LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT BERGERAK

MODUL IX

MEMBUAT API DI NODE.JS

Disusun Oleh :

Yogi Hafidh Maulana

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

A. DASAR TEORI

1. Automata-Based Construction

Automata-based programming adalah paradigma pemrograman yang memandang program sebagai finite-state machine (FSM) atau formal automaton yang memiliki berbagai state yang saling berkaitan dan memiliki aturan tertentu yang jelas. Ciri-ciri utama Automata-Based Construction:

a. Pemrosesan berbasis state – Setiap eksekusi program dibatasi dalam satu state tanpa adanya eksekusi tumpang-tindih antar state.
b. Perpindahan antar state eksplisit – Komunikasi antara state hanya dapat dilakukan melalui variabel global atau mekanisme eksplisit lainnya.

2. Table-Driven Construction

Table-Driven Construction adalah teknik pemrograman yang menggantikan logic statements seperti if-else atau switch-case dengan tabel yang berisi informasi yang digunakan untuk pengambilan keputusan. Keunggulan Table-Driven Construction:

a. Mengurangi kompleksitas kode ketika banyak kondisi yang perlu diproses.
b. Mempermudah perubahan aturan atau logika hanya dengan mengubah tabel.

Metode Pencarian dalam Table-Driven Construction:

a. Direct Access – Menggunakan indeks langsung untuk mengakses nilai dari tabel.
b. Indexed Access – Menggunakan tabel tambahan untuk menyimpan indeks dari tabel utama agar lebih hemat memori.
c. Stair-step Access – Menggunakan tabel untuk mencocokkan rentang nilai dengan output tertentu.

B. 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)

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}`);
});
```

Output

```
D:\PROJECT\Kontruksi Perangkat Lunak\KPL_YogiHafidhMaulana_2211104061_SE062\09_Membuat
Server berjalan di http://localhost:3000
```

Deskripsi Code

Kode di atas adalah sebuah aplikasi backend sederhana menggunakan Express.js yang berjalan di port 3000. Aplikasi ini berfungsi sebagai API untuk mengelola data mahasiswa yang disimpan dalam array statis. Terdapat beberapa endpoint:

- GET /api/mahasiswa untuk menampilkan semua data mahasiswa,
- GET /api/mahasiswa/:index untuk menampilkan data mahasiswa berdasarkan index dalam array,
- POST /api/mahasiswa untuk menambahkan data mahasiswa baru melalui request body (dengan properti nama dan nim), dan
- DELETE /api/mahasiswa/:index untuk menghapus data mahasiswa berdasarkan index. Semua operasi ini dilakukan tanpa menggunakan database eksternal, hanya array biasa di memori. Saat saya mempelajari kode ini, saya melihat bagaimana REST API dibentuk dengan metode HTTP seperti GET, POST, dan DELETE, serta bagaimana Express digunakan untuk menangani routing dan middleware seperti express.json() untuk mem-parsing JSON dari request body. Ini membantu saya memahami dasar-dasar membuat API CRUD dengan Node.js.

BAB III
PENUGASAN (UNGUIDED)

Code

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

app.use(express.json());

// Data awal mahasiswa
let mahasiswa = [
  { nama: "Muhamad Taufiq Hidayat", nim: "1302010001" },
  { 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}`);
});
```

Output

Dokumentasi : https://documenter.getpostman.com/view/39435565/2sB2j4eAwy

Deskripsi Code

Pada proyek ini, saya membuat sebuah Web API sederhana menggunakan Node.js dan framework Express sebagai bagian dari tugas pemrograman backend. API ini berfungsi untuk mengelola data mahasiswa yang disimpan dalam bentuk array statis, tanpa menggunakan database. Saya mengimplementasikan beberapa endpoint seperti GET untuk menampilkan semua data atau data berdasarkan indeks, POST untuk menambahkan data baru, serta DELETE untuk menghapus data berdasarkan indeks. Semua data diuji menggunakan Postman, dimulai dari melihat data awal, menambahkan mahasiswa baru, sampai menghapus data yang telah ada. Proyek ini membantu saya memahami cara kerja server sederhana, bagaimana Express memproses permintaan HTTP, serta pentingnya pengujian API dalam pengembangan aplikasi backend.