

**LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT LUNAK**

**MODUL IX
REST API NODE.JS**



**Disusun Oleh :
Dimas Cahyo Margono
S1SE-06-02**

**Asisten Praktikum :
Muhamad Taufiq Hidayat**

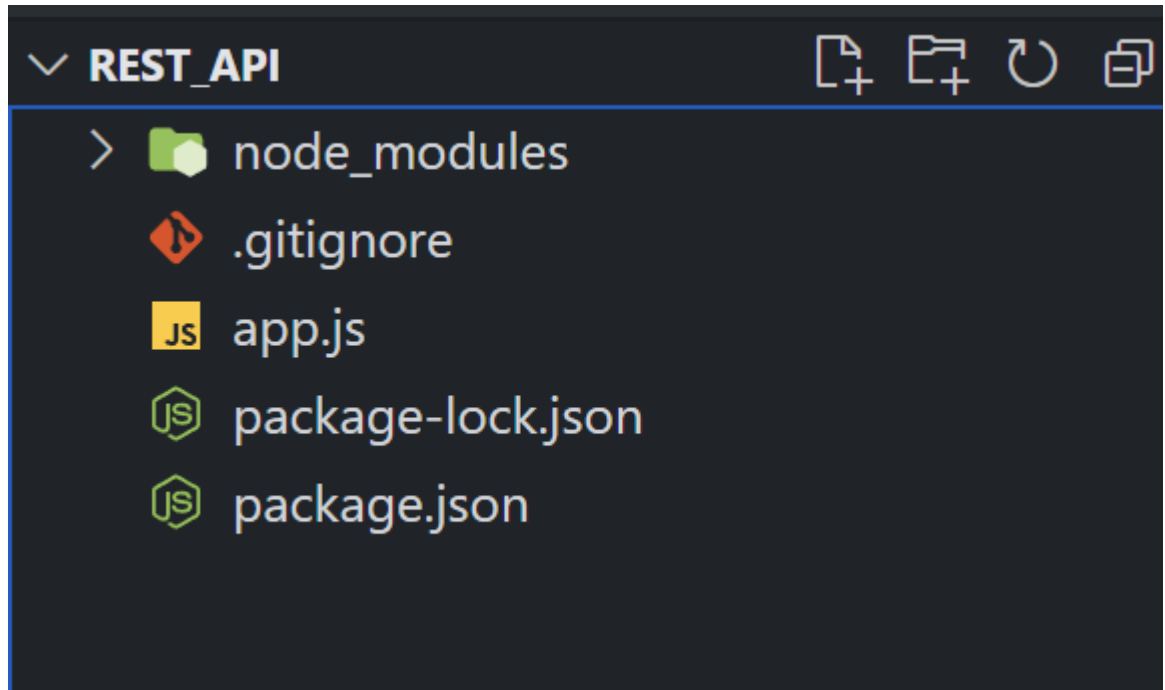
**Dosen Pengampu :
Riyan Dwi Yulian Prakoso, S.Kom., M.Kom.**

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
DIREKTORAT TELKOM KAMPUS PURWOKERTO
2025**

BAB I GUIDED

Hasil Program:

1. Inisialisasi Struktur Folder



2. app.js

```
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.use(express.json());
6
7  // Simpan data mahasiswa di array static
8  let mahasiswa = [
9    { nama: "Muhamad Taufiq Hidayat", nim: "21102206" },
10   { nama: "Febrilia Ananda", nim: "220220106" },
11   { nama: "LeBron James", nim: "1302000003" }
12 ];
13
14 // GET semua mahasiswa
15 app.get('/api/mahasiswa', (req, res) => {
16   res.json(mahasiswa);
17 });
18
19 // GET mahasiswa berdasarkan index
20 app.get('/api/mahasiswa/:index', (req, res) => {
21   const index = parseInt(req.params.index);
22   if (index >= 0 && index < mahasiswa.length) {
23     res.json(mahasiswa[index]);
24   } else {
25     res.status(404).json({ message: 'Data tidak ditemukan' });
26   }
27 });
28
29 // POST mahasiswa baru
30 app.post('/api/mahasiswa', (req, res) => {
31   const { nama, nim } = req.body;
32   mahasiswa.push({ nama, nim });
33   res.status(201).json({ message: 'Data berhasil ditambahkan' });
34 });
35
36 // DELETE mahasiswa berdasarkan index
37 app.delete('/api/mahasiswa/:index', (req, res) => {
38   const index = parseInt(req.params.index);
39   if (index >= 0 && index < mahasiswa.length) {
40     mahasiswa.splice(index, 1);
41     res.json({ message: 'Data berhasil dihapus' });
42   } else {
43     res.status(404).json({ message: 'Data tidak ditemukan' });
44   }
45 });
46
47 app.listen(port, () => {
48   console.log(`Server berjalan di http://localhost:${port}`);
49 });
```

Contoh output:

```

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

```

Penjelasan Kode:

Kode di atas merupakan implementasi API sederhana menggunakan Express.js untuk mengelola data mahasiswa yang disimpan dalam array statis. API ini mendukung operasi dasar seperti membaca semua data mahasiswa (GET /api/mahasiswa), membaca satu mahasiswa berdasarkan index (GET /api/mahasiswa/:index), menambah data mahasiswa baru dengan mengirim JSON lewat body (POST /api/mahasiswa), serta menghapus data mahasiswa berdasarkan index (DELETE /api/mahasiswa/:index). Middleware `express.json()` digunakan agar server dapat membaca data JSON dari body permintaan. Server berjalan pada port 3000, dan saat dijalankan akan menampilkan pesan bahwa server aktif di <http://localhost:3000>.

BAB II

UNGUIDED

Dokumentasi REST API dengan Postman

1. app.js

```
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.use(express.json());
6
7  // Simpan data mahasiswa di array static
8  let mahasiswa = [
9    { nama: "Dimas Cahyo Margono", nim: "2211104060" },
10   { nama: "Cristiano Ronaldo", nim: "1302000001" },
11   { nama: "Lionel Messi", nim: "1302000002" }
12 ];
13
14 // GET semua mahasiswa
15 app.get('/api/mahasiswa', (req, res) => {
16   res.json(mahasiswa);
17 });
18
19 // GET mahasiswa berdasarkan index
20 app.get('/api/mahasiswa/:index', (req, res) => {
21   const index = parseInt(req.params.index);
22   if (index >= 0 && index < mahasiswa.length) {
23     res.json(mahasiswa[index]);
24   } else {
25     res.status(404).json({ message: 'Data tidak ditemukan' });
26   }
27 });
28
29 // POST mahasiswa baru
30 app.post('/api/mahasiswa', (req, res) => {
31   const { nama, nim } = req.body;
32   mahasiswa.push({ nama, nim });
33   res.status(201).json({ message: 'Data berhasil ditambahkan' });
34 });
35
36 // DELETE mahasiswa berdasarkan index
37 app.delete('/api/mahasiswa/:index', (req, res) => {
38   const index = parseInt(req.params.index);
39   if (index >= 0 && index < mahasiswa.length) {
40     mahasiswa.splice(index, 1);
41     res.json({ message: 'Data berhasil dihapus' });
42   } else {
43     res.status(404).json({ message: 'Data tidak ditemukan' });
44   }
45 });
46
47 app.listen(port, () => {
48   console.log(`Server berjalan di http://localhost:${port}`);
49 });
```

2. Dokumentasi Postman

a. GET /api/mahasiswa untuk Menampilkan seluruh data mahasiswa

Modul 9 KPL / Menampilkan seluruh data mahasiswa

GET `{{baseUrl}} /api/mahasiswa` **Send**

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results **200 OK** • 11 ms • 378 B • Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "nama": "Dimas Cahyo Margono",
4     "nim": "2211104060"
5   },
6   {
7     "nama": "Cristiano Ronaldo",
8     "nim": "1302000001"
9   },
10  {
11    "nama": "Lionel Messi",
12    "nim": "1302000002"
13  }
14 ]
```

b. GET /api/mahasiswa/:index untuk Menampilkan data mahasiswa berdasarkan indeks array

Modul 9 KPL / Menampilkan data mahasiswa berdasarkan index

GET `{{baseUrl}} /api/mahasiswa/0` **Send**

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results **200 OK** • 8 ms • 284 B • Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "nama": "Dimas Cahyo Margono",
3   "nim": "2211104060"
4 }
```

- c. Menjalankan POST `/api/mahasiswa` untuk menambahkan John Doe – 1302199999

The screenshot shows the Postman interface for a POST request to `{{baseUrl}}/api/mahasiswa`. The request body is a JSON object: `{ "nama": "John Doe", "nim": "1302199999" }`. The response is a 201 Created status with a 36 ms response time and 279 B of data. The response body is a JSON object: `{ "message": "Data berhasil ditambahkan" }`.

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

Body Cookies Headers (7) Test Results 201 Created • 36 ms • 279 B • Save Response

Pretty Raw Preview Visualize JSON

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

- d. Menampilkan GET `/api/mahasiswa` dan pastikan nama John Doe muncul

The screenshot shows the Postman interface for a GET request to `{{baseUrl}}/api/mahasiswa`. The response is a 200 OK status with a 6 ms response time and 417 B of data. The response body is a JSON array of student objects, including John Doe.

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK • 6 ms • 417 B • Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "nama": "Dimas Cahyo Margono",
4     "nim": "2211104060"
5   },
6   {
7     "nama": "Cristiano Ronaldo",
8     "nim": "1302000001"
9   },
10  {
11    "nama": "Lionel Messi",
12    "nim": "1302000002"
13  },
14  {
15    "nama": "John Doe",
16    "nim": "1302199999"
17  }
18 ]
```

- e. Jalankan GET /api/mahasiswa/0 dan pastikan data pertama (nama Anda) muncul

The screenshot shows the Postman interface for a GET request to the endpoint `{{baseUrl}} /api/mahasiswa/0`. The request is successful, returning a 200 OK status with a response time of 8 ms and a body size of 284 B. The response body is displayed in JSON format, showing a single student record:

```
1 {
2   "nama": "Dimas Cahyo Margono",
3   "nim": "2211104060"
4 }
```

The interface includes tabs for Params, Authorization, Headers (6), Body, Scripts, and Settings. The Body tab is selected, and the response is formatted as JSON.

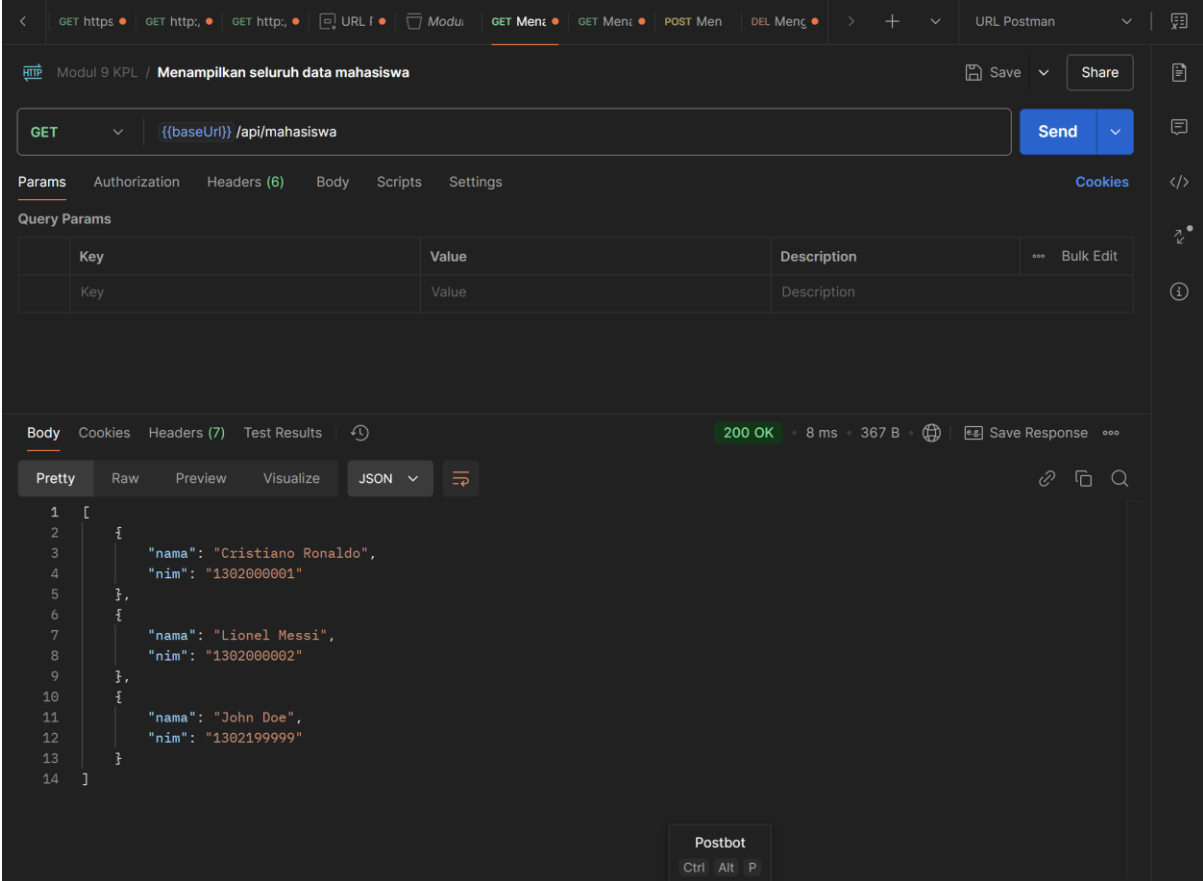
- f. Jalankan DELETE /api/mahasiswa/0 untuk menghapus data pertama

The screenshot shows the Postman interface for a DELETE request to the endpoint `{{baseUrl}} /api/mahasiswa/0`. The request is successful, returning a 200 OK status with a response time of 7 ms and a body size of 270 B. The response body is displayed in JSON format, showing a success message:

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

The interface includes tabs for Params, Authorization, Headers (6), Body, Scripts, and Settings. The Body tab is selected, and the response is formatted as JSON. The bottom of the interface shows a status bar with icons for Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and other utilities.

g. Jalankan GET /api/mahasiswa lagi dan pastikan data Anda sudah hilang



The screenshot shows the Postman interface for a REST client. The top bar indicates the current request is a GET method to the URL `{{baseUri}} /api/mahasiswa`. The status bar at the bottom shows a successful response with a 200 OK status, 8 ms execution time, and 367 B of data. The response body is displayed in JSON format, showing an array of three student objects.

Query Params

Key	Value	Description
Key	Value	Description

Body

200 OK • 8 ms • 367 B

Save Response

Postbot
Ctrl Alt P

```
1 [
2   {
3     "nama": "Cristiano Ronaldo",
4     "nim": "1302000001"
5   },
6   {
7     "nama": "Lionel Messi",
8     "nim": "1302000002"
9   },
10  {
11    "nama": "John Doe",
12    "nim": "1302199999"
13  }
14 ]
```