Faustino Aguilar faustinoaq@gmail.com

Bankathon by Towerbank Codigo

"SISTEMA PARA AUTOMATIZAR ÓRDENES"

Este desafío tiene por objetivo crear un sistema que permita a clientes del banco poder canalizar órdenes hacia diferentes plataformas de intercambio de criptomonedas para encontrar y ejecutar los mejores precios de compra y de venta.

Repositorio de Github

https://github.com/faustinoaq/Bankaton2023

Codigo Inicial Utilizado

1. **Importaciones:** Importamos los componentes necesarios de la biblioteca Ant Design, como Table, Button y Typography, que utilizaremos para crear la interfaz de usuario con estilo.
2. **Hook de Estado:** Utilizamos los hooks de estado de React para almacenar y actualizar los datos que recibimos de la API. accountBalance guarda el saldo de la cuenta, providers almacena la lista de proveedores, y transactionResult guarda el resultado de una transacción. **Ver Página siguiente**

```
import React, { useEffect, useState } from 'react';
import { Table, Button, Typography } from 'antd';

const { Title } = Typography;

function App() {
  const [accountBalance, setAccountBalance] = useState(null);
  const [providers, setProviders] = useState([]);
  const [transactionResult, setTransactionResult] = useState(null);
  const [loading, setLoading] = useState(true);
  const API_KEY =
    '86e7b63e-a39a-4774-90eb-c1fdb364fed0!f09908602ebc2743940871375d940d9d6ed7dc0d805e3f4544b4c397f1a91b46dcbd6af6c18c29';

  useEffect(() => {
    // Fetch account balance
    fetch('https://towerbank.bankathontb.com/bankathon/v1/account', {
      headers: {
        'Content-Type': 'application/json',
        Authorization: API_KEY,
      },
    })
      .then((response) => response.json())
      .then((data) => {
        setAccountBalance(data.balance);
        setLoading(false);
      })
      .catch((error) => {
        console.error('Error fetching account balance:', error);
        setLoading(false);
      });

    // Fetch providers
    fetch('https://towerbank.bankathontb.com/bankathon/v1/providers', {
      headers: {
        'Content-Type': 'application/json',
        Authorization: API_KEY,
      },
    })
      .then((response) => response.json())
      .then((data) => setProviders(data.data))
      .catch((error) => console.error('Error fetching providers:', error));
  }, []);
```

1. **Hook de Efecto:** Utilizamos el hook de efecto useEffect para realizar las solicitudes a la API cuando el componente se monta. En este caso, hacemos una solicitud para obtener el saldo de la cuenta y otra solicitud para obtener la lista de proveedores. Al recibir los datos, actualizamos los estados correspondientes.
2. **Manejo de transacción:** La función handleTransaction se ejecuta cuando se hace clic en el botón "Make Transaction". En esta función, creamos los datos de la transacción y realizamos una solicitud POST a la API para ejecutar la transacción. El resultado de la transacción se almacena en el estado transactionResult.

```
const handleTransaction = () => {
  const transactionData = {
    accountId: '18400aee-00a5-11ee-be56-0242ac120002',
    amount: 100.0,
    transactionType: 'purchase',
  };

  fetch('https://towerbank.bankathontb.com/bankathon/v1/transaction', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      Authorization: API_KEY,
    },
    body: JSON.stringify(transactionData),
  })
    .then((response) => response.json())
    .then((data) => setTransactionResult(data))
    .catch((error) => console.error('Error performing transaction:', error));
};

const columns = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
  },
  {
    title: 'Balance',
    dataIndex: 'balance',
    key: 'balance',
  },
];
                                                                                }
```

1. **Diseño de la interfaz de usuario:** Utilizamos los componentes de Ant Design para estructurar y mostrar los datos en la interfaz. Mostramos el saldo de la cuenta, la lista de proveedores en una tabla y un botón para realizar la transacción. Si se realiza una transacción, se muestra el resultado en la parte inferior.
2. **Estilos:** Utilizamos el atributo style en los elementos HTML para aplicar estilos personalizados. Por ejemplo, aplicamos espaciado, márgenes y tamaños de fuente a través de estilos en línea.

```
  return (
    <div style={{ padding: '20px' }}>
      {loading ? (
        <p>Loading...</p>
      ) : (
        <>
          <Title level={3}>Account Balance: {accountBalance}</Title>
          <Title level={4}>Providers:</Title>
          <Table dataSource={providers} columns={columns} pagination={false} />

          <Title level={4} style={{ marginTop: '20px' }}>
            Perform Transaction:
          </Title>
          <Button type="primary" onClick={handleTransaction}>
            Make Transaction
          </Button>

          {transactionResult && (
            <div style={{ marginTop: '20px' }}>
              <Title level={4}>Transaction Result:</Title>
              <p>Transfer ID: {transactionResult.transferId}</p>
              <p>New Account Balance: {transactionResult.balance}</p>
            </div>
          )}
        </>
      )}
    </div>
  );
}

export default App;
```