Desenvolvimento de Formulários em React: Um Guia Prático

Por Escola Dnc

Introdução

Este ebook aborda o desenvolvimento de formulários em React, focando na criação de uma interface de contato funcional e estilizada. Vamos explorar desde a estruturação básica do HTML até a estilização detalhada com CSS, passando por boas práticas de desenvolvimento e integração com APIs REST.

1. Estruturação Básica do Formulário

Nesta seção, abordaremos a criação da estrutura HTML do nosso formulário de contato.

Elementos do Formulário

O formulário é composto pelos seguintes elementos principais:

- Input para nome
- Input para e-mail
- Textarea para mensagem
- Botão de envio

Código Base

Aqui está o código base para o nosso formulário:

```
<form className="deflex form-group">  <input     className="form-input"     type="text"     id="name"     name="name"     placeholder="Nome*"   />  <input     className="form-input"     type="email"     id="email"     name="email"     placeholder="E-mail*"   />  <textarea     className="form-input"     id="message"     name="message"     placeholder="Mensagem"     rows="5"  >  </textarea>  <div className="form-group">    <Button type="submit"     buttonStyle="secondary">      Enviar    </Button>  </div></form>
```

Observações importantes:

- Utilizamos `className` em vez de `class` por estarmos trabalhando com React.
- O atributo `type="email"` no input de e-mail fornece uma validação básica.

- O atributo `rows` no textarea define o número de linhas visíveis.

2. Estilização do Formulário

A estilização é crucial para criar uma experiência de usuário agradável e coerente com o design do projeto.

Estilos Gerais

Definimos alguns estilos gerais para o formulário:

```
.contact-form {  margin: 40px 0 0 0;  width: 100%;}.contact-form p
{  font-size: 20px;  margin: 0;}.contact-form .form-group {
column-gap: 30px;  margin: 40px 0 0 0;  display: flex;}
```

Estilos dos Inputs

Para os inputs e textarea, aplicamos os seguintes estilos:

```
.form-input {  border: none;  border-bottom: 1px solid #29292E;
color: #29292E;  font-size: 22px;  font-family: 'JOST', sans-serif;
padding: 0 0 13px;  outline: none;  width: 100%;}.form-
input::placeholder {  color: #29292E;  opacity: 1;}
```

Pontos importantes:

- Removemos as bordas padrão e adicionamos apenas uma borda inferior.
- Utilizamos `outline: none` para remover o contorno padrão do navegador ao focar.
- Personalizamos o estilo do placeholder.

Alinhamento do Botão

Para alinhar o botão à direita, usamos:

```
.form-group:last-child {  display: flex;  justify-content: flex-
end;  align-items: center;}
```

3. Integração com API REST

Nesta seção, discutiremos como integrar nosso formulário com uma API REST para envio dos dados.

Preparação para Requisições

Antes de implementar a lógica de envio, é importante preparar o estado do componente:

```
import React, { useState } from 'react';function ContactForm() {
const [formData, setFormData] = useState({    name: '',    email:
'',    message: ''  });  const handleChange = (e) => {
setFormData({...formData, [e.target.name]: e.target.value});  };
// ... resto do código}
```

Implementação do Envio

Para enviar os dados do formulário, podemos usar a função `fetch`:

```
const handleSubmit = async (e) => {  e.preventDefault();  try {
const response = await fetch('URL_DA_SUA_API', {      method:
'POST',      headers: {        'Content-Type': 'application/json',
},      body: JSON.stringify(formData),    });    if (response.ok)
{      alert('Mensagem enviada com sucesso!');      setFormData({
name: '', email: '', message: '' });    } else {      throw new
Error('Falha no envio');    }  } catch (error) {    alert('Erro ao
enviar mensagem: ' + error.message);  }};
```

Observações:

- Lembre-se de substituir 'URL_DA_SUA_API' pela URL real da sua API.
- Este código inclui tratamento básico de erros e feedback ao usuário.

4. Boas Práticas e Otimizações

Para finalizar, vamos abordar algumas boas práticas e otimizações para nosso formulário.

Validação de Campos

Implemente validação de campos para melhorar a experiência do usuário:

```
const validate = () => {  const errors = {};  if (!formData.name.trim()) errors.name = "Nome é obrigatório";  if (!formData.email.trim()) errors.email = "E-mail é obrigatório";  if (!/\S+@\S+\.\S+/.test(formData.email)) errors.email = "E-mail inválido";  return errors;};
```

Feedback Visual

Adicione feedback visual para campos inválidos:

```
.form-input.error {  border-bottom-color: #ff0000;}.error-message {  color: #ff0000;  font-size: 14px;  margin-top: 5px;}
```

Acessibilidade

Melhore a acessibilidade do seu formulário:

- Adicione labels associados aos inputs
- Use atributos `aria-` para descrever campos e erros
- Garanta que o formulário seja navegável por teclado

```
<label htmlFor="name">Nome:</label><input   id="name"  name="name"
aria-required="true"  aria-invalid={errors.name ? "true" : "false"}
aria-describedby="name-error"/>{errors.name && <span id="name-
error" className="error-message">{errors.name}</span>}
```

Conclusão

Neste ebook, exploramos o processo de criação de um formulário de contato em React, desde a estruturação básica até a integração com APIs e implementação de boas práticas. Lembre-se de que um bom formulário não é apenas funcional, mas também oferece uma experiência agradável e acessível ao usuário. Continue praticando e iterando sobre esses conceitos para aprimorar suas habilidades de desenvolvimento front-end.