Rakamin Academy

React Native

Case Study: Mobile Application for Checking Customer Account Balances with React Native - Intermediate

Raih Mimpi #TanpaBatas

Objektif sesi

- Peserta mengimplementasikan react native yang telah dipelajari dengan membuat Mobile Application for Checking Customer Account Balances with React Native - Level Intermediate

Deskripsi

Pada tingkat ini, tugas Anda adalah memperluas aplikasi sebelumnya untuk mencakup fitur pencatatan transaksi keuangan. Aplikasi ini harus memungkinkan pengguna untuk menambahkan, melihat, memperbarui, dan menghapus transaksi mereka. Selain itu, Anda akan belajar dan menerapkan konsep-konsep baru seperti penggunaan Promises, async/await, dan penggunaan REST API untuk berinteraksi dengan server.

Langkah-Langkah

1. Buat entitas Java untuk merepresentasikan transaksi keuangan
2. Buat repositori Spring Data JPA untuk berinteraksi dengan basis data
3. Buat kontroler untuk menangani permintaan HTTP
4. Pastikan konfigurasi basis data di file application.properties atau application.yml sesuai dengan pengaturan basis data Anda.
5. Jalankan aplikasi Spring Boot Anda dan pastikan tidak ada kesalahan pada konsol.

Langkah-Langkah

Implementasi Fetch dengan Axios di React Native

jsx Copy code

```jsx
// TransactionService.js
import axios from 'axios';

const API_URL = 'http://localhost:8080/transactions'; // Sesuaikan deng

const getAllTransactions = async () => {
  try {
    const response = await axios.get(API_URL);
    return response.data;
  } catch (error) {
    console.error('Error fetching transactions:', error);
    throw error;
  }
};

const addTransaction = async (newTransaction) => {
  try {
    const response = await axios.post(API_URL, newTransaction);
    return response.data;
  } catch (error) {
    console.error('Error adding transaction:', error);
    throw error;
  }
};

const updateTransaction = async (transactionId, updatedTransaction) =>
  try {
    const response = await axios.put(`${API_URL}/${transactionId}`, upd
    return response.data;
  } catch (error) {
    console.error('Error updating transaction:', error);
    throw error;
  }
};
```

```jsx
const deleteTransaction = async (transactionId) => {
  try {
    await axios.delete(`${API_URL}/${transactionId}`);
  } catch (error) {
    console.error('Error deleting transaction:', error);
    throw error;
  }
};

export { getAllTransactions, addTransaction, updateTransaction, deleteT
```

Langkah-Langkah

Gunakan service yang telah dibuat dalam komponen React Native

jsx Copy code

```jsx
// src/screens/HomeScreen.js
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, TouchableOpacity, FlatList } from 'rea
import { getAllTransactions, addTransaction, updateTransaction, deleteT

const HomeScreen = () => {
  const [transactions, setTransactions] = useState([]);

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

  const fetchTransactions = async () => {
    try {
      const data = await getAllTransactions();
      setTransactions(data);
    } catch (error) {
      console.error('Error fetching transactions:', error);
    }
  };

  const handleAddTransaction = async () => {
    const newTransaction = { description: 'New Transaction', amount: 5
    try {
      await addTransaction(newTransaction);
      fetchTransactions();
    } catch (error) {
      console.error('Error adding transaction:', error);
    }
  };
```

```jsx
  const handleAddTransaction = async () => {
    const newTransaction = { description: 'New Transaction', amount: 5
    try {
      await addTransaction(newTransaction);
      fetchTransactions();
    } catch (error) {
      console.error('Error adding transaction:', error);
    }
  };

  const handleUpdateTransaction = async (id, updatedTransaction) => {
    try {
      await updateTransaction(id, updatedTransaction);
      fetchTransactions();
    } catch (error) {
      console.error('Error updating transaction:', error);
    }
  };

  const handleDeleteTransaction = async (id) => {
    try {
      await deleteTransaction(id);
      fetchTransactions();
    } catch (error) {
      console.error('Error deleting transaction:', error);
    }
  };
```

```jsx
  return (
    <View style={styles.container}>
      <FlatList
        data={transactions}
        keyExtractor={(item) => item.id.toString()}
        renderItem={({ item }) => (
          <View style={styles.transactionItem}>
            <Text>{item.description}</Text>
            <Text>${item.amount}</Text>
            <TouchableOpacity onPress={() => handleUpdateTransaction(it
              <Text>Update</Text>
            </TouchableOpacity>
            <TouchableOpacity onPress={() => handleDeleteTransaction(it
              <Text>Delete</Text>
            </TouchableOpacity>
          </View>
        )}
      />
      <TouchableOpacity style={styles.button} onPress={handleAddTransac
        <Text style={styles.buttonText}>Add Transaction</Text>
      </TouchableOpacity>
    </View>
  );
};
```


Rakamin
Academy
Thank you
Raih Mimpi #TanpaBatas