

**LAPORAN PRAKTIKUM
PERANCANGAN DAN PEMROGRAMAN WEB**

**MODUL 13
NODE JS**



**Universitas
Telkom**

Oleh:

Tiurma Grace Angelina 2311104042

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

BAB I

PENDAHULUAN

A. Dasar Teori

Node.js merupakan platform runtime JavaScript yang digunakan untuk membangun aplikasi server-side dengan model asynchronous dan event-driven, sehingga mampu menangani banyak request secara efisien. Dalam praktikum ini, Node.js dipadukan dengan framework Express.js yang berfungsi untuk mempermudah pembuatan server dan pengelolaan routing RESTful API. RESTful API sendiri merupakan arsitektur layanan web yang memungkinkan komunikasi antara client dan server melalui protokol HTTP dengan metode seperti GET, POST, PUT, dan DELETE.

Selain itu, MySQL digunakan sebagai sistem manajemen basis data relasional untuk menyimpan dan mengelola data mahasiswa. Integrasi antara Node.js, Express.js, dan MySQL memungkinkan terbentuknya aplikasi web yang terstruktur, dimana frontend berperan sebagai client dan backend bertanggung jawab dalam pengolahan data serta komunikasi dengan database.

B. Tujuan

Tujuan dari praktikum ini adalah untuk memahami konsep dasar pengembangan aplikasi web berbasis Node.js, khususnya dalam pembuatan RESTful API dan implementasi operasi CRUD (Create, Read, Update, Delete). Selain itu, praktikum ini bertujuan untuk melatih mahasiswa dalam mengintegrasikan backend dengan database MySQL serta menghubungkannya dengan antarmuka web sederhana sebagai implementasi aplikasi client-server.

BAB II

HASIL

A. GUIDED (Praktikum Terbimbing)

- a. Inisialisasi Proyek Node.js

Langkah-langkah:

1. Membuat folder project
 2. Masuk ke folder project
 3. Inisialisasi Node.js

Perintah yang dijalankan di terminal:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\USER> cd C:\xampp\htdocs
● PS C:\xampp\htdocs> mkdir mahasiswa-app

Directory: C:\xampp\htdocs

Mode                LastWriteTime          Length Name
----              - - - - - - - - - - - - - - - - - - -
d----- 12/30/2025 6:44 PM               mahasiswa-app

● PS C:\xampp\htdocs> cd mahasiswa-app
○ PS C:\xampp\htdocs\mahasiswa-app>
```

The screenshot shows a terminal window in a code editor interface. The command line is PS C:\xampp\htdocs\mahasiswa-app> and the user has run npm init -y to generate a package.json file. The content of the package.json file is displayed as follows:

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

At the bottom of the terminal, there is a prompt: % PS C:\xampp\htdocs\mahasiswa-app>

Penjelasan:

Perintah `npm init -y` digunakan untuk membuat file `package.json` secara otomatis sebagai konfigurasi awal proyek Node.js.

- Terminal setelah npm init -y
 - Isi file package.json
- b. Instalasi Dependency (Express & MySQL)

Langkah-langkah:

Menginstal framework Express.js dan library MySQL.

```

MAHASISWA-APP
> node_modules
package-lock.json
package.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

●          npm install express mysql
added 75 packages, and audited 76 packages in 5s
22 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
PS C:\xampp\htdocs\mahasiswa-app>

```

Penjelasan:

- **Express.js** digunakan untuk membuat server dan RESTful API
 - **MySQL** digunakan untuk koneksi database MySQL dari Node.js
- c. Konfigurasi Database MySQL
1. membuat database & tabel

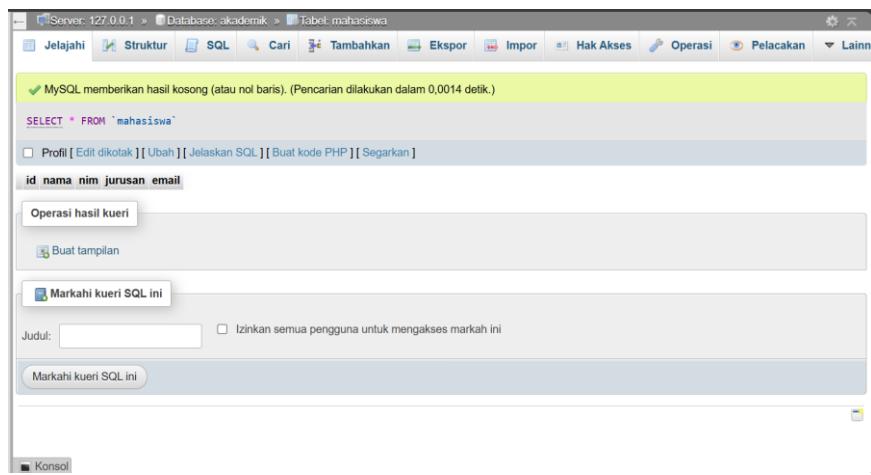
Query SQL yang dijalankan di phpMyAdmin:

```

CREATE DATABASE akademik;
USE akademik;

CREATE TABLE mahasiswa (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nama VARCHAR(100),
  nim VARCHAR(20),
  jurusan VARCHAR(50),
  email VARCHAR(100)
);

```



Penjelasan:

Database akademik digunakan untuk menyimpan data mahasiswa yang akan diakses melalui RESTful API.

- Database akademik
- Struktur tabel mahasiswa
- Minimal 5 data mahasiswa

2. Membuat File Koneksi Database (db.js)

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'akademik'
});

connection.connect(err => {
  if (err) {
    console.error('Koneksi database gagal:', err);
    return;
  }
  console.log('Database connected');
});

module.exports = connection;
```

Penjelasan:

File db.js digunakan untuk mengatur koneksi antara aplikasi Node.js dengan database MySQL.

- Kode db.js

- Terminal muncul tulisan **Database connected**

d. Implementasi CRUD (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) {
  connection.query("SELECT * FROM mahasiswa", 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) {
  connection.query("DELETE FROM mahasiswa WHERE id=?", [id], callback);
}

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

Penjelasan:

File crud.js berisi fungsi CRUD yang berinteraksi langsung dengan database MySQL.

- Kode crud.js

e. Pembuatan Server & Routing (app.js)

```
const express = require("express");
```

```

const crud = 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;
  crud.createMahasiswa(nama, nim, jurusan, email, () => {
    res.send("Mahasiswa created");
  });
});

// READ
app.get("/mahasiswaGet", (req, res) => {
  crud.getAllMahasiswa((err, results) => {
    res.json(results);
  });
});

// UPDATE
app.put("/mahasiswaUpdate/:id", (req, res) => {
  const { id } = req.params;
  const { nama, nim, jurusan, email } = req.body;
  crud.updateMahasiswa(id, nama, nim, jurusan, email, () => {
    res.send("Mahasiswa updated");
  });
});

// DELETE
app.delete("/mahasiswaDelete/:id", (req, res) => {
  crud.deleteMahasiswa(req.params.id, () => {
    res.send("Mahasiswa deleted");
  });
});

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

```

Penjelasan:

File app.js merupakan pusat aplikasi yang mengatur routing RESTful API menggunakan Express.js.

- Kode app.js
- Terminal: Server running on http://localhost:3000

f. Pengujian API Menggunakan Postman

1. POST (Create)

- Method: POST
- URL: http://localhost:3000/mahasiswaCreate
- Body → raw → JSON

The screenshot shows the Postman interface with a successful POST request. The URL is set to `http://localhost:3000/mahasiswaCreate`. The request method is `POST`. The `Body` tab is selected, showing the following JSON payload:

```

1  {
2   "nama": "Andi",
3   "nim": "2311104001",
4   "jurusan": "RPL",
5   "email": "andi@email.com"
6 }
7

```

The response section shows a green `201 Created` status bar with metrics: 302 ms, 250 B. The response body is displayed as `HTML` and contains the message: `1 Mahasiswa created`.

2. GET (Read)

- Method: GET
- URL: http://localhost:3000/mahasiswaGet

HTTP Get Request

GET http://localhost:3000/mahasiswaGet

200 OK • 51 ms • 319 B

Key	Value	Description
Key	Value	Description

```
[{"id": 1, "nama": "Andi", "nim": "2311184001", "jurusan": "RPL", "email": "andi@email.com"}]
```

3. PUT (Update)

- Method: PUT
- URL: http://localhost:3000/mahasiswaUpdate/1

HTTP Update Request

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

200 OK

```
{"nama": "Budi Update", "nim": "2311104001", "jurusan": "RPL", "email": "budiupdate@email.com"}
```

Mahasiswa updated

4. DELETE

- Method: DELETE
- URL: http://localhost:3000/mahasiswaDelete/1

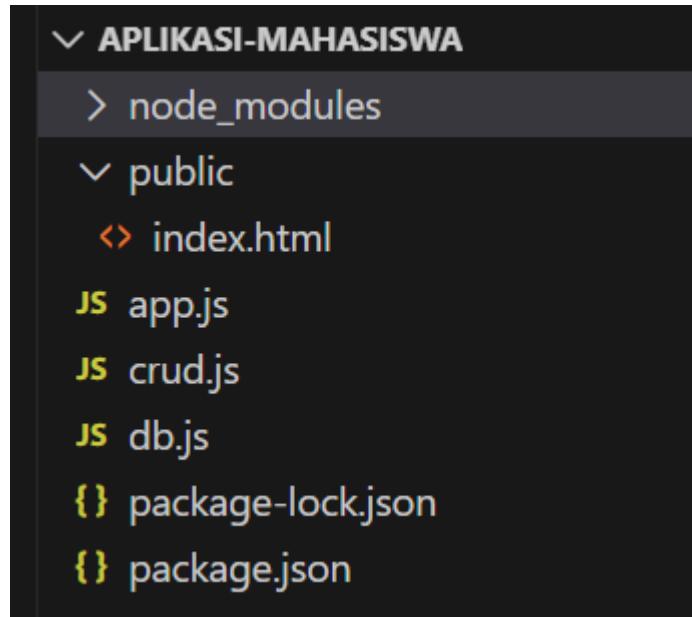
B. UNGUIDED (Tugas Mandiri)

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:

Struktur Folder Aplikasi



Penjelasan singkat:

- index.html → antarmuka (frontend) pengelolaan data mahasiswa
- app.js → server Express dan routing API
- crud.js → logika CRUD database
- db.js → koneksi database MySQL

Source Code:

Database

The screenshot shows a MySQL database interface. On the left, there's a tree view of databases and tables:

- Baru
- akademik
- Baru
- mahasiswa
- db_ptgrace
- information_schema
- mahasiswa_db

In the main pane, there's a SQL query editor with the following code:

```
SELECT * FROM `mahasiswa`
```

Below the query, there are several buttons: Profil [Edit dikotak], [Ubah], [Jelaskan SQL], [Buat kode PHP], and [Segarkan].

At the bottom, there's a table with columns id, nama, nim, jurusan, and email. The table has one row of data: 1, 'Andrea', '1234567890', 'Teknik Informatika', 'andrea@ptgrace.ac.id'.

At the very bottom, there's a button labeled Operasi hasil kueri.

C:\xampp\htdocs\PERTEMUAN 13\aplikasi-mahasiswa\public\index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pengelolaan Data Mahasiswa</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #373739 0%, #c9bdd4 100%);
            min-height: 100vh;
            padding: 20px;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            background: white;
            border-radius: 15px;
            box-shadow: 0 10px 40px rgba(0,0,0,0.2);
            overflow: hidden;
        }

        header {
            background: linear-gradient(135deg, #e89ce4 0%, #414141 100%);
            color: white;
            padding: 30px;
            text-align: center;
        }

        header h1 {
            font-size: 2.5em;
            margin-bottom: 10px;
        }

        header p {
            opacity: 0.9;
        }
    </style>

```

```
}

.content {
    padding: 30px;
}

.form-section {
    background: #f8f9fa;
    padding: 25px;
    border-radius: 10px;
    margin-bottom: 30px;
}

.form-section h2 {
    color: #c53260;
    margin-bottom: 20px;
    font-size: 1.5em;
}

.form-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 15px;
    margin-bottom: 20px;
}

.form-group {
    display: flex;
    flex-direction: column;
}

label {
    font-weight: 600;
    margin-bottom: 5px;
    color: #333;
}

input {
    padding: 12px;
    border: 2px solid #e0e0e0;
    border-radius: 8px;
    font-size: 14px;
    transition: all 0.3s;
}
```

```
input:focus {  
    outline: none;  
    border-color: #7a0d50;  
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);  
}  
  
.btn-group {  
    display: flex;  
    gap: 10px;  
    flex-wrap: wrap;  
}  
  
button {  
    padding: 12px 30px;  
    border: none;  
    border-radius: 8px;  
    font-size: 14px;  
    font-weight: 600;  
    cursor: pointer;  
    transition: all 0.3s;  
}  
  
.btn-primary {  
    background: linear-gradient(135deg, #ea66c5 0%, #d2c8dc 100%);  
    color: white;  
}  
  
.btn-primary:hover {  
    transform: translateY(-2px);  
    box-shadow: 0 5px 15px rgb(98, 61, 99);  
}  
  
.btn-secondary {  
    background: #6c757d;  
    color: white;  
}  
  
.btn-secondary:hover {  
    background: #5a6268;  
}  
  
.btn-success {  
    background: #340f2dca;
```

```
color: white;
padding: 8px 15px;
font-size: 12px;
}

.btn-success:hover {
    background: #d76196;
}

.btn-danger {
    background: #dc3545;
    color: white;
    padding: 8px 15px;
    font-size: 12px;
}

.btn-danger:hover {
    background: #cd4b95;
}

.table-section {
    overflow-x: auto;
}

table {
    width: 100%;
    border-collapse: collapse;
    background: white;
    border-radius: 10px;
    overflow: hidden;
}

thead {
    background: linear-gradient(135deg, #261727 0%, #ff015a57 100%);
    color: white;
}

th, td {
    padding: 15px;
    text-align: left;
}

tbody tr {
    border-bottom: 1px solid #e0e0e0;
```

```
        transition: background 0.3s;
    }

tbody tr:hover {
    background: #f8f9fa;
}

.action-buttons {
    display: flex;
    gap: 5px;
}

.alert {
    padding: 15px;
    border-radius: 8px;
    margin-bottom: 20px;
    display: none;
}

.alert-success {
    background: #d4edda;
    color: #155724;
    border: 1px solid #c3e6cb;
}

.alert-error {
    background: #f8d7da;
    color: #b74f8fe5;
    border: 1px solid #f5c6cb;
}

.loading {
    text-align: center;
    padding: 20px;
    color: #d47ab2f5;
}

@media (max-width: 768px) {
    header h1 {
        font-size: 1.8em;
    }
}

.form-grid {
    grid-template-columns: 1fr;
```

```

        }

    .btn-group {
        flex-direction: column;
    }

    button {
        width: 100%;
    }

}
</style>
</head>
<body>
    <div class="container">
        <header>
            <h1>SISTEM PENGELOLAAN DATA MAHASISWA</h1>
            <p>Aplikasi berbasis Node.js, Express.js, dan MySQL</p>
        </header>

        <div class="content">
            <!-- Alert Messages -->
            <div id="alertSuccess" class="alert alert-success"></div>
            <div id="alertError" class="alert alert-error"></div>

            <!-- Form Input -->
            <div class="form-section">
                <h2 id="formTitle">Tambah Data Mahasiswa</h2>
                <form id="mahasiswaForm">
                    <input type="hidden" id="mahasiswaId">
                    <div class="form-grid">
                        <div class="form-group">
                            <label for="nama">Nama Lengkap *</label>
                            <input type="text" id="nama" placeholder="Masukkan
nama lengkap" required>
                        </div>
                        <div class="form-group">
                            <label for="nim">NIM *</label>
                            <input type="text" id="nim" placeholder="Masukkan NIM"
required>
                        </div>
                        <div class="form-group">
                            <label for="jurusan">Jurusan *</label>
                            <input type="text" id="jurusan" placeholder="Masukkan
jurusan" required>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>

```

```
</div>
<div class="form-group">
    <label for="email">Email *</label>
    <input type="email" id="email" placeholder="Masukkan
email" required>
    </div>
</div>
<div class="btn-group">
    <button type="submit" class="btn-primary"
id="submitBtn">Tambah Mahasiswa</button>
    <button type="button" class="btn-secondary" id="cancelBtn"
style="display:none;">Batal</button>
    </div>
</form>
</div>

<!-- Tabel Data -->
<div class="table-section">
    <h2 style="color: #c9006e; margin-bottom: 20px;"> Daftar
Mahasiswa</h2>
    <div id="loading" class="loading" style="display:none;">Loading
data...</div>
    <table id="mahasiswaTable">
        <thead>
            <tr>
                <th>No</th>
                <th>Nama</th>
                <th>NIM</th>
                <th>Jurusan</th>
                <th>Email</th>
                <th>Aksi</th>
            </tr>
        </thead>
        <tbody id="mahasiswaTableBody">
            <!-- Data akan dimuat dari API -->
        </tbody>
    </table>
</div>
</div>
</div>

<script>
// Variabel global
let isEditMode = false;
```

```

// Fungsi untuk menampilkan alert
function showAlert(message, type) {
    const alertSuccess = document.getElementById('alertSuccess');
    const alertError = document.getElementById('alertError');

    if (type === 'success') {
        alertSuccess.textContent = message;
        alertSuccess.style.display = 'block';
        alertError.style.display = 'none';
    } else {
        alertError.textContent = message;
        alertError.style.display = 'block';
        alertSuccess.style.display = 'none';
    }

    // Hide alert after 3 seconds
    setTimeout(() => {
        alertSuccess.style.display = 'none';
        alertError.style.display = 'none';
    }, 3000);
}

// Fungsi untuk load data mahasiswa
async function loadMahasiswa() {
    document.getElementById('loading').style.display = 'block';

    try {
        const response = await fetch('/mahasiswaGet');
        const data = await response.json();

        const tbody = document.getElementById('mahasiswaTableBody');
        tbody.innerHTML = "";

        if (data.length === 0) {
            tbody.innerHTML = '<tr><td colspan="6" style="text-align:center;">Tidak ada data</td></tr>';
        } else {
            data.forEach((mhs, index) => {
                const row = `
                    <tr>
                    <td>${index + 1}</td>
                    <td>${mhs.nama}</td>
                    <td>${mhs.nim}</td>
                `;
            });
        }
    } catch (error) {
        console.error(error);
    }
}

```

```

        <td>${mhs.jurusan}</td>
        <td>${mhs.email}</td>
        <td class="action-buttons">
            <button class="btn-success"
                onclick="editMahasiswa(${mhs.id}, '${mhs.nama}', '${mhs.nim}',
                '${mhs.jurusan}', '${mhs.email}')">Edit</button>
            <button class="btn-danger"
                onclick="deleteMahasiswa(${mhs.id})">Hapus</button>
        </td>
    </tr>
    ';
    tbody.innerHTML += row;
}
}

} catch (error) {
    showAlert('Gagal memuat data: ' + error.message, 'error');
} finally {
    document.getElementById('loading').style.display = 'none';
}
}

// Fungsi untuk submit form (tambah/update)
document.getElementById('mahasiswaForm').addEventListener('submit',
async (e) => {
    e.preventDefault();

    const id = document.getElementById('mahasiswaId').value;
    const nama = document.getElementById('nama').value;
    const nim = document.getElementById('nim').value;
    const jurusan = document.getElementById('jurusan').value;
    const email = document.getElementById('email').value;

    const data = { nama, nim, jurusan, email };

    try {
        let response;
        if (isEditMode) {
            // Update
            response = await fetch('/mahasiswaUpdate/${id}', {
                method: 'PUT',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(data)
            });
        } else {
    
```

```

// Create
response = await fetch('/mahasiswaCreate', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(data)
});
}

const result = await response.json();

if (response.ok) {
  showAlert(isEditMode ? 'Data berhasil diupdate!' : 'Data berhasil
ditambahkan!', 'success');
  resetForm();
  loadMahasiswa();
} else {
  showAlert('Gagal menyimpan data', 'error');
}
} catch (error) {
  showAlert('Error: ' + error.message, 'error');
}
});

// Fungsi untuk edit mahasiswa
function editMahasiswa(id, nama, nim, jurusan, email) {
  isEditMode = true;
  document.getElementById('mahasiswaId').value = id;
  document.getElementById('nama').value = nama;
  document.getElementById('nim').value = nim;
  document.getElementById('jurusan').value = jurusan;
  document.getElementById('email').value = email;

  document.getElementById('formTitle').textContent = '📝 Edit Data
Mahasiswa';
  document.getElementById('submitBtn').textContent = 'Update
Mahasiswa';
  document.getElementById('cancelBtn').style.display = 'inline-block';

  // Scroll ke form
  document.getElementById('mahasiswaForm').scrollIntoView({
behavior: 'smooth' });
}

// Fungsi untuk delete mahasiswa

```

```

async function deleteMahasiswa(id) {
    if (!confirm('Apakah Anda yakin ingin menghapus data ini?')) return;

    try {
        const response = await fetch(`/mahasiswaDelete/${id}`, {
            method: 'DELETE'
        });

        const result = await response.json();

        if (response.ok) {
            showAlert('Data berhasil dihapus!', 'success');
            loadMahasiswa();
        } else {
            showAlert('Gagal menghapus data', 'error');
        }
    } catch (error) {
        showAlert('Error: ' + error.message, 'error');
    }
}

// Fungsi untuk reset form
function resetForm() {
    isEditMode = false;
    document.getElementById('mahasiswaForm').reset();
    document.getElementById('mahasiswaId').value = "";
    document.getElementById('formTitle').textContent = 'Tambah Data Mahasiswa';
    document.getElementById('submitBtn').textContent = 'Tambah Mahasiswa';
    document.getElementById('cancelBtn').style.display = 'none';
}

// Event listener untuk tombol cancel
document.getElementById('cancelBtn').addEventListener('click', resetForm);

// Load data saat halaman pertama kali dibuka
window.addEventListener('DOMContentLoaded', loadMahasiswa);
</script>
</body>
</html>

```

C:\xampp\htdocs\PERTEMUAN 13\aplikasi-mahasiswa\app.js

```
const express = require("express");
const dbOperations = require("./crud");
const path = require("path");
const app = express();
const port = 3000;

app.use(express.json());

// TAMBAHAN INI BIAR BISA AKSES FILE HTML/CSS/JS
app.use(express.static('public'));

// Endpoint untuk halaman utama
app.get("/", (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

// API Endpoints
app.post("/mahasiswaCreate", (req, res) => {
  const { nama, nim, jurusan, email } = req.body;
  dbOperations.createMahasiswa(nama, nim, jurusan, email, (error) => {
    if (error) {
      return res.status(500).json({ message: "Error creating" });
    }
    res.status(201).json({ message: "Mahasiswa created" });
  });
});

app.get("/mahasiswaGet", (req, res) => {
  dbOperations.getAllMahasiswa((error, mahasiswa) => {
    if (error) {
      return res.status(500).json({ message: "Error fetching data" });
    }
    res.json(mahasiswa);
  });
});

app.put("/mahasiswaUpdate/: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).json({ message: "Error updating" });
    }
    res.json({ message: "Mahasiswa updated" });
});
```

```

    });
});

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

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

```

C:\xampp\htdocs\PERTEMUAN 13\aplikasi-mahasiswa\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,
};

```

C:\xampp\htdocs\PERTEMUAN 13\aplikasi-mahasiswa\db.js

```

const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'akademik'
});

connection.connect(err => {
  if (err) {

```

```
        console.error('Koneksi database gagal:', err);
        return;
    }
    console.log('Database connected');
});

module.exports = connection;
```

C:\xampp\htdocs\PERTEMUAN 13\aplikasi-mahasiswa\package.json

```
{
  "name": "aplikasi-mahasiswa",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "dependencies": {
    "express": "^5.2.1",
    "mysql": "^2.18.1"
  }
}
```

Output:

SISTEM PENGELOLAAN DATA MAHASISWA

Aplikasi berbasis Node.js, Express.js, dan MySQL

Tambah Data Mahasiswa

Nama Lengkap *
Masukkan nama lengkap

NIM *
Masukkan NIM

Jurusan *
Masukkan jurusan

Email *
Masukkan email

Tambah Mahasiswa

Daftar Mahasiswa

No	Nama	NIM	Jurusan	Email	Aksi
Tidak ada data					

SISTEM PENGELOLAAN DATA MAHASISWA

Aplikasi berbasis Node.js, Express.js, dan MySQL

Tambah Data Mahasiswa

Nama Lengkap *
Masukkan nama lengkap

NIM *
Masukkan NIM

Jurusan *
Masukkan jurusan

Email *
Masukkan email

Tambah Mahasiswa

Daftar Mahasiswa

No	Nama	NIM	Jurusan	Email	Aksi
1	tiurma grace	2311104042	RPL	2311104042@ittelkom-pwt.ac.id	<button>Edit</button> <button>Hapus</button>

SISTEM PENGELOLAAN DATA MAHASISWA

Aplikasi berbasis Node.js, Express.js, dan MySQL

Data berhasil diupdate!

Tambah Data Mahasiswa

Nama Lengkap *	NIM *	Jurusan *	Email *
<input type="text" value="Masukkan nama lengkap"/>	<input type="text" value="Masukkan NIM"/>	<input type="text" value="Masukkan jurusan"/>	<input type="text" value="Masukkan email"/>

[Tambah Mahasiswa](#)

Daftar Mahasiswa

No	Nama	NIM	Jurusan	Email	Aksi
1	turma grace angelina sihaloho	2311104042	RPL	2311104042@ittelkom-pwt.ac.id	Edit Hapus

localhost:3000 says
Apakah Anda yakin ingin menghapus data ini?

[OK](#) [Cancel](#)

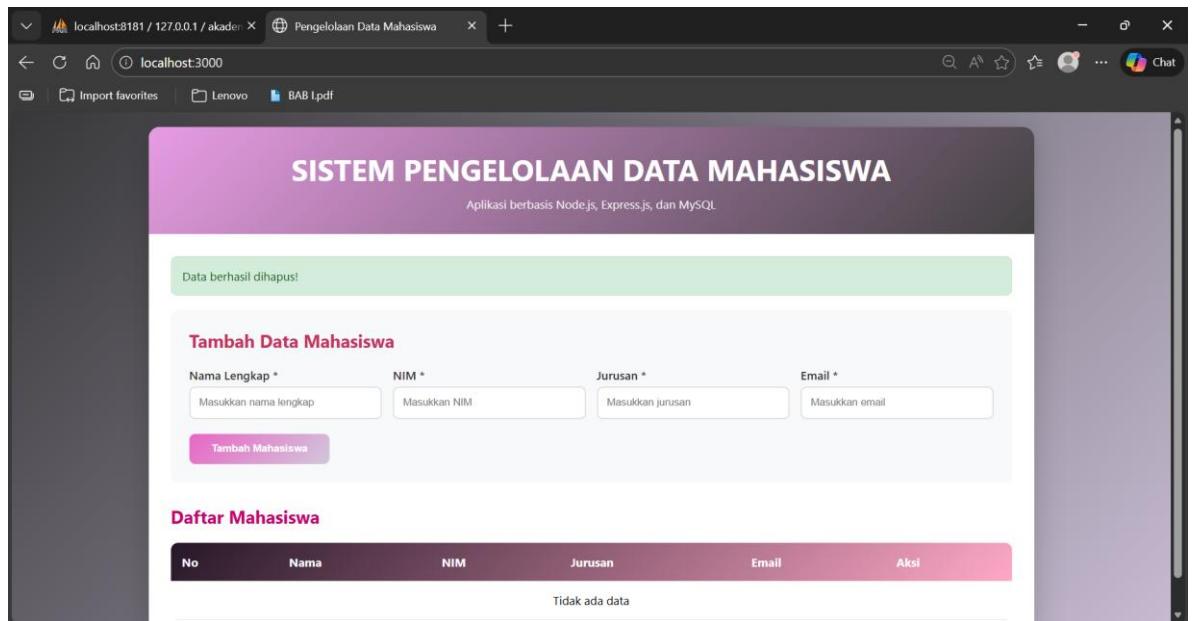
Tambah Data Mahasiswa

Nama Lengkap *	NIM *	Jurusan *	Email *
<input type="text" value="Masukkan nama lengkap"/>	<input type="text" value="Masukkan NIM"/>	<input type="text" value="Masukkan jurusan"/>	<input type="text" value="Masukkan email"/>

[Tambah Mahasiswa](#)

Daftar Mahasiswa

No	Nama	NIM	Jurusan	Email	Aksi
1	turma grace angelina sihaloho	2311104042	RPL	2311104042@ittelkom-pwt.ac.id	Edit Hapus



Penjelasan:

File index.html berfungsi sebagai antarmuka pengguna untuk mengelola data mahasiswa. Halaman ini menyediakan form input data, tabel daftar mahasiswa, serta tombol edit dan hapus. Komunikasi dengan backend dilakukan menggunakan Fetch API yang terhubung ke RESTful API Node.js.

File app.js berfungsi sebagai server utama yang mengatur routing API serta menghubungkan frontend dengan backend. Middleware express.static() digunakan agar file HTML dapat diakses melalui browser.

File crud.js berisi fungsi Create, Read, Update, dan Delete yang digunakan untuk mengelola data mahasiswa pada database MySQL.

File db.js digunakan untuk menghubungkan aplikasi Node.js dengan database MySQL bernama akademik.

Pengguna dapat mengakses aplikasi melalui browser. Data mahasiswa dimasukkan melalui form yang tersedia dan dikirim ke server menggunakan Fetch API. Server Node.js memproses data tersebut melalui RESTful API dan menyimpannya ke database MySQL. Data yang tersimpan kemudian ditampilkan kembali ke halaman web dalam bentuk tabel secara real-time.

BAB III

PENUTUP

A. Kesimpulan

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa mahasiswa berhasil memahami dan mengimplementasikan pembuatan aplikasi pengelolaan data mahasiswa menggunakan Node.js, Express.js, dan MySQL. Pada praktikum ini dipelajari konsep RESTful API, operasi CRUD, serta pengujian API menggunakan Postman. Pada tugas mandiri, aplikasi dikembangkan dengan menambahkan antarmuka berbasis web sehingga dapat diakses melalui browser dan terhubung dengan backend menggunakan Fetch API. Kendala yang dihadapi antara lain kesalahan pengiriman data JSON dan konfigurasi routing, namun dapat diatasi dengan pengaturan request yang benar dan penyesuaian kode pada server.