Danis Keysara Saputra

**Rest API dan Implementasi pada Javascript**

API (Application Programming Interface)adalah sekumpulan aturan atau kode yang menghubungkan 2 program atau lebih untuk berkomunikasi, API menjadi penghubung antar aplikasi / website dalam satu platform maupun lintas platform.

Salah satu API Protocol (Sekumpulan aturan untuk mengatur penggunaan database dalam server) yang sering digunakan yaitu REST API.

REST API (Representational State Transfer Application Programming Interface) adalah gaya arsitektur untuk membangun layanan web yang menggunakan protokol HTTP untuk berkomunikasi antara klien dan server. REST API memungkinkan klien untuk melakukan operasi seperti mengambil (GET), menambahkan (POST), memperbarui (PUT/PATCH), dan menghapus (DELETE) data dengan menggunakan permintaan HTTP yang sesuai. REST API didasarkan pada prinsip-prinsip seperti sumber daya yang dapat diidentifikasi dengan URL, manipulasi sumber daya menggunakan metode HTTP, representasi data yang dikirim melalui format seperti JSON atau XML, dan pemisahan antara klien dan server

Langkah:

1. Klien mengirimkan permintaan (request) ke server melalui JSON, yang berisi data yang diperlukan untuk operasi yang diinginkan. Permintaan ini dapat berupa GET, POST, PUT, atau DELETE.
2. Permintaan dikirim menggunakan metode HTTP yang sesuai, seperti GET, POST, PUT, atau DELETE. Metode HTTP ini mengindikasikan tindakan yang diinginkan terhadap sumber daya yang dituju.
3. Server menerima permintaan dari klien dan memprosesnya sesuai dengan metode HTTP yang dikirimkan. Server menerjemahkan permintaan tersebut dan mengambil atau memodifikasi data yang relevan.
4. Server mengirimkan respons (response) ke klien. Respons ini berisi status HTTP, yang menunjukkan apakah permintaan berhasil (misalnya, status 200 OK) atau gagal (misalnya, status 404 Not Found atau 500 Internal Server Error). Respons juga berisi data yang diminta atau informasi tambahan yang relevan.

Konsep/istilah:

1. Sumber daya (Resource): Merupakan entitas yang direpresentasikan dalam sistem, seperti pengguna, produk, atau pesan.
2. Operasi HTTP: REST API menggunakan metode HTTP seperti GET (mengambil data), POST (mengirim data baru), PUT (mengganti data yang ada), dan DELETE (menghapus data).
3. Representasi: Data yang dikirim melalui REST API dapat diwakili dalam berbagai format, seperti JSON (JavaScript Object Notation) atau XML (eXtensible Markup Language).
4. Endpoint: Setiap sumber daya pada REST API diakses melalui URL yang disebut endpoint. Endpoint ini berfungsi sebagai titik akhir atau alamat dari suatu layanan web yang dapat diakses melalui HTTP.
5. Stateless: REST API dianggap stateless, artinya setiap permintaan ke API harus mencakup semua informasi yang diperlukan untuk memproses permintaan tersebut. Server tidak menyimpan informasi status dari klien antara permintaan.

Implementasi pada javascript:

JavaScript dapat digunakan untuk mengakses dan berinteraksi dengan REST API

Contoh:

* Membuat get:

fetch(‘URL endpoint pada rest api (HTTP)/users’)

.then(response => response.json())

.then(data => {

console.log(data);

})

.catch(error => {

console.error('Error:', error);

});

Fetch untuk membuat permintaan GET ke URL endpoint /users pada REST API. Setelah mendapatkan respons dari API, metode json() mengonversi respons ke dalam format JSON. Data JSON tersebut dapat digunakan untuk tindakan selanjutnya, seperti menampilkan data ke pengguna atau memprosesnya.

* Membuat permintaan POST

fetch(URL endpoint pada rest api (HTTP)’), {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

name: 'John Doe',

email: 'john.doe@example.com'

})

})

.then(response => response.json())

.then(data => {

console.log(data);

})

.catch(error => {

console.error('Error:', error);

});

* Membuat permintaan PUT

fetch(URL endpoint pada rest api (HTTP)’), {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

name: ‘Danis’,

email: danis@upi.edu.

})

})

.then(response => {

if (response.ok) {

console.log('Data berhasil diperbarui');

} else {

console.log('Error saat memperbarui data');

}

})

.catch(error => {

console.error('Error:', error);

});

* Membuat permintaan DELETE

fetch(URL endpoint pada rest api (HTTP)’), {

method: 'DELETE'

})

.then(response => {

if (response.ok) {

console.log('Data berhasil dihapus');

} else {

console.log('Error saat menghapus data');

}

})

.catch(error => {

console.error('Error:', error);

});