LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 14

(Node JS)

Oleh:

(Haikal Fadhilah Mufid) (2311104027)

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Node.js adalah platform runtime berbasis JavaScript yang berjalan di server. Dikembangkan oleh Ryan Dahl pada tahun 2009, Node.js memungkinkan penggunaan JavaScript di sisi server, di luar lingkungan browser, yang sebelumnya lebih dominan untuk pengembangan front-end. Node.js menggunakan mesin JavaScript V8 milik Google yang diintegrasikan dengan lingkungan runtime di server. Teknologi ini memungkinkan JavaScript untuk berjalan di luar browser dan dieksekusi dengan performa tinggi. Node.js menerapkan model event-driven dan non-blocking I/O, yang artinya sistem dapat menangani banyak permintaan secara bersamaan tanpa harus menunggu satu permintaan selesai sebelum memproses yang lain.

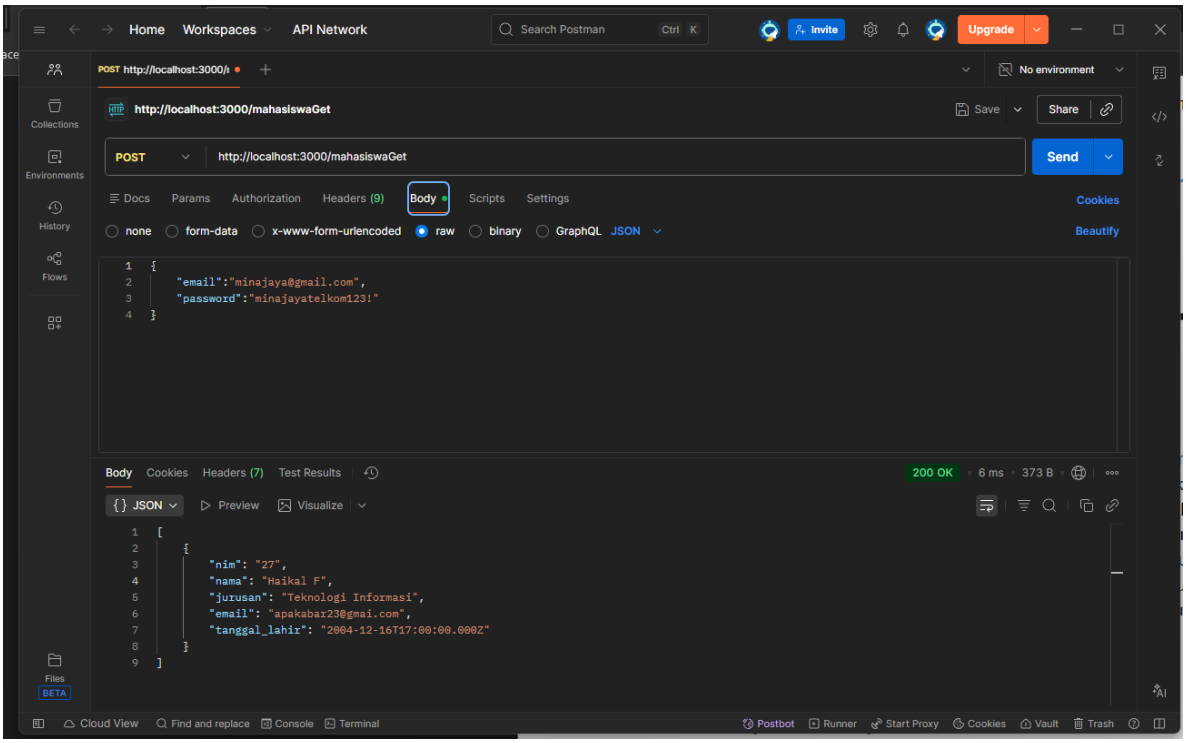
B. Tujuan

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

BAB II

HASIL

GUIDED



UNGUIDED

Server.js

```
const express = require('express');
const mysql = require('mysql2');
const bodyParser = require('body-parser');
const cors = require('cors');
const path = require('path');

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

// Middleware
app.use(cors());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

```
// Menyajikan file statis dari folder public
app.use(express.static(path.join(__dirname, 'public')));

// Konfigurasi Database
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'kampus'
});

db.connect((err) => {
    if (err) throw err;
    console.log('Terhubung ke Database MySQL...');
});

// --- RESTful API ROUTES ---

// 1. GET: Ambil semua data mahasiswa
app.get('/api/mahasiswa', (req, res) => {
    const sql = 'SELECT * FROM mahasiswa';
    db.query(sql, (err, results) => {
        if (err) return res.status(500).send(err);
        res.json(results);
    });
});

// 2. POST: Tambah data mahasiswa baru
app.post('/api/mahasiswa', (req, res) => {
    const { nim, nama, jurusan } = req.body;
    const sql = 'INSERT INTO mahasiswa (nim, nama, jurusan) VALUES (?, ?,
?)';
    db.query(sql, [nim, nama, jurusan], (err, result) => {
        if (err) return res.status(500).send(err);
        res.json({ message: 'Data berhasil ditambahkan', id:
result.insertId });
    });
});

// 3. PUT: Update data mahasiswa
app.put('/api/mahasiswa/:id', (req, res) => {
    const { nim, nama, jurusan } = req.body;
```

```
    const { id } = req.params;
    const sql = 'UPDATE mahasiswa SET nim = ?, nama = ?, jurusan = ? WHERE
id = ?';
    db.query(sql, [nim, nama, jurusan, id], (err, result) => {
        if (err) return res.status(500).send(err);
        res.json({ message: 'Data berhasil diupdate' });
    });
});

// 4. DELETE: Hapus data mahasiswa
app.delete('/api/mahasiswa/:id', (req, res) => {
    const { id } = req.params;
    const sql = 'DELETE FROM mahasiswa WHERE id = ?';
    db.query(sql, [id], (err, result) => {
        if (err) return res.status(500).send(err);
        res.json({ message: 'Data berhasil dihapus' });
    });
});

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

Index.html

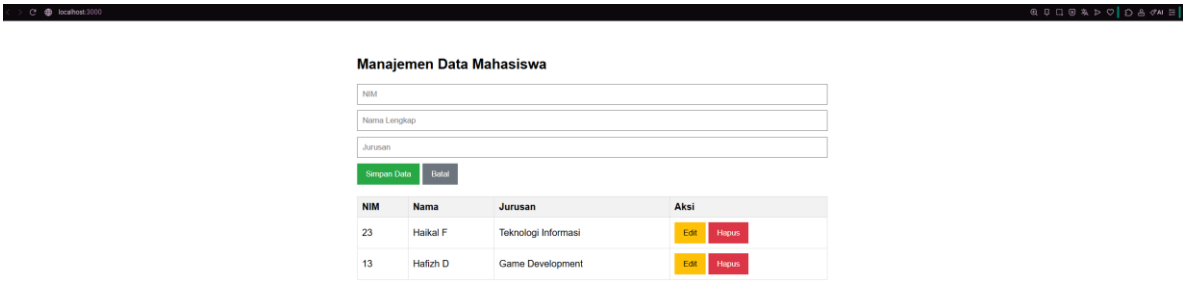
```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Data Mahasiswa</title>
    <style>
        body { font-family: sans-serif; max-width: 800px; margin: 20px
auto; padding: 20px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px;
}
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background-color: #f2f2f2; }
        .form-group { margin-bottom: 10px; }
        input { padding: 8px; width: 100%; box-sizing: border-box; }
        button { padding: 10px 15px; cursor: pointer; background-color:
#28a745; color: white; border: none; }
        button.delete { background-color: #dc3545; }
```

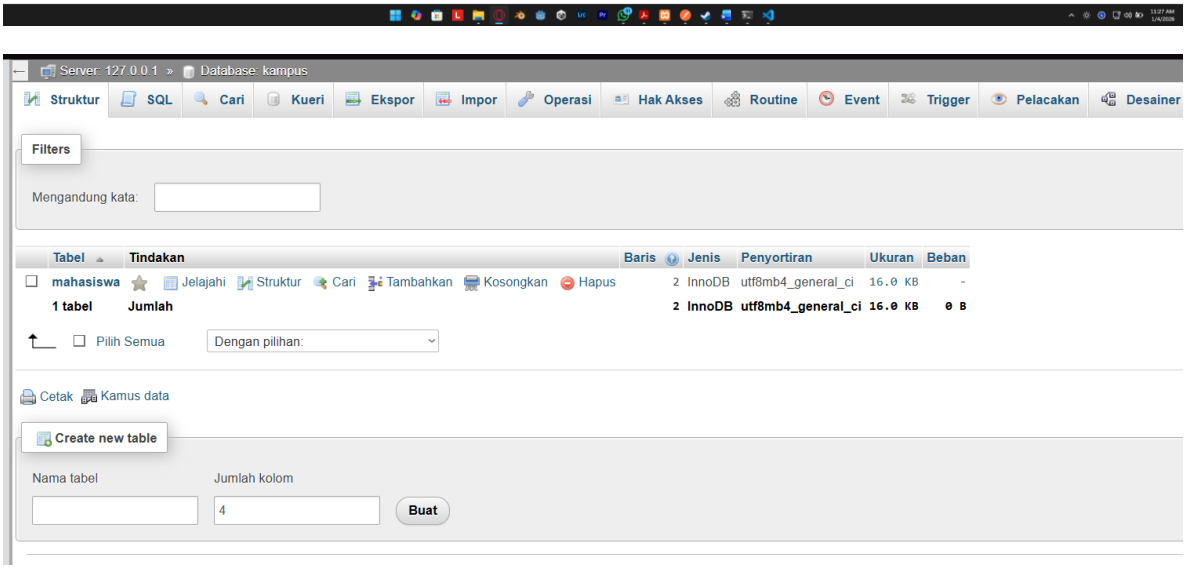
```
        button.edit { background-color: #ffc107; color: black; }
    </style>
</head>
<body>

    <h2>Manajemen Data Mahasiswa</h2>

    <div id="form-container">
        <input type="hidden" id="mhs-id">
        <div class="form-group">
            <input type="text" id="nim" placeholder="NIM" required>
        </div>
        <div class="form-group">
            <input type="text" id="nama" placeholder="Nama Lengkap"
required>
        </div>
        <div class="form-group">
            <input type="text" id="jurusan" placeholder="Jurusan"
required>
        </div>
        <button onclick="simpanData()">Simpan Data</button>
        <button onclick="resetForm()" style="background-color:
#6c757d;">Batal</button>
    </div>

    <table>
        <thead>
            <tr>
                <th>NIM</th>
                <th>Nama</th>
                <th>Jurusan</th>
                <th>Aksi</th>
            </tr>
        </thead>
        <tbody id="tabel-mahasiswa">
            </tbody>
    </table>

    <script>
        const apiUrl = 'http://localhost:3000/api/mahasiswa';

        // 1. Fungsi Load Data (READ)
        async function loadData() {
```

```
            const response = await fetch(apiUrl);
            const data = await response.json();
            const tabelBody = document.getElementById('tabel-mahasiswa');
            tabelBody.innerHTML = '';

            data.forEach(mhs => {
                const row = `
                    <tr>
                        <td>${mhs.nim}</td>
                        <td>${mhs.nama}</td>
                        <td>${mhs.jurusan}</td>
                        <td>
                            <button class="edit"
onclick="editData(${mhs.id}, '${mhs.nim}', '${mhs.nama}',
'${mhs.jurusan}')">Edit</button>
                            <button class="delete"
onclick="hapusData(${mhs.id})">Hapus</button>
                        </td>
                    </tr>
                `;
                tabelBody.innerHTML += row;
            });
        }

        // 2. Fungsi Simpan (CREATE & UPDATE)
        async function simpanData() {
            const id = document.getElementById('mhs-id').value;
            const nim = document.getElementById('nim').value;
            const nama = document.getElementById('nama').value;
            const jurusan = document.getElementById('jurusan').value;

            if(!nim || !nama || !jurusan) {
                alert('Semua field harus diisi!');
                return;
            }

            const mhsData = { nim, nama, jurusan };

            let method = 'POST';
            let url = apiUrl;

            // Jika ada ID, berarti mode EDIT (PUT)
            if (id) {
```

```
                method = 'PUT';
                url = `${apiUrl}/${id}`;
            }

            await fetch(url, {
                method: method,
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(mhsData)
            });

            resetForm();
            loadData();
        }

        // 3. Fungsi Hapus (DELETE)
        async function hapusData(id) {
            if(confirm('Yakin ingin menghapus data ini?')) {
                await fetch(`${apiUrl}/${id}`, { method: 'DELETE' });
                loadData();
            }
        }

        // Helper: Menyiapkan form untuk edit
        function editData(id, nim, nama, jurusan) {
            document.getElementById('mhs-id').value = id;
            document.getElementById('nim').value = nim;
            document.getElementById('nama').value = nama;
            document.getElementById('jurusan').value = jurusan;
        }

        // Helper: Reset form
        function resetForm() {
            document.getElementById('mhs-id').value = '';
            document.getElementById('nim').value = '';
            document.getElementById('nama').value = '';
            document.getElementById('jurusan').value = '';
        }

        // Load data saat halaman dibuka
        loadData();
    </script>
</body>
</html>
```

Output:


Manajemen Data Mahasiswa
NIM
Nama Lengkap
Jurusan
Simpan Data
Batal
NIM
Nama
Jurusan
Aksi
23
Haikal F
Teknologi Informasi
Edit
Hapus
13
Hafizh D
Game Development
Edit
Hapus


Server: 127.0.0.1 » Database: kampus
Struktur
SQL
Cari
Kueri
Ekspor
Impor
Operasi
Hak Akses
Routine
Event
Trigger
Pelacakan
Desainer
Filters
Mengandung kata:
Tabel
Tindakan
Baris
Jenis
Penyortiran
Ukuran
Beban
mahasiswa
Jelajahi
Struktur
Cari
Tambahkan
Kosongkan
Hapus
2
InnoDB
utf8mb4_general_ci
16.0 KB
-
1 tabel
Jumlah
2
InnoDB
utf8mb4_general_ci
16.0 KB
0 B
Pilih Semua
Dengan pilihan:
Cetak
Kamus data
Create new table
Nama tabel
Jumlah kolom
4
Buat

BAB III

PENUTUP

Kesimpulan

Aplikasi tersebut merupakan implementasi CRUD (Create, Read, Update, Delete) sederhana yang menggunakan Node.js dan Express sebagai backend untuk menyediakan RESTful API dalam pengelolaan data pada database MySQL. Pada sisi antarmuka, HTML dan JavaScript murni dimanfaatkan untuk berkomunikasi dengan API melalui mekanisme fetch, sehingga pengguna dapat mengelola data mahasiswa langsung melalui browser tanpa harus melakukan pemuatan ulang halaman.