Exemplo: Fetch de Dados e Exibição Condicional

Neste exemplo, vamos criar dois componentes:

1. **UserList:** Este componente faz uma requisição para uma API fictícia e exibe uma lista de usuários.
2. **ToggleMessage:** Este componente alterna a exibição de uma mensagem quando um botão é clicado.

Passo 1: Configuração do Projeto

1. **Crie um novo projeto React com TypeScript usando Vite:**

bash

Copiar código

```
npm create vite@latest outro-projeto -- --template react-ts

cd outro-projeto

npm install

npm run dev
```

2. **Estrutura do Projeto:**

Certifique-se de que a estrutura básica do projeto é semelhante a esta:

arduino

Copiar código

```
outro-projeto/
├── node_modules/
├── public/
├── src/
│   ├── App.tsx
│   ├── index.css
│   ├── main.tsx
│   └── components/
│       ├── UserList.tsx
│       └── ToggleMessage.tsx
├── .gitignore
├── index.html
├── package.json
```

├── tsconfig.json

└── vite.config.ts

Passo 2: Implementação dos Componentes

1. **Componente UserList (UserList.tsx):**

Este componente faz uma requisição fetch para uma API fictícia (neste caso, utilizaremos um exemplo de uma API estática) e exibe os nomes dos usuários.

typescript

Copiar código

```
import { useState, useEffect } from 'react';


function UserList() {
  const [users, setUsers] = useState<string[]>([]);
  const [loading, setLoading] = useState<boolean>(true);


  useEffect(() => {
    fetch('https://jsonplaceholder.typicode.com/users')
      .then(response => response.json())
      .then(data => {
        const usernames = data.map((user: any) => user.name);
        setUsers(usernames);
        setLoading(false);
      })
      .catch(error => {
        console.error('Erro ao buscar os usuários:', error);
        setLoading(false);
      });
  }, []);


  if (loading) {
    return <p>Carregando usuários...</p>;
  }
```

```
  return (
    <div>
      <h2>Lista de Usuários</h2>
      <ul>
        {users.map(user => (
          <li key={user}>{user}</li>
        ))}
      </ul>
    </div>
  );
}

export default UserList;
```

2. **Componente ToggleMessage (ToggleMessage.tsx):**

Este componente exibe ou oculta uma mensagem quando um botão é clicado. O useEffect é usado para exibir uma mensagem no console sempre que o componente é renderizado.

typescript

Copiar código

```
import { useState, useEffect } from 'react';

function ToggleMessage() {
  const [showMessage, setShowMessage] = useState<boolean>(false);

  useEffect(() => {
    console.log('O componente ToggleMessage foi renderizado.');
  });

  return (
    <div>
      <button onClick={() => setShowMessage(!showMessage)}>
```

```
        {showMessage ? 'Ocultar Mensagem' : 'Exibir Mensagem'}
      </button>
      {showMessage && <p>Esta é a mensagem!</p>}
    </div>
  );
}

export default ToggleMessage;
```

3. **Componente Principal (App.tsx):**

O componente principal App importará e renderizará os componentes UserList e ToggleMessage.

typescript

Copiar código

```
import './App.css';
import UserList from './components/UserList';
import ToggleMessage from './components/ToggleMessage';

function App() {
  return (
    <div className="App">
      <h1>Exemplo de React com TypeScript</h1>
      <UserList />
      <ToggleMessage />
    </div>
  );
}

export default App;
```

Passo 3: Executar o Projeto

1. **Execute o servidor de desenvolvimento:**

bash

Copiar código

npm run dev

2. **Verifique a aplicação no navegador:**

Acesse o endereço fornecido pelo Vite (geralmente http://localhost:3000) para ver os componentes em ação.

Explicação do Código

- **UserList Component:**
    - Usa useState para manter o estado dos usuários e do carregamento.
    - Usa useEffect para buscar dados de uma API quando o componente é montado. A lista de usuários é exibida após os dados serem carregados.
- **ToggleMessage Component:**
    - Usa useState para controlar a exibição de uma mensagem.
    - Usa useEffect para exibir uma mensagem no console sempre que o componente é renderizado. A mensagem pode ser alternada entre visível e oculta.
- **App Component:**
    - Importa e renderiza UserList e ToggleMessage, compondo a interface da aplicação.

Este exemplo mostra como usar useEffect para lidar com efeitos colaterais em React, como buscar dados de uma API e exibir mensagens condicionais, utilizando componentes separados e organizados.