saga-blue/theme.css'; import { api } from './services'; const App = () => { const [produtos, setProdutos] = useState([]); useEffect(() => { const buscarProdutos = async () => { try { const resposta = await api.get('/products'); setProdutos(resposta.data); } catch (erro) { console.error('Erro ao buscar produtos:', erro); } }; buscarProdutos(); }, []); return ( <section className="p-p-4"> <h1>Lista de Produtos</h1> <ul className="p-grid p-nogutter"> {produtos.map(produto => ( <li key={produto.id} className="p-col-12 p-md-3 p-mb-2"> <div className="p-card p-shadow-4 p-p-3"> <img src={produto.image} alt={produto.title} className="p-card-image" style={{ width: '100%', height: '200px', objectFit: 'contain' }} /> <div className="p-card-content"> <h3 className="p-card-title p-mb-2 p-text-nowrap p-text-overflow-ellipsis">{produto.title}</h3> <h6 className="p-card-category p-text-uppercase p-mb-2">{produto.category}</h6> <h2 className="p-card-price">${produto.price}</h2> <div className="p-card-rating p-bg-primary p-py-1 p-px-2 p-border-round-md p-text-bold"> {produto.rating.rate} </div> </div> </div> </li> ))} </ul> </section> ); }; export default App;
```

Explicação do Código

1. **Produto Map:** Utilizamos o método `map` para iterar sobre a lista de produtos e renderizar cada um deles dentro de um `li`.
2. **Imagem do Produto:** Ajustamos a imagem do produto usando `style={{ width: '100%', height: '200px', objectFit: 'contain' }}` para garantir que a imagem mantenha suas proporções.
3. **Título do Produto:** Utilizamos classes como `p-text-nowrap` e `p-text-overflow-ellipsis` para garantir que o texto do título não ultrapasse os limites do card e adicione reticências quando necessário.

Melhorias no Estilo

Adicionamos algumas classes de estilo adicionais para garantir que o layout dos produtos fique mais organizado e legível.

Adicionando Classes

jsx

Copiar código

```
<h3 className="p-card-title p-mb-2 p-text-nowrap p-text-overflow-ellipsis">{produto.title}</h3> <h6 className="p-card-category p-text-uppercase p-mb-2">{produto.category}</h6> <h2 className="p-card-price">${produto.price}</h2> <div className="p-card-rating p-bg-primary p-py-1 p-px-2 p-border-round-md p-text-bold"> {produto.rating.rate} </div>
```

Conclusão

Com essas melhorias, nossa aplicação está agora exibindo produtos de maneira mais organizada e visualmente agradável. Aprendemos a consumir uma API externa com Axios, manipular o estado com hooks do React, e aplicar estilos utilizando classes do PrimeFlex.

Espero que tenham gostado desta aula. Continuaremos aprimorando nossa aplicação nas próximas aulas. Até lá!

4.1. Vídeo Aula

dia 23 video 4