LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT LUNAK

MODUL IX
Membuat API di NODE.JS

Disusun Oleh :

Arzario Irsyad Al Fatih

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
PENDAHULUAN

1. DASAR TEORI

1. API

API adalah antarmuka yang memungkinkan dua aplikasi atau komponen perangkat lunak untuk saling berkomunikasi. Dalam konteks web, API yang umum digunakan adalah RESTful API, di mana client dapat melakukan operasi seperti GET (mengambil data), POST (menambahkan data), DELETE (menghapus data), dan lainnya melalui protokol HTTP.

2. Express.js

Express.js adalah framework minimalis dan fleksibel untuk Node.js yang digunakan untuk membangun aplikasi web dan API. Express menyederhanakan penulisan server HTTP dengan menyediakan metode routing dan middleware yang mudah digunakan.

3. Routing dan Metode HTTP

Routing adalah proses menentukan bagaimana aplikasi merespons permintaan ke endpoint tertentu. Express mendukung berbagai metode HTTP seperti:

- GET: Mengambil data dari server.
- POST: Mengirimkan data baru ke server.
- DELETE: Menghapus data dari server.

2. MAKSUD DAN TUJUAN

1. MAKSUD

Materi ini bertujuan untuk memperkenalkan konsep dasar pembuatan Web API menggunakan Node.js dengan framework Express. Web API (Application Programming Interface) adalah antarmuka yang memungkinkan aplikasi frontend (seperti React, mobile app, dll) berkomunikasi dengan backend.

2. TUJUAN

Setelah mempelajari dan mempraktikkan modul ini, peserta diharapkan mampu:

a. Menginisialisasi proyek Node.js dengan npm init.
b. Menginstal dan menggunakan Express untuk menangani routing HTTP (GET, POST, DELETE).
c. Membuat struktur folder sederhana agar proyek rapi dan terorganisir.
d. Menulis kode API sederhana yang menyimpan data mahasiswa dalam array (tanpa database).
e. Mencoba endpoint API secara lokal menggunakan node app.js.
f. Dengan mempraktikkan ini, kita bisa memahami bagaimana API bekerja, bagaimana menangani request dan response, serta bagaimana backend menerima dan mengolah data dari client.

BAB II

IMPLEMENTASI (GUIDED)

1. app.js

a. Code:

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

app.use(express.json());

// Simpan data mahasiswa di array static
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}`);
});
```

b. Output

```
PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 6\Praktikum\
Server berjalan di http://localhost:3000
```

c. Penjelasan

Kode di atas merupakan implementasi API sederhana menggunakan Node.js dan Express.js untuk mengelola data mahasiswa. Data mahasiswa disimpan dalam array statis, dan API menyediakan beberapa endpoint RESTful:

a. GET /api/mahasiswa untuk mengambil seluruh data.
b. GET /api/mahasiswa/:index untuk mengambil data berdasarkan indeks.
c. POST /api/mahasiswa untuk menambahkan data baru.
d. DELETE /api/mahasiswa/:index untuk menghapus data berdasarkan indeks.

Middleware express.json() digunakan untuk memproses data JSON dari body permintaan, dan server dijalankan pada port 3000.

BAB III

PENUGASAN (UNGUIDED)

1. Soal 1 Unguided

a. Code

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

app.use(express.json());

// Data awal mahasiswa
let mahasiswa = [
  { nama: "Arzario Irsyad", nim: "2211104032" },
  { nama: "Satria Ariq", nim: "2211104033" },
  { nama: "Adam Darmawan", nim: "2211104034" },
];

// 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}`);
});
```

b. Output

https://documenter.getpostman.com/view/40504903/2sB2j6Aqho

c. Penjelasan

Aplikasi REST API sederhana menggunakan Node.js dan Express yang menyediakan layanan CRUD (Create, Read, Delete) untuk data mahasiswa. Data disimpan dalam array JavaScript dan dapat diakses melalui endpoint HTTP seperti GET untuk melihat semua mahasiswa atau berdasarkan index, POST untuk menambahkan mahasiswa baru, serta DELETE untuk menghapus mahasiswa berdasarkan index. Middleware express.json() digunakan untuk memproses data JSON dari body request, dan server berjalan di localhost pada port 3000. API ini dapat diuji menggunakan Postman untuk mengirim dan menerima data dalam format JSON.