LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 13

NODE.JS: PENGENALAN

Oleh:

Aulia Jasifa Br Ginting 2311104060

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Node.js merupakan sebuah platform runtime yang memungkinkan bahasa pemrograman JavaScript dijalankan di sisi server. Berbeda dengan JavaScript pada umumnya yang hanya berjalan di browser, Node.js memungkinkan JavaScript digunakan untuk membangun aplikasi backend seperti web server, RESTful API, dan aplikasi berbasis jaringan. Node.js pertama kali dikembangkan oleh Ryan Dahl pada tahun 2009 dan dibangun di atas mesin JavaScript V8 milik Google yang dikenal memiliki performa tinggi dalam mengeksekusi kode JavaScript.

Salah satu keunggulan utama Node.js adalah penerapan konsep event-driven dan non-blocking I/O. Model ini memungkinkan Node.js menangani banyak permintaan secara bersamaan tanpa harus menunggu satu proses selesai terlebih dahulu. Berbeda dengan pendekatan tradisional yang bersifat blocking, Node.js menggunakan mekanisme asynchronous sehingga server tetap responsif meskipun menangani banyak klien secara simultan. Hal ini menjadikan Node.js sangat efisien dan cocok digunakan untuk aplikasi web modern yang membutuhkan performa tinggi dan skalabilitas yang baik.

Node.js juga didukung oleh ekosistem package yang sangat besar melalui NPM (Node Package Manager). Dengan NPM, pengembang dapat dengan mudah menginstal dan mengelola berbagai modul atau library yang dibutuhkan dalam pengembangan aplikasi. Selain itu, Node.js menyediakan berbagai modul bawaan seperti http, fs, dan path yang memudahkan pembuatan server, pengelolaan file, dan manipulasi direktori tanpa perlu menggunakan library tambahan.

Dalam praktikum ini, Node.js digunakan untuk membuat server sederhana menggunakan modul bawaan http. Server ini berfungsi untuk menerima permintaan dari klien melalui browser dan memberikan respons berupa teks. Praktikum ini bertujuan untuk memberikan pemahaman dasar mengenai cara kerja Node.js sebagai server-side JavaScript serta konsep dasar pembuatan web server menggunakan Node.js.

B. Tujuan

1. Ketepatan penerapan pembangunan aplikasi web menggunakan NodeJS.
2. Ketepatan penerapan pembangunan aplikasi web menggunakan Framework NodeJS

BAB II

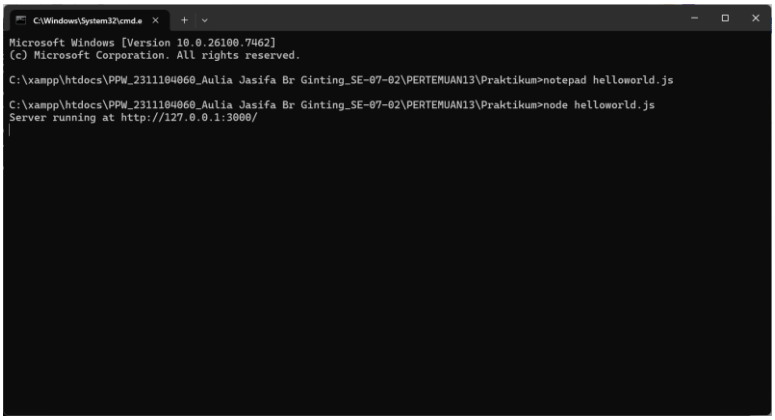
HASIL

A. GUIDED (Praktikum Terbimbing)

a. Contoh node.js

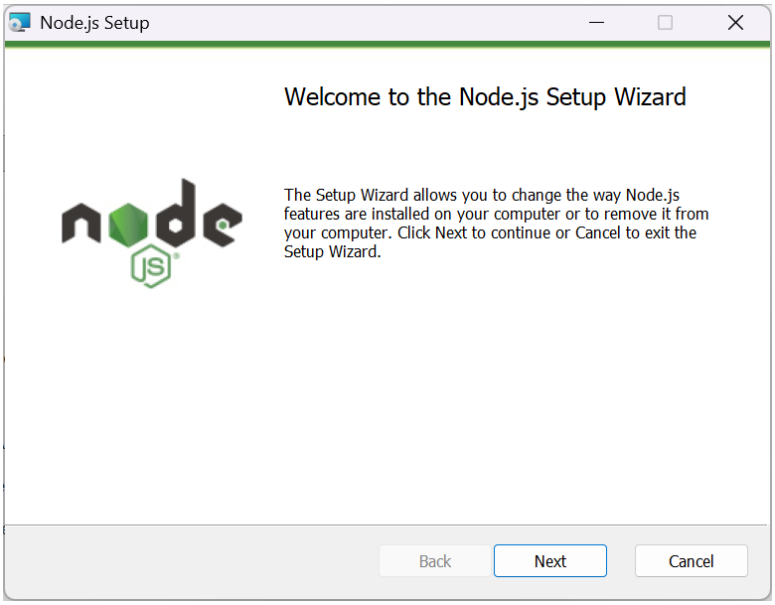
```
JS helloworld.js > ...
1   const { createServer } = require("node:http");
2
3   const hostname = "127.0.0.1";
4   const port = 3000;
5
6   const server = createServer((req, res) => {
7     res.statusCode = 200;
8     res.setHeader("Content-Type", "text/plain");
9     res.end("Hello World");
10  });
11
12  server.listen(port, hostname, () => {
13    console.log(`Server running at http://${hostname}:${port}/`);
14  });
```

Menjalankan code diatas



b. Inisialisasi Proyek Node.js

Install node.js



Cek node.js sudah terpasang

```
C:\Users\LENOVO>node -v
v24.12.0

C:\Users\LENOVO>npm -v
11.6.2
```

1. Membuat Folder Proyek

```
C:\xampp\htdocs\PPW_2311104060_Aulia Jasifa Br Ginting_SE-07-02\PERTEMUAN13\Praktikum>mkdir restfulAPI
```

2. Inisialisasi Proyek Node.js

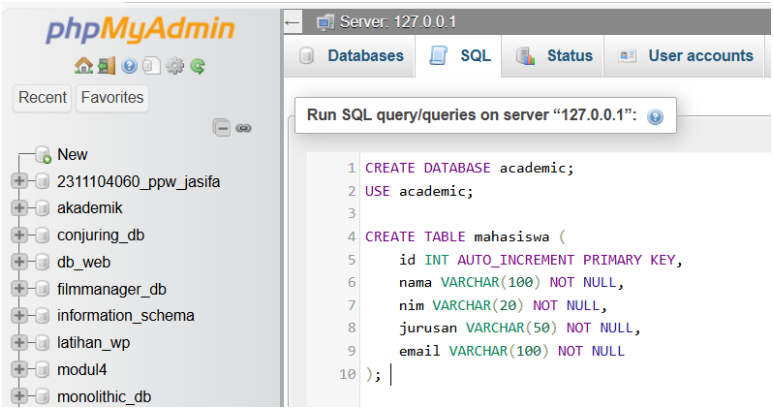
```
C:\xampp\htdocs\PPW_2311104060_Aulia Jasifa Br Ginting_SE-07-02\PERTEMUAN13\Praktikum\restfulAPI>npm init -y
Wrote to C:\xampp\htdocs\PPW_2311104060_Aulia Jasifa Br Ginting_SE-07-02\PERTEMUAN13\Praktikum\restfulAPI\package.json:

{
  "name": "restfulapi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}
```

3. Instal Depedency

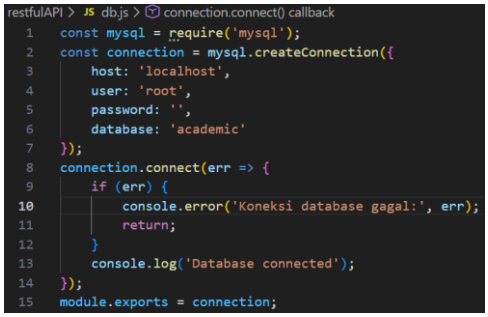
```
C:\xampp\htdocs\PPW_2311104060_Aulia Jasifa Br Ginting_SE-07-02\PERTEMUAN13\Praktikum\restfulAPI>npm install express mys
ql

added 75 packages, and audited 76 packages in 12s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

c. Konfigration Database

1. Membuat Database dan Tabel



2. Membuat File Koneksi Database (db.js)



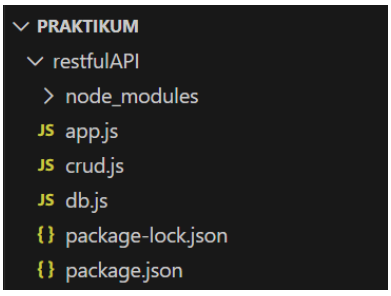
d. Restful API

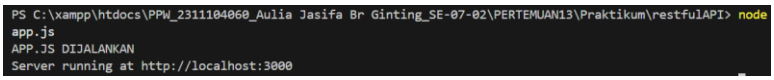
1. Implementasi Operasi CRUD

2. Pembuatan File Utama Aplikasi (app.js)

```js
const express = require("express");
const dbOperations = require("./crud");

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

app.use(express.json());

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

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

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

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

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

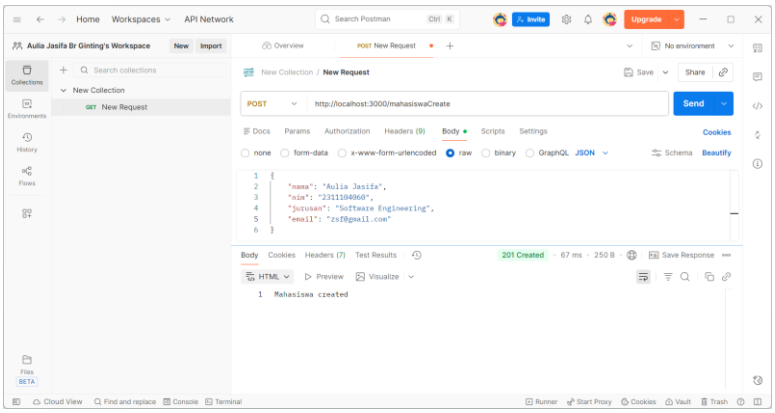
struktur proyek node.js



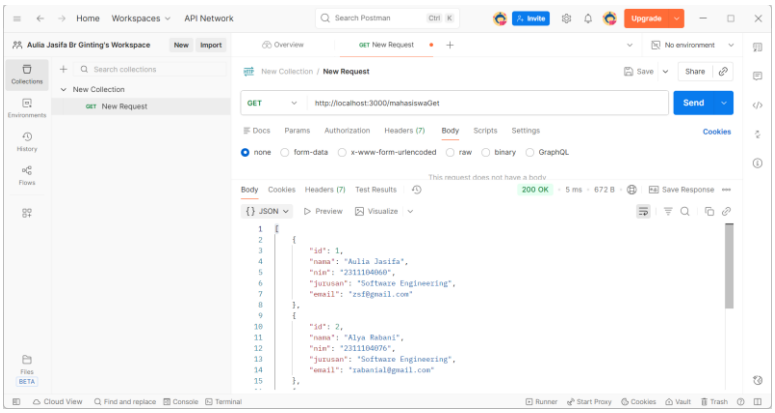


```
PS C:\xampp\htdocs\PPW_2311104060_Aulia Jasifa Br Ginting_SE-07-02\PERTEMUAN13\Praktikum\restfulAPI> node
app.js
APP.JS DIJALANKAN
Server running at http://localhost:3000
```

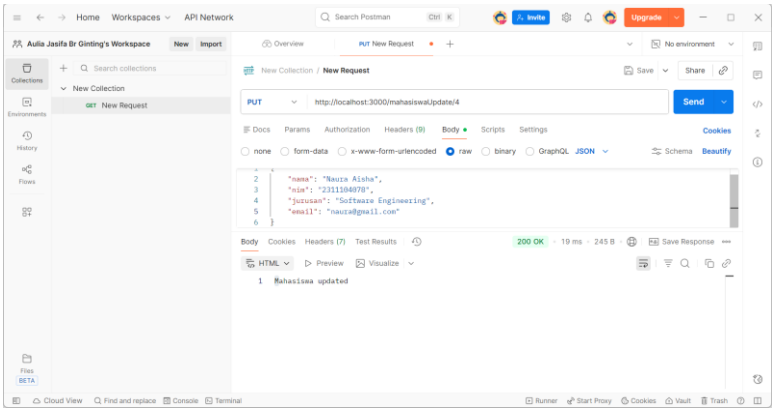
API POT



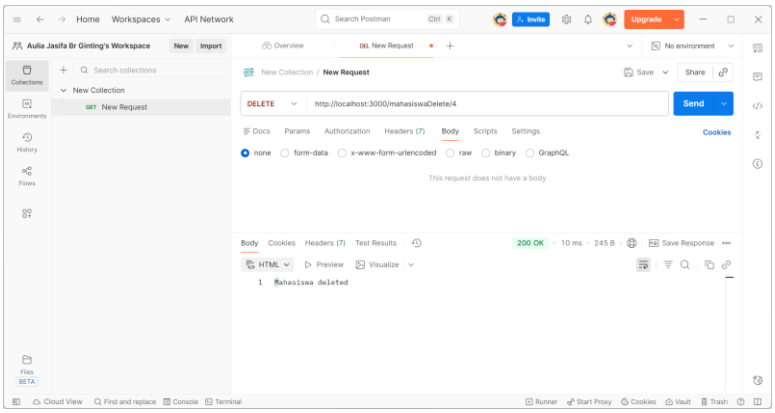
API GET

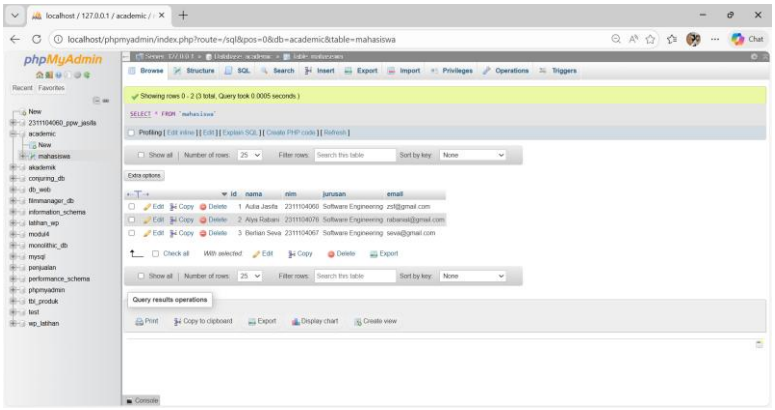


API PUT



API DELETE





Penjelasan:

Kode program pada praktikum ini berfungsi untuk membuat sebuah server web sederhana menggunakan modul bawaan http pada Node.js. Server tersebut dikonfigurasikan untuk berjalan pada alamat 127.0.0.1 dengan port 3000. Ketika server dijalankan, Node.js akan mendengarkan permintaan (request) yang dikirimkan oleh klien, seperti browser, dan memberikan respons sesuai dengan logika yang telah ditentukan di dalam kode.

Pada saat server menerima permintaan, fungsi callback akan dieksekusi untuk memproses request dan mengirimkan response. Server akan mengatur status HTTP menjadi 200 yang menandakan bahwa permintaan berhasil diproses, kemudian mengatur header Content-Type menjadi text/plain agar browser memahami bahwa respons yang dikirim berupa teks biasa. Setelah itu, server mengirimkan isi respons berupa teks “Hello World” kepada klien. Proses ini berlangsung secara asynchronous, sehingga server tetap dapat menangani permintaan lain tanpa terhambat.

Ketika server berhasil dijalankan, Node.js akan menampilkan pesan pada terminal yang menunjukkan bahwa server sedang aktif dan dapat diakses melalui browser. Dengan mengakses alamat dan port yang telah ditentukan, klien dapat melihat hasil respons yang dikirimkan oleh server. Melalui praktikum ini, mahasiswa dapat memahami alur dasar kerja server Node.js mulai dari pembuatan server, pemrosesan request, hingga pengiriman response kepada klien.

B. UNGUIDED (Tugas Mandiri)

1. Soal 1:

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.

Jawaban:

Class db.js

restfulAPI > JS db.js > ...

```
1   const mysql = require("mysql");
2
3   const connection = mysql.createConnection({
4     host: "localhost",
5     user: "root",
6     password: "",
7     database: "pendidikan",
8   });
9
10  connection.connect(err => {
11    if (err) {
12      console.error("Koneksi database gagal", err);
13      return;
14    }
15    console.log("Database connected");
16  });
17
18  module.exports = connection;
```

Class 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], (error,
        results) => {
        if (error) {
            return callback(error, null);
        }
        callback(null, results);
    });
}

// Read
function getAllMahasiswa(callback) {
    const query = "SELECT * FROM mahasiswa";
    connection.query(query, (error, results) => {
        if (error) {
            return callback(error, null);
        }
        callback(null, results);
    });
}

// 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], (error,
        results) => {
        if (error) {
            return callback(error, null);
        }
        if (results.affectedRows === 0) {
            return callback(new Error("No rows updated, ID may not exist"),
                null);
        }
        callback(null, results);
    });
}

// Delete
function deleteMahasiswa(id, callback) {
    const query = "DELETE FROM mahasiswa WHERE id = ?";
    connection.query(query, [id], (error, results) => {
        if (error) {
            return callback(error, null);
        }
        callback(null, results);
    });
}

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

Class app,js

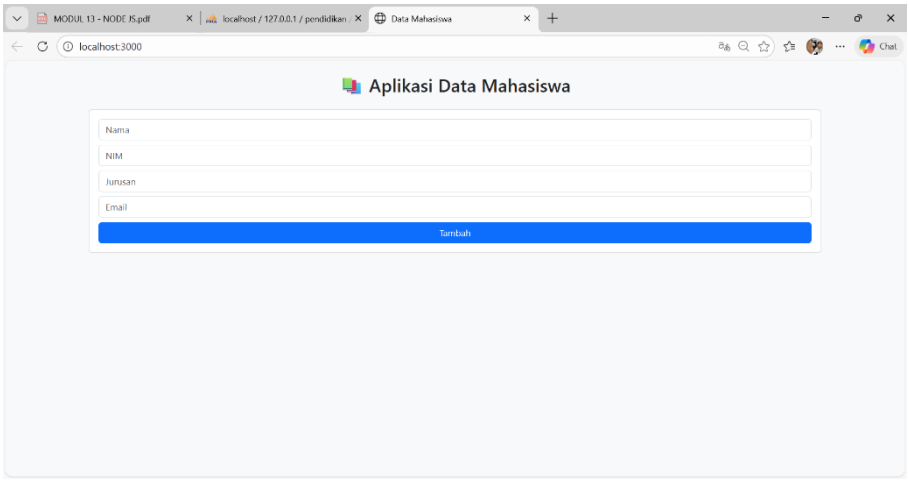
```js
const express = require("express");
const mysql = require("mysql");
const app = express();

app.use(express.json());
app.use(express.static("public"));

const db = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "pendidikan"
});

db.connect(err => {
  if (err) throw err;
  console.log("MySQL Connected");
});

// GET
app.get("/api/mahasiswa", (req, res) => {
  db.query("SELECT * FROM mahasiswa", (err, result) => {
    if (err) return res.status(500).send(err);
    res.json(result);
  });
});

// POST
app.post("/api/mahasiswa", (req, res) => {
  const { nama, nim, jurusan, email } = req.body;

  const sql = "INSERT INTO mahasiswa (nama, nim, jurusan, email) VALUES (?, ?, ?, ?)";
  db.query(sql, [nama, nim, jurusan, email], err => {
    if (err) return res.status(500).send(err);
    res.send("Data berhasil ditambahkan");
  });
});

// PUT
app.put("/api/mahasiswa/:id", (req, res) => {
  const { id } = req.params;
  const { nama, nim, jurusan, email } = req.body;

  const sql = `
    UPDATE mahasiswa
    SET nama=?, nim=?, jurusan=?, email=?
    WHERE id=?
  `;

  db.query(sql, [nama, nim, jurusan, email, id], (err, result) => {
    if (err) return res.status(500).send(err);

    if (result.affectedRows === 0) {
      return res.status(404).send("Data tidak ditemukan");
    }

    res.send("Data berhasil diupdate");
  });
});

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

  db.query("DELETE FROM mahasiswa WHERE id=?", [id], (err, result) => {
    if (err) return res.status(500).send(err);

    if (result.affectedRows === 0) {
      return res.status(404).send("Data tidak ditemukan");
    }

    res.send("Data berhasil dihapus");
  });
});

const port = 3000;

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

Class public/index.html

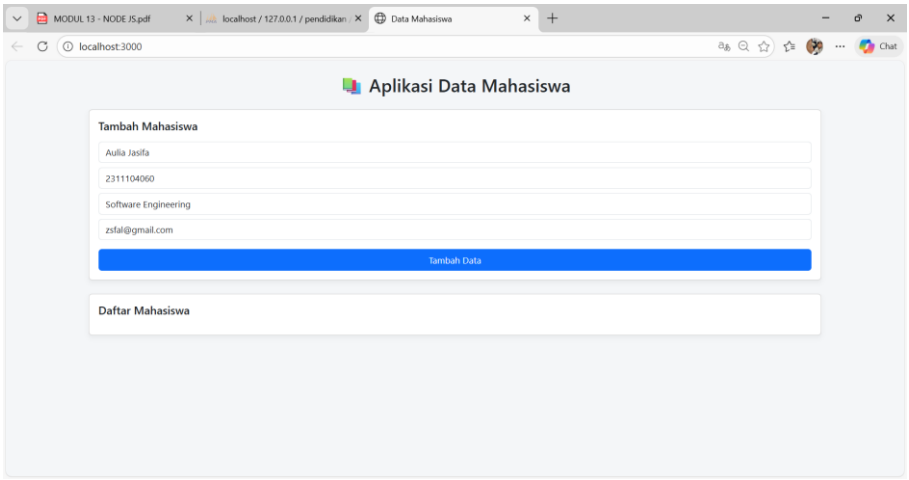
```html
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <title>Data Mahasiswa</title>

  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

  <style>
    body {
      background-color: #f4f6f8;
    }
    .card {
      box-shadow: 0 4px 8px rgba(0,0,0,0.05);
    }
  </style>
</head>
<body>

<div class="container mt-4">
  <h2 class="text-center mb-4">📚 Aplikasi Data Mahasiswa</h2>

  <!-- FORM TAMBAH -->
  <div class="card mb-4">
    <div class="card-body">
      <h5 class="mb-3">Tambah Mahasiswa</h5>

      <input id="nama" class="form-control mb-2" placeholder="Nama">
      <input id="nim" class="form-control mb-2" placeholder="NIM">
      <input id="jurusan" class="form-control mb-2" placeholder="Jurusan">
      <input id="email" class="form-control mb-3" placeholder="Email">

      <button class="btn btn-primary w-100" onclick="tambah()">Tambah Data</button>
    </div>
  </div>

  <!-- LIST DATA -->
  <div class="card">
    <div class="card-body">
      <h5 class="mb-3">Daftar Mahasiswa</h5>
      <ul class="list-group" id="list"></ul>
    </div>
  </div>
</div>

<!-- Bootstrap JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

<script>

// GET
function loadData() {
  fetch("/api/mahasiswa")
    .then(res => res.json())
    .then(data => {
      const list = document.getElementById("list");
      list.innerHTML = "";

      data.forEach(m => {
        list.innerHTML += `
          <li class="list-group-item">
            <b>${m.nama}</b><br>
            NIM: ${m.nim}<br>
            Jurusan: ${m.jurusan}<br>
            Email: ${m.email}<br>
            <button class="btn btn-danger btn-sm mt-2" onclick="hapus(${m.id})">Hapus</button>
          </li>
        `;
      });
    });
}

// POST
function tambah() {
  fetch("/api/mahasiswa", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      nama: nama.value,
      nim: nim.value,
      jurusan: jurusan.value,
      email: email.value
    })
  }).then(() => {
    document.querySelectorAll("input").forEach(i => i.value = "");
    loadData();
  });
}

// DELETE
function hapus(id) {
  if (confirm("Yakin ingin menghapus data ini?")) {
    fetch(`/api/mahasiswa/${id}`, { method: "DELETE" })
      .then(() => loadData());
  }
}

// PUT
function updateData() {
  const id = editId.value;

  fetch(`/api/mahasiswa/${id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      nama: editNama.value,
      nim: editNim.value,
      jurusan: editJurusan.value,
      email: editEmail.value
    })
  }).then(() => {
    editModal.hide();
    loadData();
  });
}
</script>

</body>
</html>

```

Output:

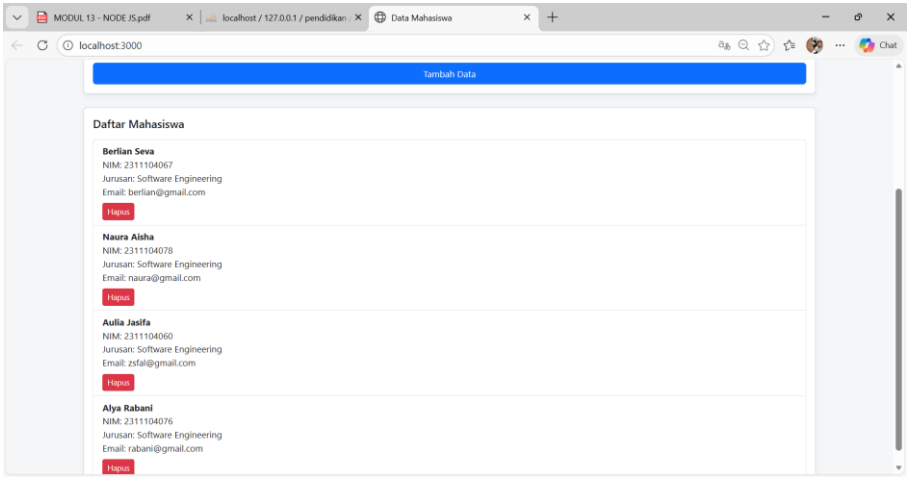
Tampilan awal dari web



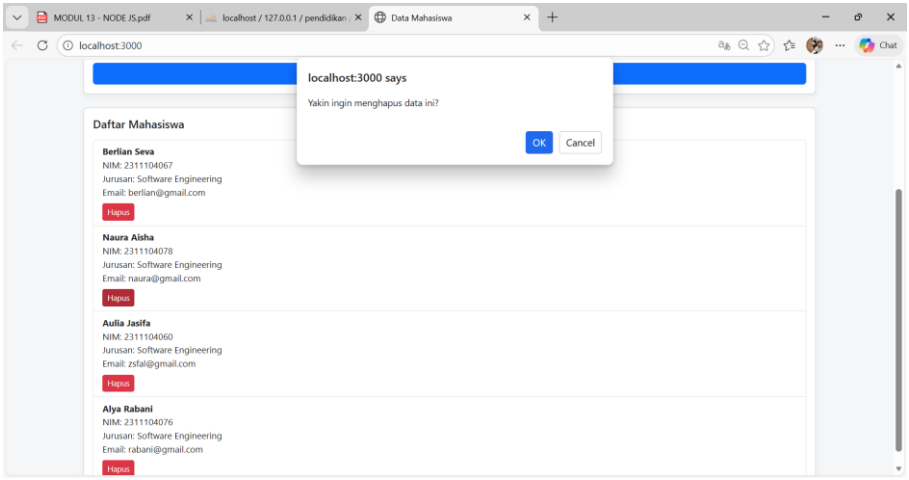
Menambahkan data mahasiswa



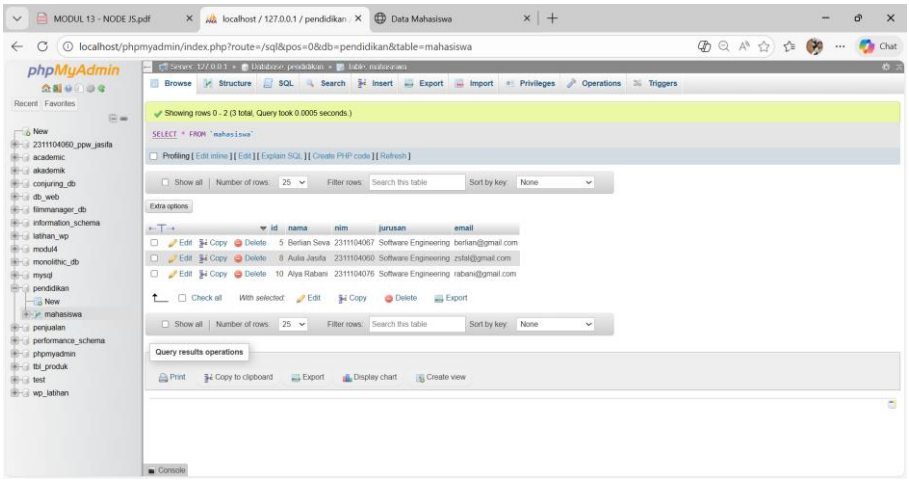
Data mahasiswa



Menghapus salah satu data mahasiswa



Data tersimpan



Penjelasan:

Aplikasi yang dikembangkan pada tugas unguided ini merupakan aplikasi web sederhana untuk pengelolaan data mahasiswa berbasis Node.js, Express.js, dan MySQL dengan konsep RESTful API. Node.js digunakan sebagai runtime JavaScript di sisi server, sedangkan Express.js berperan sebagai framework untuk menangani routing dan permintaan HTTP. MySQL digunakan sebagai media penyimpanan data mahasiswa.

Aplikasi menyediakan empat endpoint utama sesuai konsep CRUD, yaitu GET untuk menampilkan seluruh data mahasiswa dari database, POST untuk menambahkan data mahasiswa baru, PUT untuk memperbarui data mahasiswa berdasarkan ID, dan DELETE untuk menghapus data mahasiswa tertentu. Setiap

endpoint diimplementasikan menggunakan query SQL yang dijalankan melalui koneksi MySQL.

Pada sisi frontend, aplikasi menggunakan HTML, JavaScript, dan Bootstrap untuk menampilkan antarmuka pengguna. Data mahasiswa ditampilkan secara dinamis melalui pemanggilan API menggunakan fungsi fetch(). Proses edit data dilakukan melalui modal form sehingga pengguna dapat memperbarui seluruh data dalam satu tampilan, sedangkan penghapusan data dilakukan melalui tombol hapus yang memanggil endpoint DELETE. Dengan mekanisme ini, aplikasi mampu mengelola data mahasiswa secara dinamis dan terintegrasi antara frontend dan backend.

BAB III

PENUTUP

A. Kesimpulan

Berdasarkan hasil praktikum dan tugas mandiri (unguided), dapat disimpulkan bahwa penggunaan Node.js dan Express.js memungkinkan pengembangan aplikasi web berbasis server-side JavaScript secara efisien. Penerapan konsep RESTful API dengan metode GET, POST, PUT, dan DELETE memudahkan proses pengelolaan data mahasiswa secara terstruktur sesuai konsep CRUD. Selain itu, integrasi dengan MySQL sebagai database mampu menyimpan dan mengelola data secara persisten.

Selama proses pengerjaan, beberapa kendala yang dihadapi antara lain kesalahan konfigurasi server, ketidaksesuaian antara parameter endpoint dengan primary key pada database, serta error pada proses update dan delete data. Kendala tersebut dapat diatasi dengan melakukan pengecekan struktur tabel database, memastikan penggunaan ID sebagai primary key, serta melakukan debugging pada route Express dan query SQL. Dari kegiatan ini, mahasiswa memperoleh pemahaman yang lebih baik mengenai alur kerja aplikasi web dari sisi frontend hingga backend serta pentingnya sinkronisasi antara antarmuka pengguna dan database.