Projeto Prático: Consulta de Saldo e Transações com TypeScript e Web 3.0

Objetivo: Desenvolver uma aplicação em TypeScript que permita ao usuário consultar o saldo de uma carteira Ethereum e visualizar suas últimas transações.

Ferramentas e Bibliotecas Necessárias:

- **Node.js**
- **TypeScript**
- **Ethers.js** (biblioteca para interação com a blockchain Ethereum)
- **Webpack** (para compilar o projeto)

Instruções do Projeto

1. Estrutura do Projeto

Organize o projeto conforme a estrutura abaixo:

```
projeto-web3/
├── src/
│   ├── index.ts
│   └── app.ts
├── index.html
├── tsconfig.json
├── webpack.config.js
```

---

2. Instalação e Configuração

1. **Crie a pasta do projeto e inicialize o projeto Node.js**:

   ```
   mkdir projeto-web3
   ```

```
cd projeto-web3
npm init -y
```

2. **Instale as dependências**:

```
npm install typescript ethers webpack webpack-cli webpack-dev-
server html-webpack-plugin --save-dev
```

3. **Inicialize o TypeScript**:

```
npx tsc --init
```

4. **Atualize o arquivo tsconfig.json**:

```
{
  "compilerOptions": {
    "target": "ES6",
    "module": "ES6",
    "strict": true,
    "esModuleInterop": true,
    "outDir": "./dist",
    "sourceMap": true
  },
  "include": ["src/**/*"]
}
```

---

3. Configuração do Webpack

No arquivo webpack.config.js, insira o seguinte código para configurar o Webpack:

```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  entry: "./src/index.ts",
  module: {
    rules: [
      {
        test: /\.ts$/,
        use: "ts-loader",
        exclude: /node_modules/,
      },
    ],
  },
  resolve: {
    extensions: [".ts", ".js"],
  },
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "dist"),
    clean: true,
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: "./index.html",
    }),
  ],
```

```
  devServer: {
    static: "./dist",
    port: 3000,
    open: true,
  },
};
```

---

4. Código HTML

No arquivo index.html, adicione o layout básico da aplicação:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Consulta de Saldo e Transações - Ethereum</title>
</head>
<body>
    <h1>Consulta de Saldo e Transações</h1>
    <input type="text" id="wallet-address" placeholder="Digite o
endereço da carteira" />
    <button id="check-balance">Consultar Saldo</button>
    <button id="check-transactions">Ver Transações</button>
    <p id="balance"></p>
    <div id="transactions"></div>
    <script src="bundle.js"></script>
</body>
</html>
```

---

5. Implementação em TypeScript

No arquivo app.ts, insira o código para buscar saldo e transações:

```
import { ethers } from "ethers";

const provider = ethers.getDefaultProvider("mainnet");

export function initApp() {
    const walletInput = document.getElementById("wallet-address") as
HTMLInputElement;
    const balanceDisplay = document.getElementById("balance") as
HTMLParagraphElement;
    const transactionsDisplay =
document.getElementById("transactions") as HTMLDivElement;
    const checkBalanceButton = document.getElementById("check-
balance") as HTMLButtonElement;
    const checkTransactionsButton = document.getElementById("check-
transactions") as HTMLButtonElement;

    checkBalanceButton.addEventListener("click", async () => {
        const address = walletInput.value.trim();
        if (!ethers.utils.isAddress(address)) {
            balanceDisplay.textContent = "Endereço inválido!";
            return;
```

```
        }

        try {
            const balance = await provider.getBalance(address);
            balanceDisplay.textContent = `Saldo:
${ethers.utils.formatEther(balance)} ETH`;
        } catch (error) {
            balanceDisplay.textContent = "Erro ao buscar o saldo.";
            console.error(error);
        }
    });

    checkTransactionsButton.addEventListener("click", async () => {
        const address = walletInput.value.trim();
        if (!ethers.utils.isAddress(address)) {
            transactionsDisplay.textContent = "Endereço inválido!";
            return;
        }

        try {
            const history = await provider.getHistory(address);
            transactionsDisplay.innerHTML = "<h3>Últimas
Transações:</h3>";
            history.slice(0, 5).forEach(tx => {
                const txElement = document.createElement("p");
                txElement.textContent = `De: ${tx.from} Para: ${tx.to}
- Valor: ${ethers.utils.formatEther(tx.value)} ETH`;
                transactionsDisplay.appendChild(txElement);
            });
        } catch (error) {
            transactionsDisplay.textContent = "Erro ao buscar as
transações.";
            console.error(error);
        }
    });
}
```

No `index.ts`, importe a função `initApp` e inicialize a aplicação:

```
import { initApp } from "./app";

document.addEventListener("DOMContentLoaded", () => {
    initApp();
});
```

---

6. Executando o Projeto

1. **Compile o projeto com Webpack**:

   ```
   npm run build
   ```

2. **Inicie o servidor de desenvolvimento**:

   ```
   npm start
   ```

3. **Acesse a aplicação** em http://localhost:3000 para testar.

Desafios Extras

1. **Adicionar Verificação de Endereço**: Melhore a validação de entrada do endereço Ethereum.
2. **Exibir Data das Transações**: Adicione a data e hora de cada transação listada.
3. **Múltiplas Redes**: Permita que o usuário escolha entre diferentes redes (mainnet, rinkeby, etc.).
4. **Paginação de Transações**: Adicione uma funcionalidade para carregar mais transações, em blocos de cinco.

Este projeto simples e prático é uma introdução ao desenvolvimento Web 3.0 com TypeScript e interações na blockchain.

Pode ser que os SCRIPTS apresentem pequenos erros e precisem ser revisados, é AÍ QUE VOCÊ ENTRA!!!

Experimente os desafios adicionais para aprimorar sua aplicação e reforçar o aprendizado.

Vamos Juntos e Curta o PROCESSO!