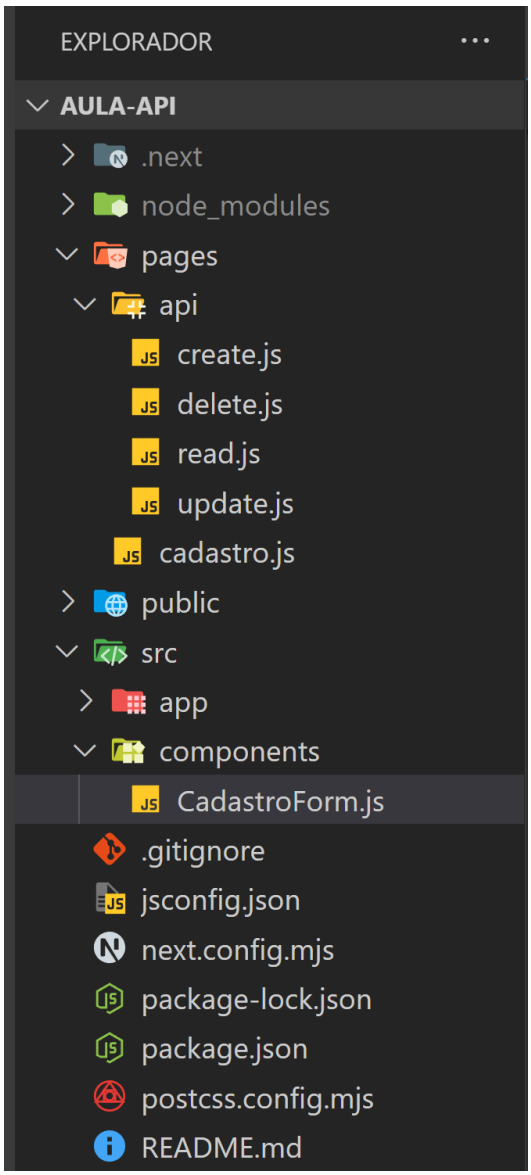
CRIANDO UMA VIEW NO NEXT.JS

Por João Siles

Olá amigos! Para prosseguir nossa aula precisamos ter criado as APIs de CRUD como fizemos na última aula.

Primeiro, vamos criar o componente da view com a pasta **components** dentro de **src** (se ainda não existir). Dentro da pasta **components**, crie um arquivo chamado **CadastroForm.js** (ou o nome que preferir).



Adicione o seguinte código ao arquivo **CadastroForm.js**:

```jsx
import React, { useState } from 'react';

const CadastroForm = () => {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  // Removendo o estado da senha

  const handleSubmit = async (event) => {
    event.preventDefault();

    try {
      const response = await fetch('/api/create', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ name, email }), // Enviando apenas nome e email
      });

      if (response.ok) {
        console.log('Cadastro realizado com sucesso!');
        // Aqui você pode adicionar alguma lógica de redirecionamento ou feedback ao usuário
      } else {
        console.error('Erro ao cadastrar:', response.status);
        // Aqui você pode adicionar alguma lógica de tratamento de erro
      }
    } catch (error) {
      console.error('Ocorreu um erro:', error);
      // Aqui você pode adicionar alguma lógica de tratamento de erro geral
    }
  };

  return (
    <div className="flex items-center justify-center min-h-screen bg-gray-100">
      <div className="bg-white p-8 rounded shadow-md w-full max-w-md">
        <h2 className="text-2xl font-semibold mb-6 text-gray-800">Cadastro</h2>
        <form onSubmit={handleSubmit}>
          <div className="mb-4">
            <label htmlFor="nome" className="block text-gray-700 text-sm font-bold mb-2">
              Nome:
            </label>
            <input
              type="text"
              id="nome"
              className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"
              value={name}
              onChange={(e) => setName(e.target.value)}
            />
          </div>
          <div className="mb-4">
            <label htmlFor="email" className="block text-gray-700 text-sm font-bold mb-2">
              Email:
            </label>
            <input
              type="email"
              id="email"
              className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
            />
          </div>
          {/* Removendo o campo de senha */}
          <div className="flex items-center justify-between">
            <button
              className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline"
              type="submit"
            >
              Cadastrar
            </button>
          </div>
        </form>
      </div>
    </div>
  );
};

export default CadastroForm;
```

Explicação do Código:

- **import React, { useState } from 'react';**: Importamos o React e o useState para gerenciar o estado dos campos do formulário.
- **const [nome, setNome] = useState('');**, **const [email, setEmail] = useState('');**, **const [senha, setSenha] = useState('');**: Declaramos variáveis de estado para cada campo do formulário, inicializando-as com *strings* vazias.
- **handleSubmit**: Esta função é chamada quando o formulário é enviado.
    - *event.preventDefault()*; impede o comportamento padrão de envio do formulário (recarregar a página).
    - *fetch('/api/create, ...)* faz uma requisição POST para a sua API de cadastro (/pages/api/create.js).
    - Definimos os *headers* para indicar que estamos enviando dados JSON.
    - *JSON.stringify({ name, email })* converte os dados do formulário em uma string JSON para enviar no corpo da requisição.
    - Verificamos se a resposta da API foi bem-sucedida (response.ok). Se sim, exibimos uma mensagem no console (você pode adicionar uma lógica mais elaborada aqui, como redirecionar o usuário). Caso contrário, exibimos um erro.
    - Utilizamos um bloco **try...catch** para lidar com possíveis erros durante a requisição.
- **Estrutura do Formulário (return (...))**:
    - Utilizamos classes do **Tailwind CSS** para estilizar o contêiner principal (*flex, items-center, justify-center, min-h-screen, bg-gray-100*) para centralizar o formulário na tela.
    - O *div* interno (*bg-white, p-8, rounded, shadow-md, w-full, max-w-md*) define o fundo branco, *padding*, bordas arredondadas, sombra e largura máxima do formulário.
    - O *h2* estiliza o título do formulário.

- Dentro do <*form*>, cada campo (*label* e i*nput*) é envolvido em um div com margem inferior (*mb-4* ou *mb-6*).
- As classes do **Tailwind** são aplicadas aos elementos *label* e *input* para definir a aparência (texto cinza, tamanho da fonte, negrito, sombra, bordas, padding, etc.).
- O botão de "Cadastrar" também é estilizado com classes do Tailwind para cor de fundo, cor do texto, *padding*, bordas arredondadas e efeitos de *hover* e foco.

Dentro da pasta **pages**, você pode criar um arquivo chamado **cadastro.js**:

```
import CadastroForm from '../src/components/CadastroForm';

const CadastroPage = () => {
  return (
    <div>
      <h1 className="text-3xl font-bold mb-4 text-center">Página de Cadastro</h1>
      <CadastroForm />
    </div>
  );
};

export default CadastroPage;
```

- Importamos o componente **CadastroForm** que criamos anteriormente.
- Definimos um componente funcional **CadastroPage** que renderiza um título e o formulário de cadastro.
- Aplicamos algumas classes básicas do **Tailwind** ao título para estilização.

Como usar:

1. Certifique-se de que o Tailwind CSS esteja corretamente configurado em seu projeto Next.js. Você provavelmente já fez isso, mas caso contrário, siga as instruções na documentação oficial do Tailwind CSS.
2. Execute seu servidor de desenvolvimento do Next.js (*npm run dev* ou *yarn dev*).
3. Acesse a página que você criou para o formulário (por exemplo, http://localhost:3000/cadastro).

Agora você terá um formulário de cadastro básico estilizado com Tailwind CSS que envia os dados para a sua API *create,* falta agora criar páginas ou elementos que façam outras operações! Fica o desafio para a próxima semana!