

**LAPORAN PRAKTIKUM
PERANCANGAN DAN PEMROGRAMAN WEB**

MODUL 13

NODE.JS



Oleh:

Dhiemas Tulus Ikhsan - 2311104046

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

BAB I

PENDAHULUAN

1.1 Dasar Teori

Node.js adalah platform runtime yang memperluas fungsionalitas JavaScript agar dapat dioperasikan pada sisi server. Jika biasanya JavaScript hanya terbatas pada lingkungan browser, Node.js membuka peluang bagi pengembang untuk merancang infrastruktur backend, mulai dari server web hingga API yang kompleks. Teknologi ini diperkenalkan oleh Ryan Dahl pada 2009 dengan memanfaatkan kecepatan mesin V8 Google, yang secara teknis mampu mengonversi kode JavaScript menjadi instruksi mesin dengan sangat cepat.

Kekuatan utama platform ini terletak pada arsitektur event-driven dan sistem non-blocking I/O. Melalui mekanisme ini, Node.js mampu mengelola ribuan koneksi secara simultan tanpa mengalami hambatan antrean proses, berbeda dengan server konvensional yang seringkali tertahan saat menunggu satu tugas selesai. Pendekatan asinkron ini memastikan aplikasi tetap lincah dan responsif, menjadikannya solusi ideal bagi sistem web modern yang menuntut performa tinggi serta kemudahan dalam peningkatan kapasitas (scalability).

Pengembangan aplikasi menggunakan Node.js juga sangat terbantu oleh kehadiran NPM (Node Package Manager), sebuah gudang pustaka digital yang memudahkan pengelolaan berbagai modul tambahan. Selain dukungan komunitas yang masif tersebut, Node.js juga sudah dilengkapi dengan fungsionalitas internal seperti modul http untuk komunikasi data, fs untuk manajemen dokumen, dan path untuk pengaturan alamat direktori. Keberadaan fitur bawaan ini memungkinkan pengembang membangun aplikasi fungsional tanpa harus selalu bergantung pada pihak ketiga.

Pada sesi praktikum kali ini, Node.js dimanfaatkan untuk merancang sebuah server minimalis melalui modul http. Fokus utamanya adalah memahami interaksi antara permintaan dari pengguna dan tanggapan yang dihasilkan oleh server dalam bentuk teks. Melalui latihan ini, diharapkan muncul pemahaman mendasar mengenai peran JavaScript di ranah server-side serta prinsip-prinsip awal dalam mengelola komunikasi data melalui web server berbasis Node.js.

1.2 Tujuan

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

BAB II

HASIL PRAKTIKUM

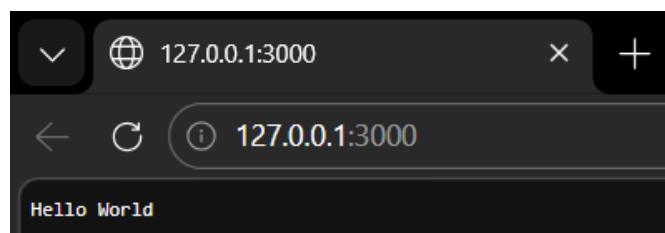
2.1 GUIDED (Praktikum Terbimbing)

a. Contoh [node.js](#)

```
Pertemuan_13 > 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 });
15
```

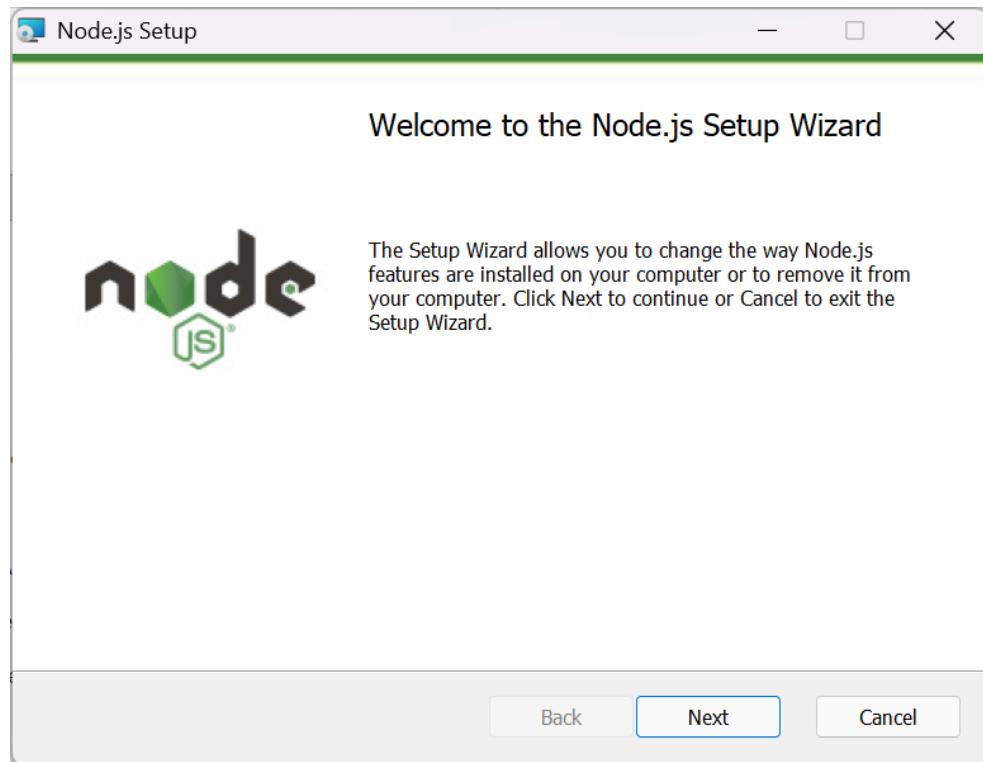
Menjalankan code diatas

```
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web> cd Pertemuan_13
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13> notepad helloworld.js
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13> node helloworld.js
Server running at http://127.0.0.1:3000/
|
```



b. Inisialisasi Proyek [Node.js](#)

Install [node.js](#)



Cek node.js sudah terpasang

```
C:\Users\Lenovo>node -v
v21.2.0

C:\Users\Lenovo>npm -v
10.2.3
```

1. Membuat folder proyek

```
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web> cd Pertemuan_13
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13> cd guided
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13\guided> mkdir restfulAPI
```

2. Inisialisasi Proyek node.js

```
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13\guided> npm init -y
Wrote to C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13\guided\package.json:

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

3. Instal Depedency

```
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13\guided> npm install express mysql
added 75 packages, and audited 76 packages in 4s
22 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

c. Konfigurasi Database & Koneksi Database

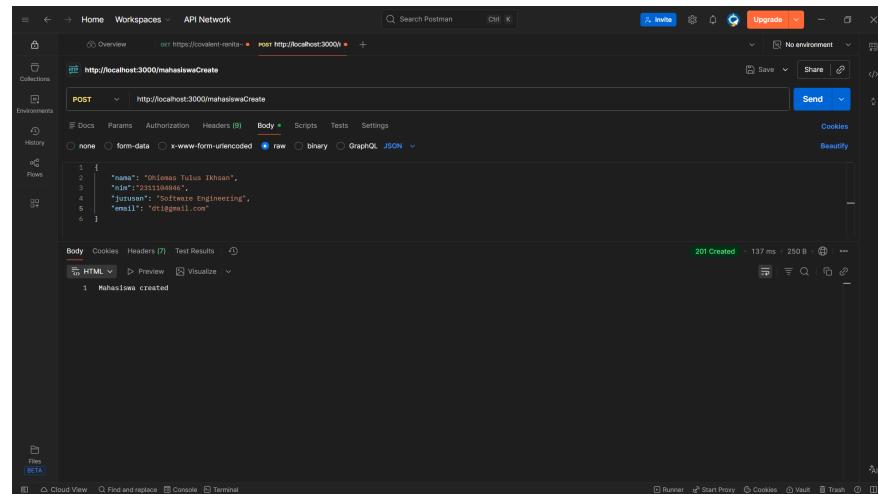
The screenshot shows the phpMyAdmin interface. On the left, the database structure is listed under 'New': 231104046_ppw_tulus, db_tpqsmart, information_schema, mysql, performance_schema, perpustakaan_db, phpmyadmin, tbl_product, and test. On the right, the 'Run SQL query/queries on server "127.0.0.1":' section contains the following SQL code:

```
1 CREATE DATABASE academic;
2 USE academic;
3
4 CREATE TABLE mahasiswa (
5     id INT AUTO_INCREMENT PRIMARY KEY,
6     nama VARCHAR(100) NOT NULL,
7     nim VARCHAR(20) NOT NULL,
8     jurusan VARCHAR(50) NOT NULL,
9     email VARCHAR(100) NOT NULL
10 );|
```

```
Pertemuan_13 > guided > restfulAPI > js dbjs > ...
1  const mysql = require('mysql'); //
2
3  const connection = mysql.createConnection({ //
4      host: 'localhost', //
5      user: 'root', //
6      password: '', //
7      database: 'academic' //
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; //
```

```
PS C:\xampp\htdocs\Praktikum Perancangan dan Pemrograman Web\Pertemuan_13\guided\restfulAPI> node app.js
APP.JS DIJALANKAN
Server running on http://localhost:3000
Database connected
```

API POST

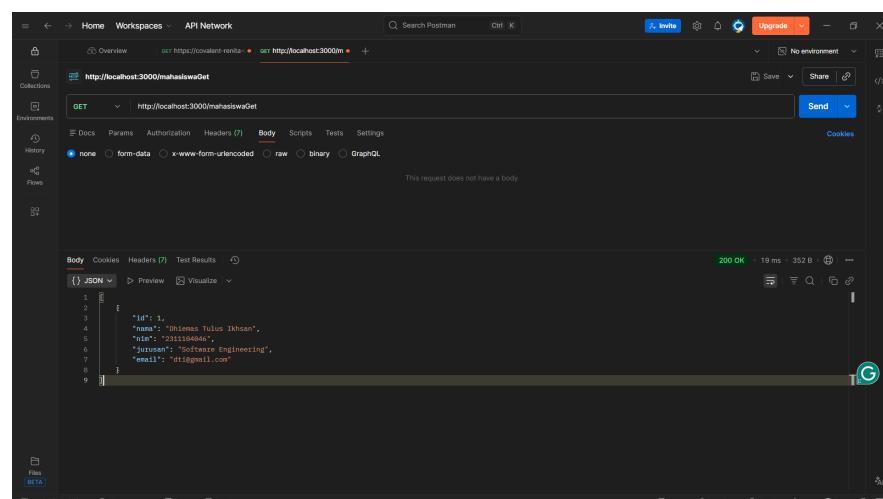


POST <http://localhost:3000/mahasiswaCreate>

```
{  
  "id": 1,  
  "name": "Dhieness Tulus Ichsan",  
  "nisn": "2311061994061",  
  "jurusan": "Software Engineering",  
  "email": "dt@gmail.com"  
}
```

201 Created · 137 ms · 250 B

API GET



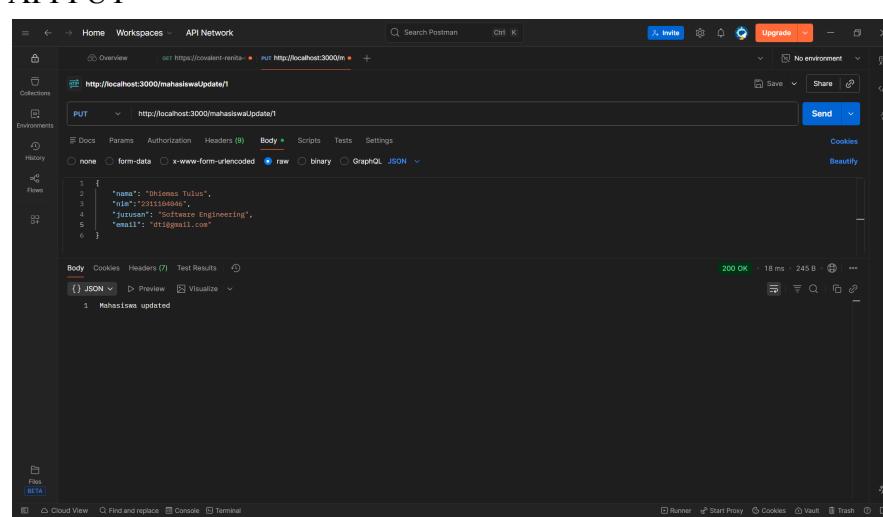
GET <http://localhost:3000/mahasiswaGet>

```
[{}]
```

200 OK · 19 ms · 352 B

```
{  
  "id": 1,  
  "name": "Dhieness Tulus Ichsan",  
  "nisn": "2311061994061",  
  "jurusan": "Software Engineering",  
  "email": "dt@gmail.com"  
}
```

API PUT

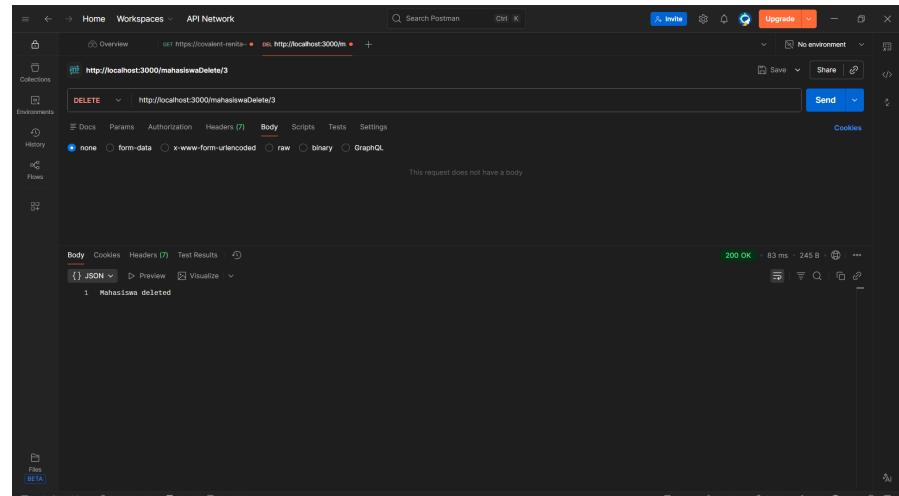


PUT <http://localhost:3000/mahasiswaUpdate/1>

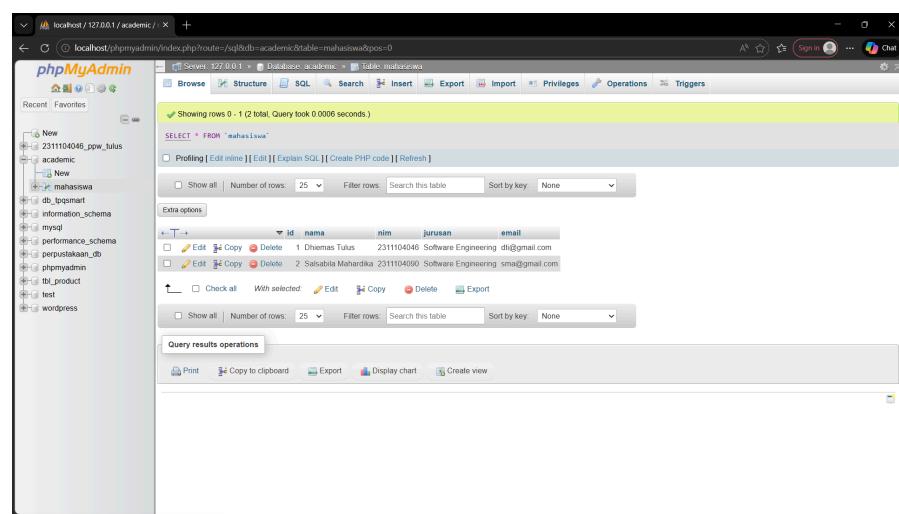
```
{  
  "id": 1,  
  "name": "Dhieness Tulus",  
  "nisn": "2311061994061",  
  "jurusan": "Software Engineering",  
  "email": "dt@gmail.com"  
}
```

200 OK · 18 ms · 245 B

API DELETE



The screenshot shows the Postman application interface. A DELETE request is made to `http://localhost:3000/mahasiswaDelete/3`. The response status is `200 OK` with a response body containing the message `Mahasiswa deleted`.



The screenshot shows the phpMyAdmin interface connected to the `academic` database. The `mahasiswa` table is selected. The table structure includes columns `id`, `nama`, `nim`, `jurusan`, and `email`. Two rows are present in the table:

	id	nama	nim	jurusan	email
1	1	Dhemas Tulus	231104046	Software Engineering	djh@gmail.com
2	2	Salisabila Mahardika	231104090	Software Engineering	sma@gmail.com

Penjelasan:

Praktikum ini mendemonstrasikan pembuatan server web menggunakan modul `http` yang merupakan bawaan dari `Node.js`. Server dikonfigurasikan pada host `127.0.0.1` dengan port `3000` untuk mendengarkan setiap permintaan yang masuk dari sisi klien atau browser. Secara teknis, `Node.js` akan memproses request tersebut melalui fungsi callback yang dirancang untuk memberikan respons tertentu secara spesifik.

Dalam prosesnya, server akan memberikan tanda bahwa permintaan berhasil melalui status kode HTTP `200` serta menetapkan header `Content-Type` sebagai `text/plain`. Hal ini bertujuan agar browser mengenali bahwa data yang diterima adalah teks biasa, yang dalam kasus ini adalah pesan "Hello World". Mekanisme ini bekerja secara asynchronous, yang memungkinkan server tetap responsif dalam menangani berbagai permintaan secara bersamaan tanpa adanya hambatan proses (blocking).

Keberhasilan jalannya server ditandai dengan munculnya notifikasi pada terminal yang menginformasikan bahwa server telah aktif. Mahasiswa kemudian dapat melakukan verifikasi dengan mengakses alamat tersebut melalui browser untuk melihat hasil keluarannya. Melalui latihan ini, konsep dasar mengenai siklus hidup server, mulai dari inisialisasi, penanganan permintaan, hingga pengiriman data balik ke pengguna, dapat dipahami dengan lebih mendalam.

2.2 UNGUIDED (Tugas Mandiri)

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:

Source Code:

- db.js

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

- crud.js

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

function getAllMahasiswa(callback) {
    db.query("SELECT * FROM mahasiswa", callback);
}

function createMahasiswa(data, callback) {
    const sql = "INSERT INTO mahasiswa SET ?";
    db.query(sql, data, callback);
}
```

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

- app.js

```
const express = require("express");
const path = require("path"); // Tambahkan ini
console.log("APP.JS DIJALANKAN");
const dbOperations = require("./crud");

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

app.use(express.json());

// --- ROUTE UNTUK HALAMAN WEB ---
app.get("/", (req, res) => {
    // Tambahkan 'public' di dalam path.join
    res.sendFile(path.join(__dirname, "public",
    "index.html"));
});

// --- ENDPOINT API (Disamakan dengan fetch di HTML)
---

// Ambil Data
app.get("/api/mahasiswa", (req, res) => {
    dbOperations.getAllMahasiswa((error, results) => {
        if (error) return res.status(500).send("Error
fetching");
        res.json(results);
    });
});

// Tambah Data
app.post("/api/mahasiswa", (req, res) => {
    const data = req.body;
    dbOperations.createMahasiswa(data, (error) => {
        if (error) return res.status(500).send("Error
creating");
        res.status(201).send("Mahasiswa created");
    });
});

// Update Data
app.put("/api/mahasiswa/:id", (req, res) => {
```

```

        const { id } = req.params;
        const { nama, nim, jurusan, email } = req.body;
        dbOperations.updateMahasiswa(id, nama, nim, jurusan,
email, (error) => {
            if (error) return res.status(500).send("Error
updating");
            res.send("Mahasiswa updated");
        });
    });

// Hapus Data
app.delete("/api/mahasiswa/:id", (req, res) => {
    const { id } = req.params;
    dbOperations.deleteMahasiswa(id, (error) => {
        if (error) return res.status(500).send("Error
deleting");
        res.send("Mahasiswa deleted");
    });
});

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

```

- public/index.htm

```

<!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>

    <!-- MODAL EDIT -->
```

```
<div class="modal fade" id="editModal"
tabindex="-1">
    <div class="modal-dialog">
        <div class="modal-content">

            <div class="modal-header">
                <h5 class="modal-title">Edit Data
Mahasiswa</h5>
                <button type="button" class="btn-close"
data-bs-dismiss="modal"></button>
            </div>

            <div class="modal-body">
                <input type="hidden" id="editId">

                    <input id="editNama" class="form-control mb-2"
placeholder="Nama">
                    <input id="editNim" class="form-control mb-2"
placeholder="NIM">
                    <input id="editJurusan" class="form-control
mb-2" placeholder="Jurusan">
                    <input id="editEmail" class="form-control
mb-2" placeholder="Email">
            </div>

            <div class="modal-footer">
                <button class="btn btn-secondary"
data-bs-dismiss="modal">Batal</button>
                <button class="btn btn-primary"
onclick="updateData()">Simpan Perubahan</button>
            </div>

        </div>
    </div>
</div>

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

<script>
// let editModal;
```

```
// loadData();

// 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()
        );
    }
}

// ===== EDIT
function edit(id) {
    fetch("/api/mahasiswa")
        .then((res) => res.json())
        .then((data) => {
            const m = data.find(x => x.id === id);

            editId.value = m.id;
            editNama.value = m.nama;
            editNim.value = m.nim;
            editJurusan.value = m.jurusan;
            editEmail.value = m.email;

            editModal = new bootstrap.Modal(
                document.getElementById("editModal")
            );
            editModal.show();
        });
}

// 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>

```

1

Output

Aplikasi Data Mahasiswa

Tambah Mahasiswa

Nama
NIM
Jurusan
Email

Tambah Data

Aplikasi Data Mahasiswa

Tambah Mahasiswa

Nama
NIM
Jurusan
Email

Tambah Data

Daftar Mahasiswa

Dhiemas Tulus Ikhsan
NIM: 2311104046
Jurusan: Software Engineering
Email: dti@gmail.com

Hapus

The top screenshot shows a modal dialog titled "Tambah Mahasiswa" (Add Student) with fields for Name, NIM, Jurusan (Major), and Email. A confirmation message from "localhost:3000 says" asks if the user wants to delete the data. The bottom screenshot shows the "Daftar Mahasiswa" (Student List) table in phpMyAdmin, containing one row of data:

	id	nama	nim	jurusan	email
	1	Dhiemas Tulus Iksan	2311104046	Software Engineering	dti@gmail.com

Penjelasan:

Proyek pada tugas unguided ini menghasilkan sebuah platform manajemen data mahasiswa yang mengintegrasikan Node.js, Express.js, dan MySQL melalui arsitektur RESTful API. Dalam ekosistem ini, Node.js berfungsi sebagai lingkungan eksekusi kode di sisi server, sementara Express.js mengelola alur routing serta komunikasi protokol HTTP secara efisien. Seluruh informasi mahasiswa disimpan secara terpusat di dalam basis data MySQL untuk menjamin persistensi data.

Sesuai dengan prinsip CRUD, aplikasi ini mengandalkan empat endpoint fundamental: GET untuk menarik daftar mahasiswa, POST untuk merekam entri baru, PUT untuk memodifikasi data yang sudah ada berdasarkan identitas unik (ID), serta DELETE untuk mengeliminasi rekaman tertentu. Setiap operasi tersebut diterjemahkan ke dalam instruksi query SQL yang dieksekusi langsung terhadap database melalui modul koneksi MySQL.

Untuk bagian antarmuka, aplikasi memanfaatkan kombinasi HTML, JavaScript, dan Bootstrap guna menciptakan tampilan yang responsif dan user-friendly. Interaksi data dilakukan secara asynchronous menggunakan metode `fetch()` untuk menjembatani komunikasi antara sisi klien dan server tanpa perlu memuat ulang halaman. Fitur pembaruan data dikemas secara praktis melalui modal form yang memungkinkan perubahan informasi secara menyeluruh, sementara aksi penghapusan dipicu oleh tombol spesifik yang langsung terhubung ke endpoint `DELETE`. Sinergi ini menciptakan sebuah sistem pengelolaan data mahasiswa yang dinamis, teratur, dan terintegrasi secara utuh antara frontend dan backend.

BAB III

PENUTUP

KESIMPULAN

Berdasarkan rangkaian praktikum dan pengerjaan tugas mandiri, dapat disimpulkan bahwa kombinasi Node.js dan Express.js memberikan kemudahan dalam membangun aplikasi server-side yang efisien menggunakan JavaScript. Implementasi arsitektur RESTful API melalui metode GET, POST, PUT, dan DELETE terbukti efektif dalam menyusun sistem manajemen data mahasiswa yang sistematis sesuai dengan prinsip CRUD. Di samping itu, penggunaan MySQL sebagai basis data memastikan seluruh informasi dapat tersimpan dengan aman dan bersifat persisten.

Dalam tahap pengembangan, muncul beberapa hambatan teknis seperti kekeliruan konfigurasi server, adanya mismatch antara parameter pada endpoint dengan primary key di database, serta kendala pada logika pembaruan dan penghapusan data. Masalah-masalah tersebut berhasil diselesaikan melalui pemeriksaan mendalam terhadap struktur tabel, memastikan konsistensi penggunaan ID sebagai kunci utama, serta melakukan debugging intensif pada rute Express dan instruksi query SQL. Melalui kegiatan ini, pemahaman mengenai sinkronisasi antara antarmuka pengguna (frontend) dan pengolahan data (backend) menjadi lebih kuat, sekaligus memperjelas alur kerja aplikasi web secara menyeluruh.