

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, 22:43

Índice

1. React-hook-form

1.1. Vídeo da aula

2. PrimeReact e React Context


2.1. Vídeo da aula

1. React-hook-form

Então, o que a gente vai fazer aqui? A ideia que vem na mente é: "Poxa, eu acho que vou utilizar a sugestão que o Gleidson deu lá no começo e usar um `useRef` para o e-mail e um `useRef` para a senha, porque aí não vou ficar renderizando muitas vezes à medida que o usuário for digitando." Sucesso! Ótimo que você pensou nisso, mas não é isso que eu vou fazer. É uma ótima ideia, mas eu quero apresentar para vocês uma ferramenta que eu utilizo muito, uma nova biblioteca. Claro, é cheio de biblioteca, pode se acostumar. A gente vai usar uma biblioteca que vai nos ajudar a capturar todos os campos do formulário. Que biblioteca é essa? O nome dela é `react-hook-form`. Vou mostrar para vocês.

Aqui está ela. Eu vou ensinar vocês a utilizá-la. Ela facilita muito na hora de pegar dados de um formulário. Vamos lá! Vou abrir meu terminal e digitar:

sh

 Copiar código


```
npm install react-hook-form
```

Pronto, já instalou. Se você quiser ter certeza que instalou, basta vir no `package.json` e ver quantas bibliotecas já baixamos. Isso aqui não é nem uma fração do que eu uso em projetos comerciais, mas já tá aumentando. Nossa aplicação é assim mesmo. Ótimo!

Como é que eu vou fazer o uso do `react-hook-form`? É desse jeito: vocês vão criar aqui comigo uma constante e vão abrir chaves. Vão dar um `useForm`. Dá um `Enter` para importar. Ótimo. A partir de agora, você pode pegar esse `useForm`. Esse hook é fornecido pelo `react-hook-form`. O retorno dele são algumas variáveis, e tem um objeto dentro dele. Eu vou desestruturar algumas coisas que vou precisar. A primeira coisa é o `register` e a segunda é o `handleSubmit`. Com esses dois já dá para fazer o que preciso.

Então, vou pegar e dizer o seguinte: bem aqui embaixo, vou criar uma função chamada `logar`. Ela vai receber um parâmetro, vou chamar de `dados`, e vou dar um log nesses dados para vocês verem o que vai vir dentro. Pronto. Agora vamos utilizar o `register` e o `handleSubmit`. O `register` é uma espécie de `useRef` que vamos atrelar a todos os nossos inputs. Assim:

jsx

 Copiar código

```
<input {...register('email', { required: true })} />
```

Se eu não quisesse que esse campo fosse obrigatório, poderia remover essa parte `{ required: true }`, mas eu quero que seja. Então, copie essa linha e cole no campo de senha, mudando para `senha`. Pronto! Meio caminho andado.

A segunda coisa que precisamos fazer é vir no `form` e chamar um evento. Quando clico no botão dentro do `form`, qual é o evento executado? O `onSubmit`. Quando o `onSubmit` for chamado, quero que utilize uma função, e a função é o `handleSubmit`. Passamos como parâmetro a função `logar`:

jsx

 Copiar código

```
<form onSubmit={handleSubmit(logar)}>
```

Quando o `handleSubmit` for chamado, ele vai passar as informações dos inputs como um objeto para a função `logar`. Nossa função `logar`, que só dá um `console.log`, vai mostrar um objeto com duas chaves: `email` e `senha`.

Vamos ver se está funcionando. Vou voltar para minha aplicação, abrir o console, digitar meu e-mail e senha, e clicar em "Entrar". O que apareceu? Um objeto com a chave `email` e a chave `senha`, cada uma com seus respectivos valores. Assim, fica muito mais fácil mandar essas informações numa requisição HTTP. Em vez de você ter que criar dois `useRefs`, montar o objeto e passar para uma função, o `react-hook-form` facilita tudo isso. Aconselho que vocês leiam a documentação, tem muita coisa massa para aprender lá. A gente se vê no próximo vídeo. Valeu, pessoal!

1.1. Vídeo da aula

dia 22 video 1



2. PrimeReact e React Context

Protegendo Rotas com PrimeReact e [React](#) Context

Introdução

Olá, pessoal! Tudo bem com vocês? Nesta aula, vamos aprender a proteger nossas rotas, garantindo que apenas usuários autenticados possam acessar certas páginas. Vamos usar o [React](#) Context para gerenciar o estado de autenticação e proteger as rotas da nossa aplicação.

Passo 1: Criar a Página Home

Vamos começar criando a página `Home` para ser acessada após o login.

Home.jsx

jsx

 Copiar código

```
import React from 'react'; const Home = () => { return ( <div> <h1>Bem-vindo à Home</h1> </div> ); }; export default Home;
```

Passo 2: Configurar as Rotas

Adicione a rota para a página Home no arquivo `Paths.jsx`.

Paths.jsx

jsx

 Copiar código

```
import React from 'react'; import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'; import Login from '../pages/Login'; import Home from '../pages/Home'; const Paths = () => { return ( <Router> <Routes> <Route path="/" element=<Login /> /> <Route path="/home" element=<Home /> /> </Routes> </Router> ); }; export default Paths;
```

Passo 3: Criar o Contexto de Autenticação

Vamos criar um contexto para gerenciar o estado de autenticação.

AuthContext.jsx

jsx

 Copiar código

```
import React, { createContext, useState, useContext } from 'react'; const AuthContext = createContext(); export const AuthProvider = ({ children }) => { const [isAuthenticated, setIsAuthenticated] = useState(false); return ( <AuthContext.Provider value={{ isAuthenticated, setIsAuthenticated }}> {children} </AuthContext.Provider> ); }; export const useAuth = () => { return useContext(AuthContext); };
```

Passo 4: Envolver a Aplicação com o AuthProvider

Envolva o componente `Paths` com o `AuthProvider` no `App.jsx`.

App.jsx

jsx

 Copiar código

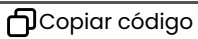
```
import React from 'react'; import Paths from './routes/Paths'; import { AuthProvider } from './contexts/AuthContext'; const App = () => { return ( <AuthProvider> <Paths /> </AuthProvider> ); }; export default App;
```

Passo 5: Proteger as Rotas

Vamos criar um componente para proteger as rotas.

PrivateRoute.jsx

jsx

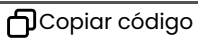


Copiar código

```
import React from 'react'; import { Navigate } from 'react-router-dom'; import { useAuth } from '../contexts/AuthContext';
const PrivateRoute = ({ children }) => { const { isAuthenticated } = useAuth(); return isAuthenticated ? children :
<Navigate to="/" /> }; export default PrivateRoute;
```

Atualize Paths.jsx para usar PrivateRoute

jsx



Copiar código

```
import React from 'react'; import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'; import Login from
'../pages/Login'; import Home from '../pages/Home'; import PrivateRoute from './PrivateRoute'; const Paths = () => { return
( <Router> <Routes> <Route path="/" element={<Login />} /> <Route path="/home" element={<PrivateRoute> <Home />
</PrivateRoute>} /> </Routes> </Router> ); }; export default Paths;
```

Passo 6: Autenticar o Usuário no Login

Vamos ajustar o componente `Login` para autenticar o usuário e redirecioná-lo para a página Home.

Login.jsx

jsx



Copiar código

```
import React, { useState } from 'react'; import { InputText } from 'primereact/inputtext'; import { Password } from
'primereact/password'; import { Button } from 'primereact/button'; import { useNavigate } from 'react-router-dom'; import {
useAuth } from '../contexts/AuthContext'; import 'primeflex/primeflex.css'; const Login = () => { const navigate =
useNavigate(); const { setIsAuthenticated } = useAuth(); const [email, setEmail] = useState(''); const [password,
setPassword] = useState(''); const handleSubmit = (event) => { event.preventDefault(); if (email === 'gleydson@gmail.com' &&
password === '123456') { setIsAuthenticated(true); navigate('/home'); } else { alert('Email ou senha incorretos'); } };
return ( <div className="p-d-flex p-jc-center p-ai-center p-h-screen p-bg-primary"> <div className="p-card p-shadow-5 p-p-4
p-text-center p-col-12 p-md-4"> <h3 className="p-text-uppercase p-text-bold p-text-lg">Seja Bem-vindo</h3> <form onSubmit=
{handleSubmit} className="p-fluid"> <div className="p-field p-mb-3"> <label htmlFor="email" className="p-text-sm p-text-bold
p-mb-2">Email</label> <InputText id="email" type="email" placeholder="email@example.com" className="p-inputtext-sm" value=
{email} onChange={(e) => setEmail(e.target.value)} /> </div> <div className="p-field p-mb-3"> <label htmlFor="password"
className="p-text-sm p-text-bold p-mb-2">Senha</label> <Password id="password" placeholder="*****" toggleMask feedback=
{false} className="p-inputtext-sm" value={password} onChange={(e) => setPassword(e.target.value)} /> </div> <Button
label="Entrar" type="submit" className="p-button-sm p-button-primary p-mt-3" /> </form> </div> </div> ); }; export default
Login;
```

Explicação dos Componentes

- **AuthContext:** Gerencia o estado de autenticação da aplicação.
- **PrivateRoute:** Componente que verifica se o usuário está autenticado antes de permitir o acesso à rota.
- **Login:** Autentica o usuário e armazena o estado de autenticação no contexto.

Testando a Aplicação

1. Tente acessar `/home` diretamente. Você será redirecionado para a página de login.
2. Faça login com as credenciais corretas (`gleydson@gmail.com` e `123456`).
3. Após o login bem-sucedido, você será redirecionado para a página Home.

Conclusão

Nesta aula, aprendemos a proteger rotas na nossa aplicação `React` utilizando `React` Context e `PrimeReact`. Implementamos um fluxo de autenticação simples e garantimos que apenas usuários autenticados possam acessar certas páginas.

Até a próxima aula!

2.1. Vídeo da aula

dia 22 video 2

