Projeto CRUD com ReactJS e TypeScript (Sem React.FC)

Este documento descreve a criação de um projeto básico de CRUD utilizando ReactJS, TypeScript, e Vite,
sem a utilização do React.FC. O projeto inclui operações de listagem, adição, edição e remoção de produtos,
e utiliza o json-server como API simulada.

1. Configuração do Projeto

1.1 Instalação do Vite com React e TypeScript:

```
npm create vite@latest my-crud-app --template react-ts
cd my-crud-app
npm install
```

1.2 Instalação das Dependências Necessárias:

```
npm install axios react-router-dom
```

1.3 Configuração do json-server:

Crie um arquivo db.json na raiz do projeto:

```
{
```

```
  "products": [
    {
      "id": "1",
      "name": "Produto 1",
      "description": "Descrição do produto 1",
      "price": 100,
      "quantity": 10
    }
  ]
}
```

Para iniciar o servidor:

```
npx json-server --watch db.json --port 3001
```

2. Estrutura do Projeto

Aqui está a estrutura básica que você pode seguir:

```
src/
components/
    ProductList.tsx
    ProductForm.tsx
    ProductItem.tsx

pages/
    Home.tsx
```

EditProduct.tsx

AddProduct.tsx

services/

api.ts

App.tsx

main.tsx

3. Implementação dos Componentes

3.1 api.ts (Serviço de API)

```
import axios from 'axios';

const api = axios.create({
    baseURL: 'http://localhost:3001'
});

export const getProducts = () => api.get('/products');
export const getProductById = (id: string) => api.get(`/products/${id}`);
export const createProduct = (product: any) => api.post('/products', product);
export const updateProduct = (id: string, product: any) => api.put(`/products/${id}`, product);
export const deleteProduct = (id: string) => api.delete(`/products/${id}`);
```

3.2 ProductList.tsx

```
import { useEffect, useState } from 'react';
import { getProducts, deleteProduct } from '../services/api';
import { Link } from 'react-router-dom';

interface Product {
    id: string;
    name: string;
    description: string;
    price: number;
    quantity: number;
}

function ProductList() {
    const [products, setProducts] = useState<Product[]>([]);

    useEffect(() => {
        loadProducts();
    }, []);

    const loadProducts = async () => {
        const response = await getProducts();
        setProducts(response.data);
    };

    const handleDelete = async (id: string) => {
        await deleteProduct(id);
        loadProducts();
```

```
    };

    return (
        <div>
            <h1>Product List</h1>
            <Link to="/add">Add Product</Link>
            <ul>
                {products.map((product) => (
                    <li key={product.id}>
                        {product.name} - ${product.price} - {product.quantity} units
                        <Link to={`/edit/${product.id}`}>Edit</Link>
                        <button onClick={() => handleDelete(product.id)}>Delete</button>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export default ProductList;
```

3.3 ProductForm.tsx

```
import { useState, useEffect } from 'react';
import { useNavigate, useParams } from 'react-router-dom';
import { createProduct, getProductById, updateProduct } from '../services/api';
```

```
interface Product {
    name: string;
    description: string;
    price: number;
    quantity: number;
}

function ProductForm() {
    const { id } = useParams<{ id: string }>();
    const navigate = useNavigate();
    const [product, setProduct] = useState<Product>({
        name: '',
        description: '',
        price: 0,
        quantity: 0,
    });

    useEffect(() => {
        if (id) {
            loadProduct();
        }
    }, [id]);

    const loadProduct = async () => {
        try {
            const response = await getProductById(id as string);
            setProduct(response.data);
```

```
    } catch (error) {
        console.error("Error loading product data", error);
    }
};

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setProduct({
        ...product,
        [e.target.name]: e.target.value,
    });
};

const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    try {
        if (id) {
            await updateProduct(id, product);
        } else {
            await createProduct(product);
        }
        navigate('/');
    } catch (error) {
        console.error("Error saving product", error);
    }
};

return (
```

```
<form onSubmit={handleSubmit}>
    <div>
        <label>Name</label>
        <input
            type="text"
            name="name"
            value={product.name}
            onChange={handleChange}
        />
    </div>
    <div>
        <label>Description</label>
        <input
            type="text"
            name="description"
            value={product.description}
            onChange={handleChange}
        />
    </div>
    <div>
        <label>Price</label>
        <input
            type="number"
            name="price"
            value={product.price}
            onChange={handleChange}
        />
```

```
            </div>
            <div>
                <label>Quantity</label>
                <input
                    type="number"
                    name="quantity"
                    value={product.quantity}
                    onChange={handleChange}
                />
            </div>
            <button type="submit">Save</button>
        </form>
    );
}

export default ProductForm;
```

3.4 Home.tsx

```
import ProductList from '../components/ProductList';

function Home() {
    return (
        <div>
            <h1>Product Management</h1>
            <ProductList />
        </div>
```

```
    );
}

export default Home;
```

3.5 AddProduct.tsx

```
import ProductForm from '../components/ProductForm';

function AddProduct() {
    return (
        <div>
            <h1>Add Product</h1>
            <ProductForm />
        </div>
    );
}

export default AddProduct;
```

3.6 EditProduct.tsx

```
import ProductForm from '../components/ProductForm';

function EditProduct() {
    return (
        <div>
```

```
            <h1>Edit Product</h1>
            <ProductForm />
        </div>
    );
}

export default EditProduct;
```

3.7 App.tsx

```
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Home from './pages/Home';
import AddProduct from './pages/AddProduct';
import EditProduct from './pages/EditProduct';

function App() {
    return (
        <Router>
            <Routes>
                <Route path="/" element={<Home />} />
                <Route path="/add" element={<AddProduct />} />
                <Route path="/edit/:id" element={<EditProduct />} />
            </Routes>
        </Router>
    );
}
```

```
export default App;
```

4. Executando o Projeto

- Certifique-se de que o json-server está em execução.
- Execute o comando npm run dev para iniciar o Vite.

Agora você tem um aplicativo básico de CRUD para produtos utilizando ReactJS, TypeScript, Vite e json-server. O aplicativo está dividido em componentes para melhor organização e manutenção do código.