

Networking

REST API
JSON
Postman



Networking

☐ REST API

☐ JSON

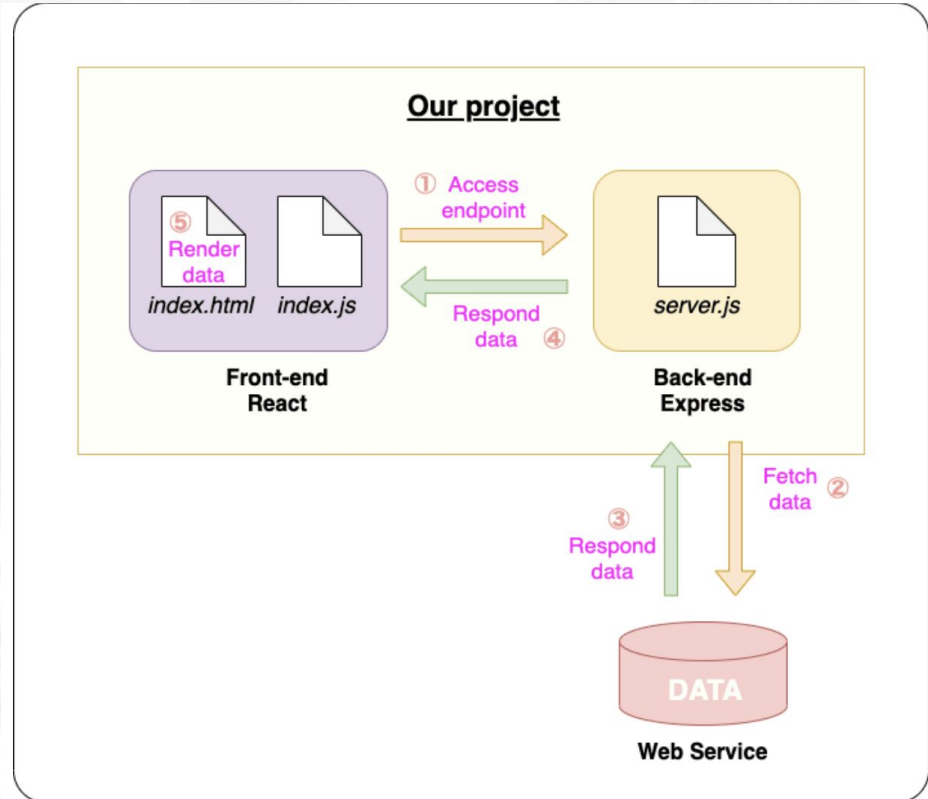
☐ Postman

Objektif sesi

- Memahami dan mengimplementasikan REST API
- Memahami dan mengimplementasikan JSON
- Memahami dan mengimplementasikan Postman

REST API

REST API memanfaatkan protokol HTTP untuk pertukaran data antara aplikasi klien dan server. Dalam konteks pengembangan aplikasi React Native, REST API sering digunakan untuk berkomunikasi antara aplikasi mobile yang dibangun dengan React Native dan server.




Menggunakan Fetch API atau library HTTP

React Native menyediakan Fetch API yang dapat digunakan untuk membuat permintaan HTTP ke server. Ini adalah API bawaan yang bisa langsung digunakan.

Selain itu, beberapa library HTTP populer seperti Axios atau Fetch API polyfill dapat digunakan untuk mempermudah pengelolaan permintaan HTTP.

javascript

 Copy code

```
// Menggunakan Fetch API
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));

// Menggunakan Axios (jika diinstal)
import axios from 'axios';

axios.get('https://api.example.com/data')
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));
```

Mengirim Permintaan (Request)

Untuk mengambil data dari server, Anda dapat menggunakan metode HTTP seperti GET.

Untuk mengirim data ke server, metode seperti POST, PUT, atau PATCH dapat digunakan.

javascript

Copy code


```
// Mengirim data dengan metode POST
fetch('https://api.example.com/create', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({ key: 'value' }),
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

Menanggapi Respons (Response)

Mengelola respons dari server, yang dapat berupa data JSON atau jenis respons lainnya.

Menanggapi respons dengan memproses data atau menangani kesalahan.

javascript


 Copy code

```
// Menggunakan Fetch API
fetch('https://api.example.com/data')
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    return response.json();
  })
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Menyimpan Data

Setelah menerima data dari server, Anda dapat menyimpannya di state React Native atau menggunakan data tersebut sesuai kebutuhan aplikasi.

javascript

 Copy code

```
// Menyimpan data dalam state React Native
const [data, setData] = useState(null);

useEffect(() => {
  fetch('https://api.example.com/data')
    .then(response => response.json())
    .then(data => setData(data))
    .catch(error => console.error('Error:', error));
}, []);
```


JSON

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan dan mudah dibaca oleh manusia. Dalam pengembangan aplikasi React Native, JSON sering digunakan untuk mengirim dan menerima data antara aplikasi mobile dan server melalui jaringan. JSON menggunakan format teks dan terdiri dari pasangan nama/kunci (key) dan nilai (value), membentuk struktur data yang mirip dengan objek JavaScript.

Dalam konteks React Native dan jaringan (networking), JSON sering digunakan dalam respons dari server atau dalam permintaan (request) ke server.

React App

```
{
  "records": [
    {
      "recordID": {
        "type": "RECORD_NUMBER",
        "value": "Unluck"
      },
      "Updated_datetime": {
        "type": "UPDATED_TIME",
        "value": "2021-05-17T11:46:00Z"
      },
      "Created_by": {
        "type": "CREATOR",
        "value": {
          "code": "Tozuka",
          "revision": "1"
        }
      },
      "CHECK_BOX": {
        "type": "CHECK_BOX",
        "value": {
          "Action": "Updated_by"
        }
      },
      "ID": {
        "type": "ID",
        "value": "7"
      },
      "recordID": {
        "type": "RECORD_NUMBER",
        "value": "Golden Kamuy"
      }
    }
  ]
}
```

Dirty data



React App

- **Gunnm** written by Yukito Kishiro
- **Usogui** written by Toshio Sako
- **Batou** written by Toshio Sako
- **Kimetsu no Yaiba** written by Koyoharu Gotouge
- **Hachiwan Diver** written by Yokusaru Shibata
- **Golden Kamuy** written by Satoru Noda
- **Undead Unluck** written by Yoshifumi Tozuka

Clean data

Mengambil Data dari Server (Fetch JSON Data)

Dalam contoh di samping, aplikasi React Native mengambil data dari URL `https://api.example.com/data` menggunakan `fetch`. Kemudian, fungsi `.json()` digunakan untuk menguraikan respons menjadi objek JavaScript. Data tersebut kemudian disimpan dalam state menggunakan `setData`.

```
javascript Copy code

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

const App = () => {
  const [data, setData] = useState(null);

  useEffect(() => {
    // Mengambil data dari server yang mengembalikan JSON
    fetch('https://api.example.com/data')
      .then(response => response.json())
      .then(jsonData => setData(jsonData))
      .catch(error => console.error('Error:', error));
  }, []);

  return (
    <View>
      {data ? (
        <Text>{JSON.stringify(data)}</Text>
      ) : (
        <Text>Loading data...</Text>
      )}
    </View>
  );
};

export default App;
```

Mengirim Data ke Server (POST JSON Data)

Pada contoh di samping, terdapat fungsi `sendDataToServer` yang menggunakan metode HTTP POST untuk mengirim data ke server. Data yang dikirim dalam contoh ini adalah objek JavaScript yang kemudian diubah menjadi format JSON menggunakan `JSON.stringify`.

```
javascript Copy code

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

const App = () => {
  const [responseData, setResponseData] = useState(null);

  const sendDataToServer = () => {
    const dataToSend = {
      username: 'john_doe',
      password: 'secret123',
    };

    // Mengirim data ke server dalam format JSON
    fetch('https://api.example.com/login', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(dataToSend),
    })
      .then(response => response.json())
      .then(jsonResponse => setResponseData(jsonResponse))
      .catch(error => console.error('Error:', error));
  };
};
```

```
return (
  <View>
    <TouchableOpacity onPress={sendDataToServer}>
      <Text>Send Data to Server</Text>
    </TouchableOpacity>

    {responseData && (
      <Text>{JSON.stringify(responseData)}</Text>
    )}
  </View>
);

export default App;
```

POSTMAN

Postman adalah sebuah developer tools, yang memungkinkan developer untuk menguji, mengelola, dan memahami fungsi API.

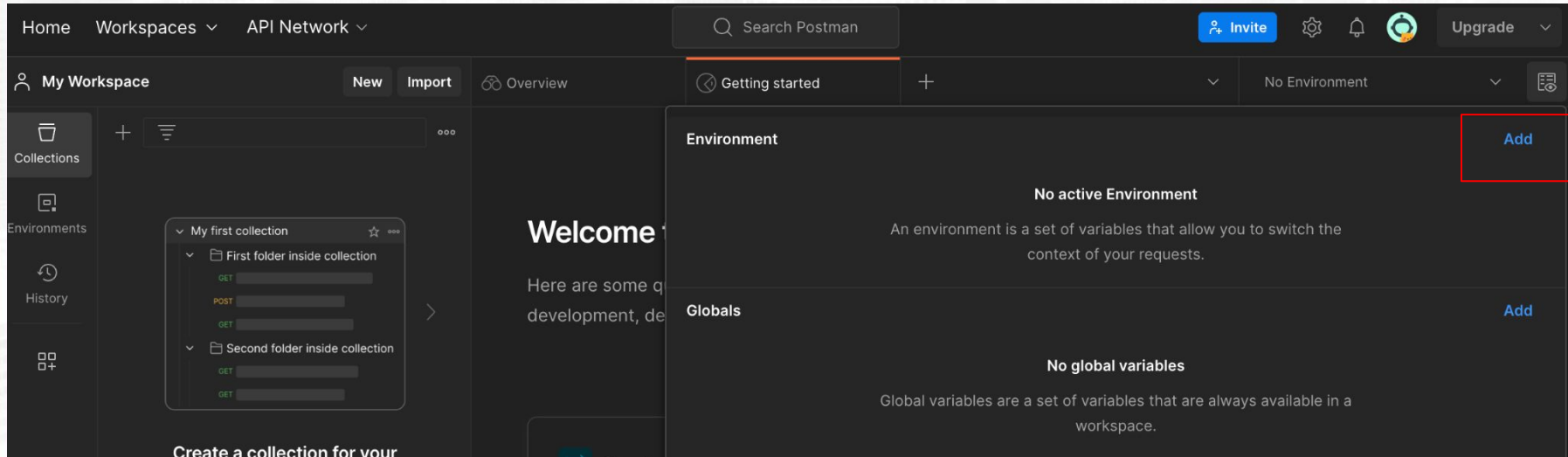
Dengan Postman, developer dapat membuat dan mengirim permintaan HTTP ke server, mengelola respons, dan mengotomatisasi pengujian API.

Postman menyediakan antarmuka pengguna yang ramah dan fitur-fitur yang membantu pengembang berinteraksi dengan API dengan lebih mudah.



Instalasi POSTMAN

Pastikan postman sudah terhubung dengan account google yang kita miliki karena setiap perubahan di collection, postman akan melakukan sync secara berkala. setting environment ini akan memudahkan kita dalam membedakan environment API dev dan API live. berikut caranya :



Fungsi POSTMAN

1. Membuat Permintaan (Request):

- Buka Postman dan buat permintaan baru.
- Pilih metode HTTP yang sesuai (GET, POST, PUT, dll.).
- Tentukan URL endpoint yang akan diuji.

1. Menambahkan Header atau Parameter:

- Jika diperlukan, Anda dapat menambahkan header atau parameter ke permintaan. Misalnya, jika mengirim data JSON, tambahkan header "Content-Type: application/json".

1. Mengirim Permintaan:

- Tekan tombol "Send" untuk mengirim permintaan ke server.
- Lihat respons yang diterima dari server, termasuk status kode, header, dan body.

1. Mengelola Respons:

- Periksa body respons untuk melihat data yang diterima dari server.
- Postman memudahkan dalam memahami dan memformat data JSON.

1. Simpan dan Mengelola Permintaan:

- Simpan permintaan yang sering digunakan sebagai koleksi.
- Gunakan fitur pengelolaan koleksi untuk menyimpan dan mengelola sekumpulan permintaan terkait.





Thank you