LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 13

(Node.js: Pengenalan)

Oleh:

(Rengganis Tantri Pramudita) (2311104065)

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Node.js merupakan platform runtime berbasis JavaScript yang memungkinkan kode JavaScript dijalankan di sisi server menggunakan mesin JavaScript V8. Node.js dirancang dengan arsitektur event-driven dan non-blocking I/O sehingga mampu menangani banyak permintaan secara bersamaan tanpa menghambat proses lain. Dalam pengembangan aplikasi web modern, Node.js banyak digunakan untuk membangun backend dan layanan RESTful API karena performanya yang efisien serta dukungan ekosistem library yang luas melalui Node Package Manager (NPM). Untuk mempermudah pengembangan aplikasi server, Node.js sering dikombinasikan dengan framework Express.js yang menyediakan fitur routing, middleware, serta pengelolaan request dan response secara lebih terstruktur. Selain itu, Node.js dapat diintegrasikan dengan berbagai sistem basis data, salah satunya MySQL, untuk menyimpan dan mengelola data secara terpusat.

B. Tujuan

Tujuan dari pelaksanaan praktikum ini adalah untuk memahami konsep dasar pengembangan aplikasi backend menggunakan Node.js dan Express.js. Praktikum ini bertujuan agar mahasiswa mampu melakukan inisialisasi proyek Node.js, mengelola dependensi menggunakan NPM, serta membangun RESTful API yang dapat menangani operasi CRUD (Create, Read, Update, Delete). Selain itu, praktikum ini juga bertujuan untuk melatih mahasiswa dalam menghubungkan aplikasi backend dengan database MySQL sehingga data dapat disimpan, diambil, diperbarui, dan dihapus melalui endpoint API yang telah dibuat.

BAB II

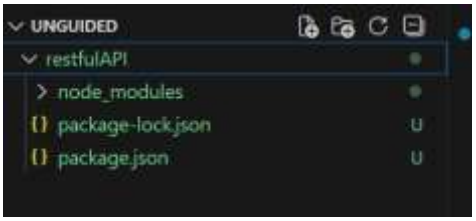
HASIL

A. GUIDED (Praktikum Terbimbing)

- Pengecekan Instalansi Node.js dan NPM

- Membuat Folder Proyek & Inisialisasi Proyek Node.js

- Instalasi Dependency



- Membuat database dan table

- Membuat file db.js

```
const mysql = require('mysql');
const connection = mysql.createConnection({
 host: 'localhost',
 user: 'root',
 password: '',
 database: 'akademic'
});
connection.connect(err => {
 if (err) {
 console.error('Koneksi database gagal:', err);
 return;
 }
 console.log('Database connected');
});
module.exports = connection;
```

```
PS C:\xampp\htdocs\PPW_2311104065_RengganisTantriPramudita\pertemuan13\unguided\restfulapi> node db.js
Database connected
```

- Membuat crud.js

```
const connection = require("./db");

// CREATE
function createMahasiswa(nama, nim, jurusan, email, callback) {
  const query =
    "INSERT INTO mahasiswa (nama, nim, jurusan, email) VALUES (?, ?, ?, ?)";
  connection.query(query, [nama, nim, jurusan, email], callback);
}

// READ
function getAllMahasiswa(callback) {
  const query = "SELECT * FROM mahasiswa";
  connection.query(query, callback);
}

// UPDATE
function updateMahasiswa(id, nama, nim, jurusan, email, callback) {
  const query =
    "UPDATE mahasiswa SET nama=?, nim=?, jurusan=?, email=? WHERE id=?";
  connection.query(query, [nama, nim, jurusan, email, id], callback);
}

// DELETE
function deleteMahasiswa(id, callback) {
```

```
  const query = "DELETE FROM mahasiswa WHERE id=?";
  connection.query(query, [id], callback);
}

module.exports = {
  createMahasiswa,
  getAllMahasiswa,
  updateMahasiswa,
  deleteMahasiswa,
};
```

- Membuat app.js

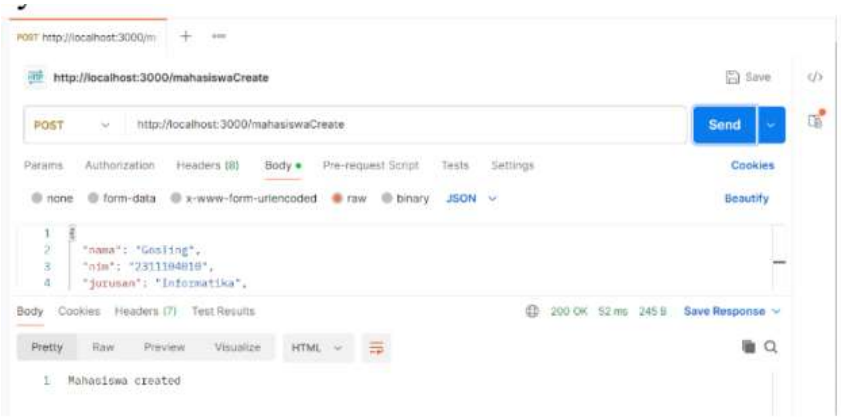
```
const express = require("express");
const db = require("./crud");
const path = require("path");

const app = express();
const port = 3000;

app.use(express.static(path.join(__dirname, "public")));

app.use(express.json());

// CREATE
app.post("/mahasiswaCreate", (req, res) => {
  const { nama, nim, jurusan, email } = req.body;
  db.createMahasiswa(nama, nim, jurusan, email, (err) => {
    if (err) return res.status(500).send("Error creating data");
    res.send("Mahasiswa created");
  });
});

// READ
app.get("/mahasiswaGet", (req, res) => {
  db.getAllMahasiswa((err, result) => {
    if (err) return res.status(500).send("Error fetching data");
    res.json(result);
  });
});

// UPDATE
app.put("/mahasiswaUpdate/:id", (req, res) => {
  const { id } = req.params;
  const { nama, nim, jurusan, email } = req.body;
  db.updateMahasiswa(id, nama, nim, jurusan, email, (err) => {
    if (err) return res.status(500).send("Error updating data");
    res.send("Mahasiswa updated");
  });
});

// DELETE
app.delete("/mahasiswaDelete/:id", (req, res) => {
  const { id } = req.params;
```

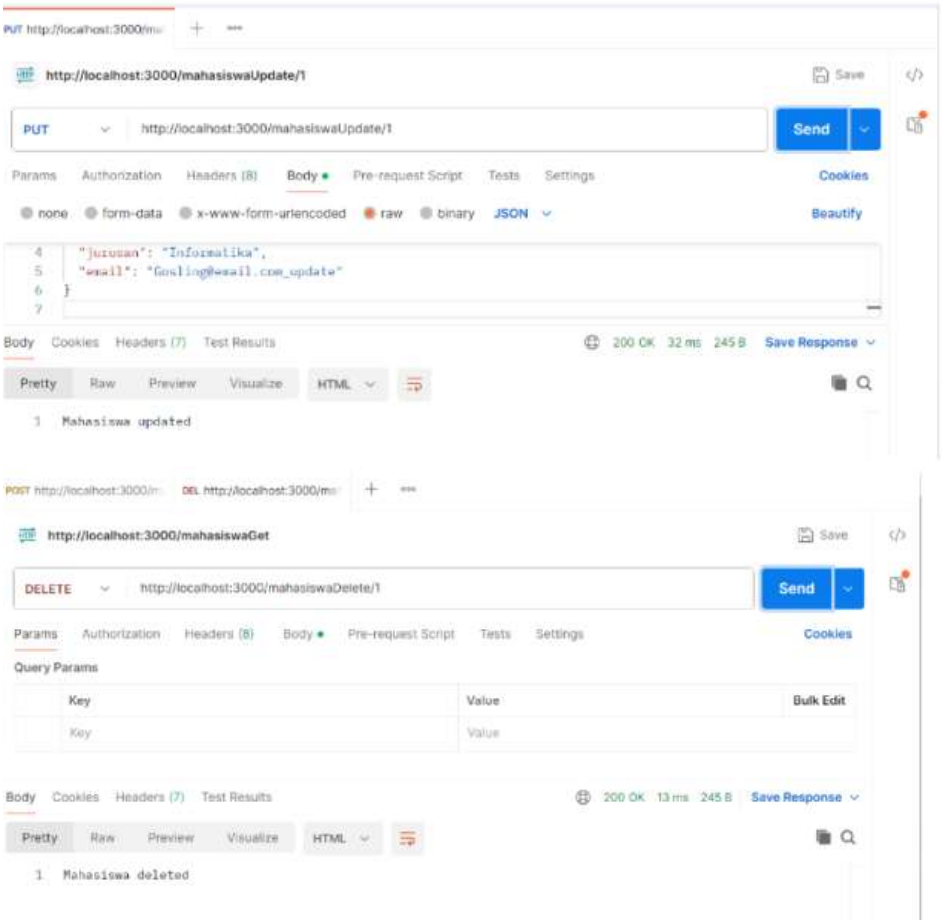
```
    db.deleteMahasiswa(id, (err) => {
      if (err) return res.status(500).send("Error deleting data");
      res.send("Mahasiswa deleted");
    });
  });

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

- Menjalankan server

```
PS C:\xampp\htdocs\PPW_2311104065_RengganisTantriPran
Server running on http://localhost:3000
```

- Melakukan pengujian dengan postman





B. UNGUIDED (Tugas Mandiri)

Mahasiswa diminta untuk membangun sebuah aplikasi web sederhana untuk pengelolaan data mahasiswa berbasis Node.js, Express.js, dan MySQL. Aplikasi harus menyediakan RESTful API serta dapat diakses melalui browser menggunakan halaman web sederhana (HTML dan JavaScript). Mahasiswa diperbolehkan menggunakan proyek yang telah dibuat di atas atau membuat proyek baru.

Jawab:

- Membuat file public/index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <title>Data Mahasiswa</title>
</head>
<body>
  <h2>Form Mahasiswa</h2>
```

```
<input type="text" id="nama" placeholder="Nama"><br><br>
<input type="text" id="nim" placeholder="NIM"><br><br>
<input type="text" id="jurusan" placeholder="Jurusan"><br><br>
<input type="email" id="email" placeholder="Email"><br><br>

<button onclick="tambahMahasiswa()">Tambah Data</button>

<h2>Daftar Mahasiswa</h2>
<ul id="listMahasiswa"></ul>

<script>
  function loadMahasiswa() {
    fetch("/mahasiswaGet")
      .then(res => res.json())
      .then(data => {
        const list = document.getElementById("listMahasiswa");
        list.innerHTML = "";
        data.forEach(m => {
          list.innerHTML += `
            <li>
              ${m.nama} (${m.nim})
              <button onclick="hapusMahasiswa(${m.id})">Hapus</button>
            </li>
          `;
        });
      });
  }

  function tambahMahasiswa() {
    const data = {
      nama: document.getElementById("nama").value,
      nim: document.getElementById("nim").value,
      jurusan: document.getElementById("jurusan").value,
      email: document.getElementById("email").value
    };

    fetch("/mahasiswaCreate", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(data)
    }).then(() => {
      loadMahasiswa();
      alert("Data berhasil ditambahkan");
    });
  }

  function hapusMahasiswa(id) {
    fetch(`/mahasiswaDelete/${id}`, {
```

```
                method: "DELETE"
              }).then(() => {
                loadMahasiswa();
              });
            }

            loadMahasiswa();
          </script>
        </body>
</html>
```

Penjelasan:

Kode HTML dan JavaScript tersebut digunakan sebagai antarmuka web sederhana untuk mengelola data mahasiswa yang terhubung dengan RESTful API berbasis Node.js. Halaman ini menyediakan form input untuk memasukkan data mahasiswa berupa nama, NIM, jurusan, dan email yang kemudian dikirim ke server menggunakan metode HTTP POST melalui fungsi fetch. Data mahasiswa yang tersimpan di database ditampilkan secara dinamis pada halaman web dengan memanfaatkan metode HTTP GET tanpa perlu melakukan refresh halaman. Selain itu, pengguna juga dapat menghapus data mahasiswa berdasarkan ID menggunakan metode HTTP DELETE. Dengan demikian, halaman web ini berfungsi sebagai frontend yang memungkinkan pengguna melakukan operasi CRUD (Create, Read, Delete) secara langsung melalui komunikasi asinkron antara frontend dan backend menggunakan format data JSON.

- Membuat app.js

```
const express = require("express");
const db = require("./crud");
const path = require("path");

const app = express();
const port = 3000;

app.use(express.static(path.join(__dirname, "public")));

app.use(express.json());

// CREATE
app.post("/mahasiswaCreate", (req, res) => {
  const { nama, nim, jurusan, email } = req.body;
  db.createMahasiswa(nama, nim, jurusan, email, (err) => {
```

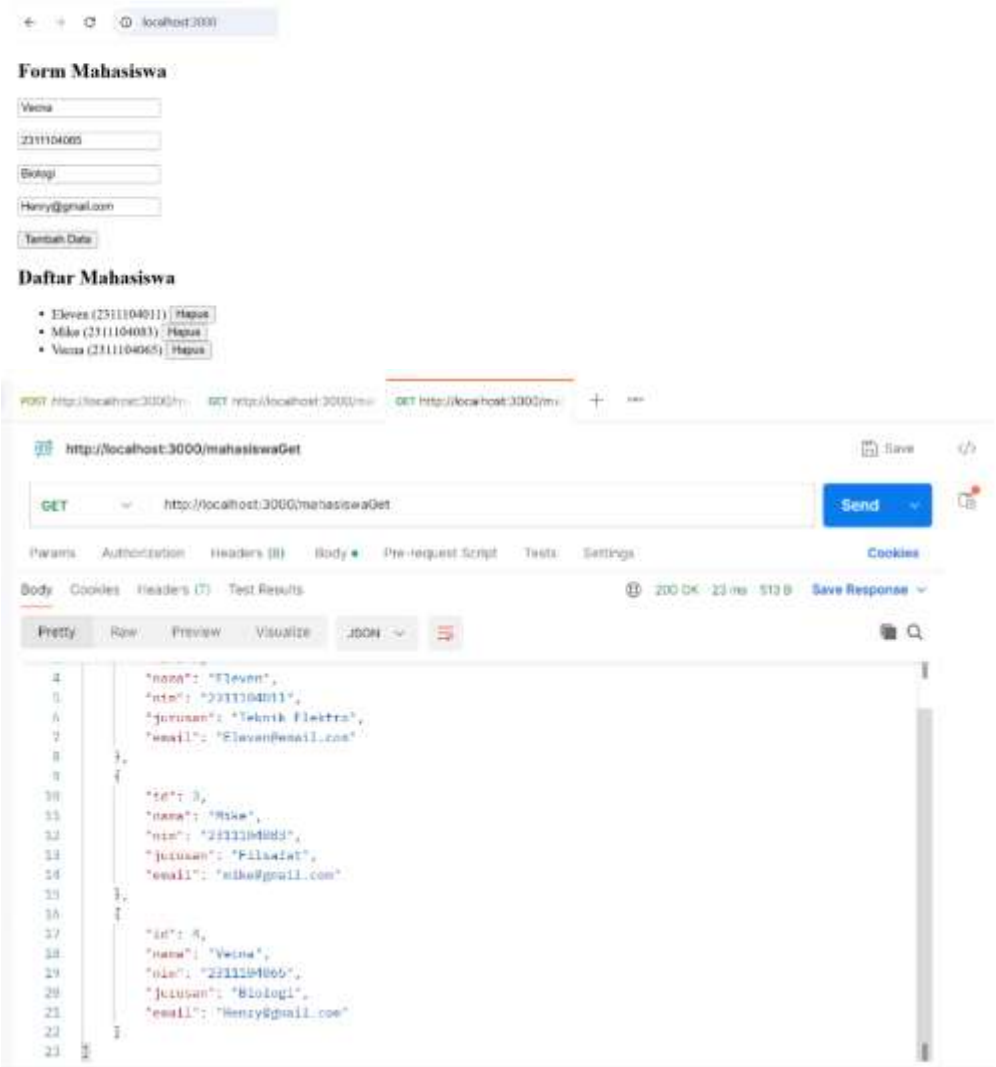
```
    if (err) return res.status(500).send("Error creating data");
    res.send("Mahasiswa created");
  });
});

// READ
app.get("/mahasiswaGet", (req, res) => {
  db.getAllMahasiswa((err, result) => {
    if (err) return res.status(500).send("Error fetching data");
    res.json(result);
  });
});

// UPDATE
app.put("/mahasiswaUpdate/:id", (req, res) => {
  const { id } = req.params;
  const { nama, nim, jurusan, email } = req.body;
  db.updateMahasiswa(id, nama, nim, jurusan, email, (err) => {
    if (err) return res.status(500).send("Error updating data");
    res.send("Mahasiswa updated");
  });
});

// DELETE
app.delete("/mahasiswaDelete/:id", (req, res) => {
  const { id } = req.params;
  db.deleteMahasiswa(id, (err) => {
    if (err) return res.status(500).send("Error deleting data");
    res.send("Mahasiswa deleted");
  });
});

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

Penjelasan:

Kode app.js tersebut berfungsi sebagai server utama aplikasi backend yang dibangun menggunakan Node.js dan framework Express.js. Pada kode ini, Express digunakan untuk menangani request HTTP serta mengatur routing RESTful API yang mendukung operasi CRUD data mahasiswa. Middleware express.static digunakan untuk menampilkan file frontend dari folder public, sedangkan express.json() berfungsi untuk memproses data JSON yang dikirim dari client. Server menyediakan endpoint untuk menambahkan data mahasiswa (POST),

menampilkan seluruh data mahasiswa (GET), memperbarui data berdasarkan ID (PUT), dan menghapus data berdasarkan ID (DELETE) dengan memanggil fungsi-fungsi yang terdapat pada file crud.js. Seluruh endpoint tersebut dihubungkan dengan database melalui lapisan logika CRUD, dan server dijalankan pada port 3000 sehingga aplikasi dapat diakses melalui browser maupun tools pengujian API seperti Postman.

Output



BAB III

PENUTUP

A. Kesimpulan

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa Node.js yang dikombinasikan dengan Express.js merupakan solusi yang efektif untuk membangun aplikasi backend dan RESTful API. Aplikasi yang dikembangkan mampu menjalankan operasi CRUD dengan baik serta terhubung ke database MySQL secara stabil. Melalui praktikum ini, mahasiswa dapat memahami alur kerja backend mulai dari inisialisasi proyek, pembuatan server, pengelolaan routing, hingga integrasi basis data. Dengan demikian, Node.js terbukti mendukung pengembangan aplikasi web yang efisien, terstruktur, dan mudah dikembangkan lebih lanjut sesuai kebutuhan sistem.