

React Native

React Native HTTP Request



React Native HTTP Request

☐ Fetch

☐ AXIOS

☐ CRUD

Objektif sesi

- Memahami dan mengimplementasikan Fetch
- Memahami dan mengimplementasikan Axios
- Memahami dan mengimplementasikan CRUD

Fetch

Fetch adalah fungsi bawaan dalam JavaScript yang digunakan untuk membuat permintaan HTTP. Dalam konteks React Native, fetch dapat digunakan untuk berkomunikasi dengan server dan mengambil atau mengirim data. fetch mengembalikan Promise yang akan terpecah ketika respons dari server sudah tersedia.

Contoh penggunaan Fetch

Buat komponen UserList.js yang akan menampilkan daftar pengguna dari server

```
jsx
Copy code

// UserList.js
import React, { useState, useEffect } from 'react';
import { View, Text, FlatList, StyleSheet } from 'react-native';

const UserList = () => {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch('https://jsonplaceholder.typicode.com/users');
        const userData = await response.json();
        setUsers(userData);
      } catch (error) {
        console.error('Error fetching data:', error);
      }
    };

    fetchData();
  }, []);
};
```

```
return (
  <View style={styles.container}>
    <Text style={styles.title}>List of Users</Text>
    <FlatList
      data={users}
      keyExtractor={item => String(item.id)}
      renderItem={({ item }) => (
        <View style={styles.userItem}>
          <Text style={styles.userName}>{item.name}</Text>
          <Text>Email: {item.email}</Text>
          <Text>Phone: {item.phone}</Text>
        </View>
      )}
    />
  </View>
);
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 16,
  },
  userItem: {
    marginBottom: 16,
    borderWidth: 1,
    borderColor: '#ddd',
    padding: 10,
    borderRadius: 8,
  },
  userName: {
    fontSize: 16,
    fontWeight: 'bold',
    marginBottom: 8,
  },
});

export default UserList;
```

Contoh penggunaan Fetch

Integrasikan komponen UserList ke dalam App.js

```
jsx
Copy code

// App.js
import React from 'react';
import { SafeAreaView, StatusBar } from 'react-native';
import UserList from './UserList';

const App = () => {
  return (
    <>
      <StatusBar barStyle="dark-content" />
      <SafeAreaView style={{ flex: 1 }}>
        <UserList />
      </SafeAreaView>
    </>
  );
};

export default App;
```

Jalankan aplikasi dengan perintah

```
bash
Copy code


npx react-native run-android
# atau
npx react-native run-ios
```

Axios

Axios adalah sebuah library JavaScript yang digunakan untuk membuat permintaan HTTP di berbagai lingkungan, termasuk React Native. Axios memberikan antarmuka yang sederhana dan bersih untuk membuat permintaan dan menangani respons. Ini juga mendukung fitur-fitur seperti pembatalan permintaan, penanganan respons JSON otomatis, dan penanganan kesalahan dengan lebih baik.

Instalasi Axios

bash

 Copy code

```
npm install axios  
# atau  
yarn add axios
```


Mengambil Data dari Server (GET Request)

```
javascript Copy code

import React, { useState, useEffect } from 'react';
import { View, Text, FlatList, StyleSheet } from 'react-native';
import axios from 'axios';

const UserList = () => {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await axios.get('https://jsonplaceholder.typicode.com/users');
        setUsers(response.data);
      } catch (error) {
        console.error('Error fetching data:', error);
      }
    };

    fetchData();
  }, []);

  return (
    <View style={styles.container}>
      <Text style={styles.title}>List of Users</Text>
      <FlatList
        data={users}
        keyExtractor={item => String(item.id)}
        renderItem={({ item }) => (
          <View style={styles.userItem}>
            <Text style={styles.userName}>{item.name}</Text>
            <Text>Email: {item.email}</Text>
            <Text>Phone: {item.phone}</Text>
          </View>
        )}
      />
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 16,
  },
  userItem: {
    marginBottom: 16,
    borderWidth: 1,
    borderColor: '#ddd',
    padding: 10,
    borderRadius: 8,
  },
  userName: {
    fontSize: 16,
    fontWeight: 'bold',
    marginBottom: 8,
  },
});

export default UserList;
```

Dalam contoh di atas, kita menggunakan `axios.get` untuk membuat permintaan GET ke URL tertentu. Axios secara otomatis akan mengurai respons JSON, dan kita dapat mengakses data dengan `response.data`.

Mengirim Data ke Server (POST Request)

```
javascript
Copy code

import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, StyleSheet } from 'react-native';
import axios from 'axios';

const AddUser = () => {
  const [userName, setUserName] = useState('');
  const [userEmail, setUserEmail] = useState('');

  const handleAddUser = async () => {
    try {
      const response = await axios.post('https://jsonplaceholder.typicode.com/users', {
        name: userName,
        email: userEmail,
      });

      console.log('User added:', response.data);
    } catch (error) {
      console.error('Error adding user:', error);
    }
  };
};
```

```
return (
  <View style={styles.container}>
    <Text style={styles.title}>Add User</Text>
    <TextInput
      style={styles.input}
      placeholder="Enter Name"
      value={userName}
      onChangeText={text => setUserName(text)}
    />
    <TextInput
      style={styles.input}
      placeholder="Enter Email"
      value={userEmail}
      onChangeText={text => setUserEmail(text)}
    />
    <TouchableOpacity style={styles.addButton} onPress={handleAddUser}>
      <Text style={styles.buttonText}>Add User</Text>
    </TouchableOpacity>
  </View>
);
```

Dalam contoh ini, kita menggunakan `axios.post` untuk membuat permintaan POST ke URL tertentu dengan menyertakan data pengguna dalam bentuk objek. Respons dari server kemudian dapat diakses dengan `response.data`.


CRUD

CRUD adalah singkatan dari Create, Read, Update, dan Delete, yang merupakan empat operasi dasar dalam pengelolaan data. Dalam konteks pengembangan aplikasi, CRUD merujuk pada kemampuan untuk membuat, membaca, memperbarui, dan menghapus data dari atau ke server.

Create (Buat Data)

Operasi "Create" melibatkan penambahan data baru ke dalam sistem. Dalam pengembangan aplikasi React Native, ini seringkali melibatkan pengiriman permintaan HTTP POST ke server untuk membuat entitas baru.

javascript

 Copy code

```
// Misalnya, membuat pengguna baru
const createUser = async () => {
  try {
    const response = await fetch('https://api.example.com/users', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ username: 'john_doe', email: 'john@example.com' })
    });

    const newUser = await response.json();
    console.log('New User Created:', newUser);
  } catch (error) {
    console.error('Error creating user:', error);
  }
};
```

Read (Baca Data)

Operasi "Read" melibatkan pengambilan atau membaca data dari server. Dalam pengembangan aplikasi React Native, ini melibatkan pengiriman permintaan HTTP GET ke server untuk mengambil data yang diperlukan.

javascript

Copy code

```
// Misalnya, membaca daftar pengguna
const fetchUserList = async () => {
  try {
    const response = await fetch('https://api.example.com/users');
    const userList = await response.json();
    console.log('User List:', userList);
  } catch (error) {
    console.error('Error fetching user list:', error);
  }
};
```


Update (Perbarui Data)

Operasi "Update" melibatkan pembaruan data yang sudah ada. Dalam pengembangan aplikasi React Native, ini melibatkan pengiriman permintaan HTTP PUT atau PATCH ke server untuk memperbarui entitas tertentu.

javascript

Copy code

```
// Misalnya, memperbarui informasi pengguna
const updateUserInfo = async (userId, newData) => {
  try {
    const response = await fetch(`https://api.example.com/users/${userId}`, {
      method: 'PUT', // atau 'PATCH' tergantung pada kebijakan server
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(newData),
    });

    const updatedUser = await response.json();
    console.log('User Updated:', updatedUser);
  } catch (error) {
    console.error('Error updating user:', error);
  }
};
```

Delete (Hapus Data)

Operasi "Delete" melibatkan penghapusan data dari sistem. Dalam pengembangan aplikasi React Native, ini melibatkan pengiriman permintaan HTTP DELETE ke server untuk menghapus entitas tertentu.

javascript

Copy code

```
// Misalnya, menghapus pengguna
const deleteUser = async (userId) => {
  try {
    const response = await fetch(`https://api.example.com/users/${userId}`, {
      method: 'DELETE',
    });

    if (response.ok) {
      console.log('User Deleted Successfully');
    } else {
      console.log('Failed to delete user');
    }
  } catch (error) {
    console.error('Error deleting user:', error);
  }
};
```

Challenge!

Buatlah aplikasi React Native sederhana yang mengimplementasikan operasi CRUD (Create, Read, Update, Delete) untuk pengelolaan daftar pengguna. Aplikasi ini akan berkomunikasi dengan API server (bisa menggunakan JSONPlaceholder sebagai contoh) untuk melakukan operasi CRUD pada data pengguna.



Thank you