

**LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT LUNAK**

**MODUL XI
API Design dan Construction**



**Disusun Oleh:
Dewi Atika Muthi / 2211104042
SE-06-02**

**Asisten Praktikum:
Muhamad Taufiq Hidayat**

**Dosen Pengampu:
Riyan Dwi Yulian Prakoso, S.Kom., M.Kom.**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

BAB I

PENDAHULUAN

A. DASAR TEORI

Web API (Application Programming Interface) merupakan antarmuka yang memungkinkan aplikasi untuk saling berkomunikasi melalui protokol HTTP. Dalam pengembangan backend, Node.js bersama dengan framework Express.js menjadi salah satu kombinasi populer untuk membangun Web API karena kemudahan penggunaan, performa tinggi, dan berbasis JavaScript yang non-blocking.

Menurut Modul Praktikum, Web API yang dibuat menggunakan Node.js dan Express bertujuan memberikan layanan (service) seperti mengakses, menambahkan, dan menghapus data melalui endpoint RESTful. Express.js mempermudah developer dalam menangani routing dan middleware.

`"Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications."` (Express.js Documentation)

Selain itu, berdasarkan jurnal oleh Agustinus dan Fauzan (2020), penggunaan REST API dengan metode HTTP (GET, POST, DELETE) menjadi standar dalam pengembangan sistem terdistribusi yang membutuhkan komunikasi antar aplikasi.

B. MAKSUD DAN TUJUAN

Maksud dari praktikum ini adalah mempelajari cara membangun sebuah Web API sederhana menggunakan Node.js dan Express.js.

Adapun tujuan dari praktikum ini adalah:

1. Memahami proses inisialisasi proyek Node.js.
2. Menerapkan konsep REST API menggunakan Express.
3. Mengelola data secara dinamis menggunakan array sebagai penyimpanan sementara.
4. Menangani operasi CRUD (Create, Read, Delete) dengan endpoint yang sesuai.

BAB II IMPLEMENTASI

A. PRAKTIKUM (GUIDED)

Soal Studi Case

Membuat backend sederhana dibutuhkan untuk mengelola data mahasiswa berupa nama dan NIM. Sistem ini tidak menggunakan database, melainkan menyimpan data sementara dalam bentuk array di memori server.

1. Inisialisasi Proyek

```
mkdir rest_api  
cd rest_api  
npm init -y
```

Hasil:

```
D:\PRAK_KPL>mkdir rest_api  
  
D:\PRAK_KPL>cd rest_api  
  
D:\PRAK_KPL\rest_api>npm init -y  
Wrote to D:\PRAK_KPL\rest_api\package.json:  
  
{  
  "name": "rest_api",  
  "version": "1.0.0",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "description": ""  
}
```

Perintah ini membuat folder proyek baru dan menginisialisasi package.json secara otomatis.

2. Install Express

```
npm install express
```

Hasil:

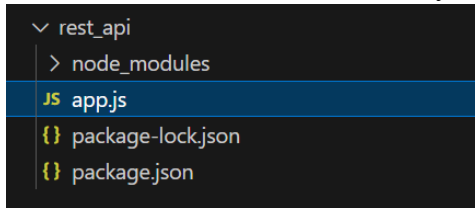
```
D:\PRAK_KPL\rest_api>npm install express  
  
added 66 packages, and audited 67 packages in 5s  
  
14 packages are looking for funding  
run `npm fund` for details  
  
found 0 vulnerabilities  
  
D:\PRAK_KPL\rest_api>
```

```
rest_api  
├── node_modules  
├── package-lock.json  
└── package.json
```

Express adalah framework web minimalis untuk Node.js yang digunakan untuk membuat routing dan handler HTTP dengan mudah.

3. Struktur Folder

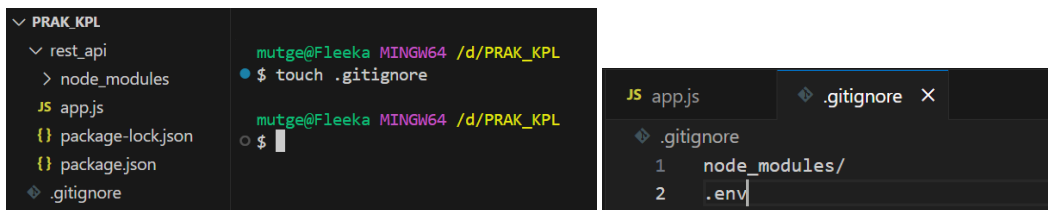
Berikut ini adalah struktur folder yang akan digunakan (menambahkan `app.js`)



4. Membuat File .gitignore

```
node_modules/  
.env
```

Hasil:



Tujuannya adalah agar folder `node_modules` dan file konfigurasi `.env` tidak diikutsertakan dalam version control (Git).

5. Source Code `app.js`

```
const express = require('express');  
const app = express();  
const port = 3000;  
  
app.use(express.json());  
  
let mahasiswa = [  
  { nama: "Muhamad Taufiq Hidayat", nim: "21102206" },  
  { nama: "Febrilia Ananda", nim: "220220106" },  
  { nama: "LeBron James", nim: "1302000003" }  
];  
  
// GET semua mahasiswa  
app.get('/api/mahasiswa', (req, res) => {  
  res.json(mahasiswa);  
});  
  
// GET mahasiswa berdasarkan index  
app.get('/api/mahasiswa/:index', (req, res) => {  
  const index = parseInt(req.params.index);  
  if (index >= 0 && index < mahasiswa.length) {  
    res.json(mahasiswa[index]);  
  } else {  
    res.status(404).json({ message: 'Data tidak ditemukan' });  
  }  
});  
  
// POST mahasiswa baru  
app.post('/api/mahasiswa', (req, res) => {  
  const { nama, nim } = req.body;  
  mahasiswa.push({ nama, nim });  
  res.status(201).json({ message: 'Data berhasil ditambahkan' });  
});
```

```

});

// DELETE mahasiswa berdasarkan index
app.delete('/api/mahasiswa/:index', (req, res) => {
  const index = parseInt(req.params.index);
  if (index >= 0 && index < mahasiswa.length) {
    mahasiswa.splice(index, 1);
    res.json({ message: 'Data berhasil dihapus' });
  } else {
    res.status(404).json({ message: 'Data tidak ditemukan' });
  }
});

app.listen(port, () => {
  console.log(`Server berjalan di http://localhost:${port}`);
});

```

Deskripsi Program:

Program di atas membuat server dengan endpoint sebagai berikut:

Endpoint	Metode	Deskripsi
/api/mahasiswa	GET	Mengambil semua data mahasiswa
/api/mahasiswa/:index	GET	Mengambil data mahasiswa berdasarkan index
/api/mahasiswa	POST	Menambahkan data mahasiswa baru
/api/mahasiswa/:index	DELETE	Menghapus data mahasiswa berdasarkan index

6. Menjalankan Proyek

```
node app.js
```

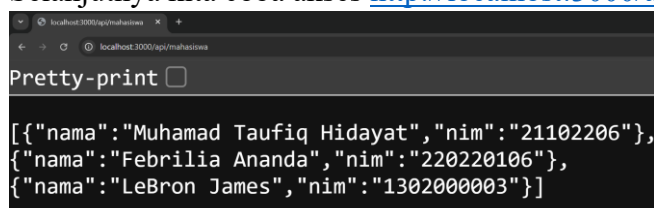
Server berhasil dijalankan,

```

D:\PRAK_KPL\rest_api>node app.js
Server berjalan di http://localhost:3000

```

Selanjutnya kita coba akses <http://localhost:3000/api/mahasiswa>



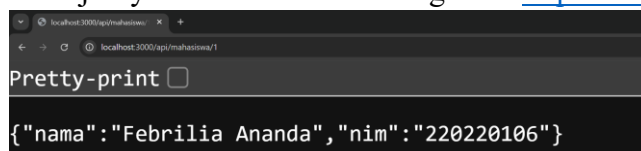
```

[{"nama":"Muhamad Taufiq Hidayat","nim":"21102206"},
{"nama":"Febrilia Ananda","nim":"220220106"},
{"nama":"LeBron James","nim":"1302000003"}]

```

Maka server mengambil data mahasiswa yang tersimpan dan menampilkannya:

Selanjutnya kita coba untuk mengakses <http://localhost:3000/api/mahasiswa/1>



```

{"nama":"Febrilia Ananda","nim":"220220106"}

```

Maka server akan mengambil data indeks ke 1 (list data ke-2) lalu menampilkannya.

B. TUGAS (UNGUIDED)

Soal Studi Case

Mahasiswa diminta membangun Web API sederhana tanpa database untuk mengelola data mahasiswa yang terdiri dari nama dan NIM. Data disimpan dalam array statik dan diuji melalui tool Postman.

Sourcecode app.js

```
const express = require('express');
const app = express();
const port = 3000;

app.use(express.json());

let mahasiswa = [
  { nama: "Dewi Atika Muthi", nim: "221114042" },
  { nama: "Anggota 2", nim: "1302010002" },
  { nama: "Anggota 3", nim: "1302010003" }
];

// GET semua mahasiswa
app.get('/api/mahasiswa', (req, res) => {
  res.json(mahasiswa);
});

// GET mahasiswa berdasarkan index
app.get('/api/mahasiswa/:index', (req, res) => {
  const index = parseInt(req.params.index);
  if (index >= 0 && index < mahasiswa.length) {
    res.json(mahasiswa[index]);
  } else {
    res.status(404).json({ message: 'Data tidak ditemukan' });
  }
});

// POST mahasiswa baru
app.post('/api/mahasiswa', (req, res) => {
  const { nama, nim } = req.body;
  mahasiswa.push({ nama, nim });
  res.status(201).json({ message: 'Data berhasil ditambahkan' });
});

// DELETE mahasiswa berdasarkan index
app.delete('/api/mahasiswa/:index', (req, res) => {
  const index = parseInt(req.params.index);
  if (index >= 0 && index < mahasiswa.length) {
    mahasiswa.splice(index, 1);
    res.json({ message: 'Data berhasil dihapus' });
  } else {
    res.status(404).json({ message: 'Data tidak ditemukan' });
  }
});

app.listen(port, () => {
  console.log(`Server berjalan di http://localhost:${port}`);
});
```

Deskripsi Program:

Web API yang saya buat mirip seperti pada bagian guided, yang menggunakan array mahasiswa sebagai penyimpanan statik. Endpoint REST yang disediakan:

Method	Endpoint	Fungsi
GET	/api/mahasiswa	Menampilkan seluruh data mahasiswa
GET	/api/mahasiswa/:index	Menampilkan data mahasiswa berdasarkan index
POST	/api/mahasiswa	Menambahkan data baru
DELETE	/api/mahasiswa/:index	Menghapus data berdasarkan index

Screenshoot Uji Coba Postman:

A. GET data

The first screenshot shows a GET request to `http://localhost:3000/api/mahasiswa` with a status of `200 OK`. The response body is a JSON array of three student objects:

```
[
  {
    "nama": "Dewi Atika Muthi",
    "nim": "221114042"
  },
  {
    "nama": "Cabine Thither",
    "nim": "2211104822"
  },
  {
    "nama": "Hilda",
    "nim": "2211104823"
  }
]
```

The second screenshot shows a POST request to `http://localhost:3000/api/mahasiswa` with a status of `200 OK`. The response body is a table with two columns: `nama` and `nim`.

	nama	nim
0	Dewi Atika Muthi	221114042
1	Cabine Thither	2211104822
2	Hilda	2211104823

B. POST data baru (John Doe – 1302199999)

The first screenshot shows a POST request to `http://localhost:3000/api/mahasiswa` with a JSON body: `{ "nama": "John Doe", "nim": "1302199999" }`. The second screenshot shows the response: `{ "message": "Data berhasil ditambahkan" }` with a status of 201 Created.

```
1 {
2   "nama": "John Doe",
3   "nim": "1302199999"
4 }
```

```
1 {
2   "message": "Data berhasil ditambahkan"
3 }
```

C. GET data setelah POST

The first screenshot shows a GET request to `http://localhost:3000/api/mahasiswa` returning a 200 OK status. The response is a JSON array of four student objects. The second screenshot shows the same response visualized as a table.

```
1 [
2   {
3     "nama": "Dewi Atika Muthi",
4     "nim": "221114042"
5   },
6   {
7     "nama": "Cabine Thithet",
8     "nim": "2211104822"
9   },
10  {
11    "nama": "Hilda",
12    "nim": "2211104823"
13  },
14  {
15    "nama": "John Doe",
16    "nim": "1302199999"
17  }
18 ]
```

	nama	nim
0	Dewi Atika Muthi	221114042
1	Cabine Thithet	2211104822
2	Hilda	2211104823
3	John Doe	1302199999

D. GET data INDEX 0 (data saya, Dewi Atika Muthi)

The first screenshot shows a REST client interface with a GET request to `http://localhost:3000/api/mahasiswa/0`. The response is a 200 OK status with a 34 ms response time and 280 B of data. The body is displayed as raw JSON:

```
1 {
2   "nama": "Dewi Atika Muthi",
3   "nim": "221114042"
4 }
```

The second screenshot shows the same request and response, but the body is displayed in a table format:

nama	Dewi Atika Muthi
nim	221114042

E. DELETE data INDEX ke-0 (data saya, Dewi Atika Muthi)

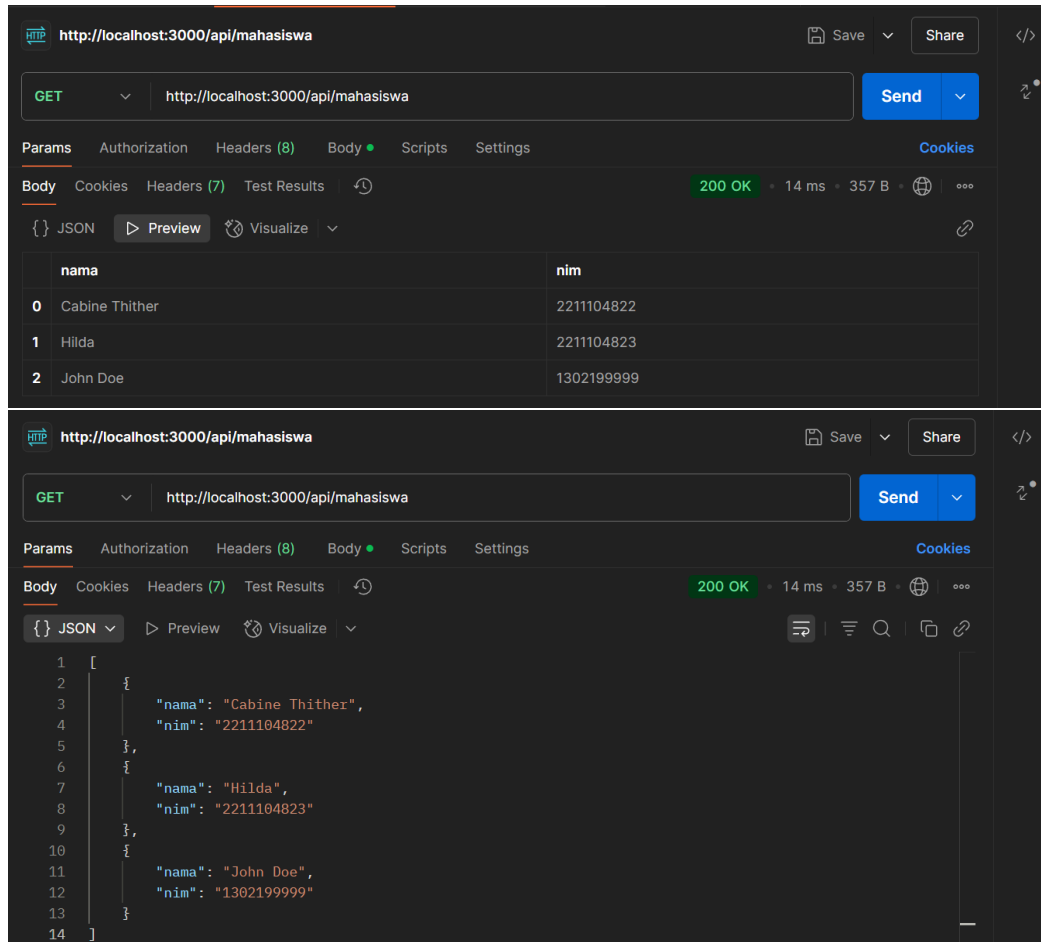
The first screenshot shows a REST client interface with a DELETE request to `http://localhost:3000/api/mahasiswa/0`. The response is a 200 OK status with a 158 ms response time and 270 B of data. The body is displayed as raw JSON:

```
1 {
2   "message": "Data berhasil dihapus"
3 }
```

The second screenshot shows a REST client interface with a GET request to `http://localhost:3000/api/mahasiswa`. The response is a 200 OK status with a 14 ms response time and 357 B of data. The body is displayed as raw JSON:

```
1 [
2   {
3     "nama": "Cabine Thither",
4     "nim": "2211104822"
5   },
6   {
7     "nama": "Hilda",
8     "nim": "2211104823"
9   },
10  {
11    "nama": "John Doe",
12    "nim": "1302199999"
13  }
14 ]
```

F. GET data setelah DELETE



Berikut ini poin dokumentai uji cobanya:

No	Skema Uji	Hasil
A	GET semua mahasiswa	3 data awal muncul
B	POST John Doe	Respons sukses dan data bertambah
C	GET semua mahasiswa	John Doe muncul di akhir
D	GET index 0	Data pertama (nama saya) muncul
E	DELETE index 0	Data pertama berhasil dihapus
F	GET semua mahasiswa	Data saya sudah tidak ada

Deskripsi tambahan:

Web API ini menggunakan Node.js dan Express.js sesuai spesifikasi tugas. Data mahasiswa disimpan dalam array statik tanpa menggunakan database. API menyediakan **empat endpoint utama** untuk melakukan operasi dasar: menampilkan seluruh data (GET), menampilkan data berdasarkan indeks (GET), menambahkan data baru (POST), dan menghapus data berdasarkan indeks (DELETE). Seluruh endpoint diuji menggunakan Postman, mulai dari pengecekan data awal hingga simulasi penambahan dan penghapusan data, dan seluruh fungsionalitas berhasil dijalankan dengan baik.

BAB III

KESIMPULAN DAN SARAN

A. KESIMPULAN

Pada praktikum ini, kita telah mempelajari bagaimana membangun Web API menggunakan Node.js dan Express.js. Dimulai dari inisialisasi proyek, instalasi dependency, hingga implementasi endpoint RESTful. Konsep CRUD telah berhasil diterapkan untuk mengelola data mahasiswa secara sederhana menggunakan array.

Materi ini memberikan pemahaman awal yang baik tentang bagaimana backend dikembangkan dan bagaimana API menjadi jembatan komunikasi antara client dan server. Praktikum ini juga menjadi dasar penting untuk melanjutkan ke pengembangan backend yang lebih kompleks seperti penggunaan database, autentikasi, dan pengelolaan data skala besar.

B. REFERENSI

Modul Praktikum: Membuat Web API di Node.js, 2025

Express.js Documentation. Express – Node.js web application framework.

<https://expressjs.com/>

Mozilla Developer Network (MDN). Working with JSON and HTTP methods.

<https://developer.mozilla.org/>

Postman Docs. API Testing with Postman. <https://learning.postman.com/>

W3Schools. Node.js Express Tutorial.

https://www.w3schools.com/nodejs/nodejs_express.asp