LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL XIII

NODE JS

Oleh:

(Zaenarif Putra ‘Ainurdin) (2311104049)

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

1.1 Dasar Teori

Node.js merupakan platform runtime JavaScript yang berjalan disisi server dan menggunakan engine V8 dari Google untuk menghasilkan performa tinggi. Node.js menerapkan konsep event-driven dan non-blocking I/O yang memungkinkan proses penanganan banyak permintaan secara bersamaan tanpa harus menunggu satu proses selesai. Dalam pengembangan aplikasi web, Node.js sering dipadukan dengan framework Express.js yang menyediakan fitur routing, middleware, dan pengelolaan request–response secara sederhana dan terstruktur. Selain itu, penerapan RESTful API memungkinkan komunikasi antara client dan server melalui protokol HTTP dengan metode CRUD (Create, Read, Update, Delete), sehingga sistem menjadi lebih modular, scalable, dan mudah diintegrasikan dengan berbagai platform.

1.2 Tujuan

a. Ketepatan penerapan pembangunan aplikasi web menggunakan NodeJS.
b. Ketepatan penerapan pembangunan aplikasi web menggunakan Framework NodeJS.

1.3 Manfaat

Penggunaan Node.js dan Express.js dalam pembangunan aplikasi web memberikan manfaat berupa efisiensi pengolahan data, kecepatan respon server, serta kemudahan pengembangan aplikasi backend. Dengan konsep asynchronous, aplikasi mampu menangani banyak permintaan klien secara bersamaan tanpa membebani sumber daya server. RESTful API mempermudah pertukaran data antara server dan client dalam format yang terstandarisasi seperti JSON, sehingga memudahkan integrasi dengan aplikasi web maupun mobile. Selain itu, dukungan ekosistem NPM yang luas membantu pengembang mempercepat proses pengembangan dengan memanfaatkan berbagai modul siap pakai.

BAB II

HASIL PRAKTIKUM

2.1 Guided

a. Source Code

app.js

```
// app.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,
    (error) => {
      if (error) {
        return res.status(500).send("Error creating mahasiswa");
      }
      res.status(201).send("Mahasiswa created");
    }
  );
});
```

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

// ======================
// 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,
    (error) => {
      if (error) {
        return res.status(500).send("Error updating mahasiswa");
      }
      res.send("Mahasiswa updated");
    }
  );
});

// ======================
// DELETE
```

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

  dbOperations.deleteMahasiswa(id, (error) => {
    if (error) {
      return res.status(500).send("Error deleting mahasiswa");
    }
    res.send("Mahasiswa deleted");
  });
});

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

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);
  });
}

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

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;
```

package.json

```
{
  "name": "restfulapi",
  "version": "1.0.0",
  "description": "guided modul 13",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^5.2.1",
    "mysql": "^2.18.1"
  }
}
```

b. Screenshoot Output

- Inisialisasi Proyek Node.js

```
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13> npm -v
10.2.4
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13> node -v
v21.2.0
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13>
```

- Membuat Folder Proyek

```
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Praktikum13> mkdir restfulAPI


    Directory: C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Praktikum13


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/31/2025  10:36 AM                restfulAPI


PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Praktikum13> cd .\restfulAPI\
```

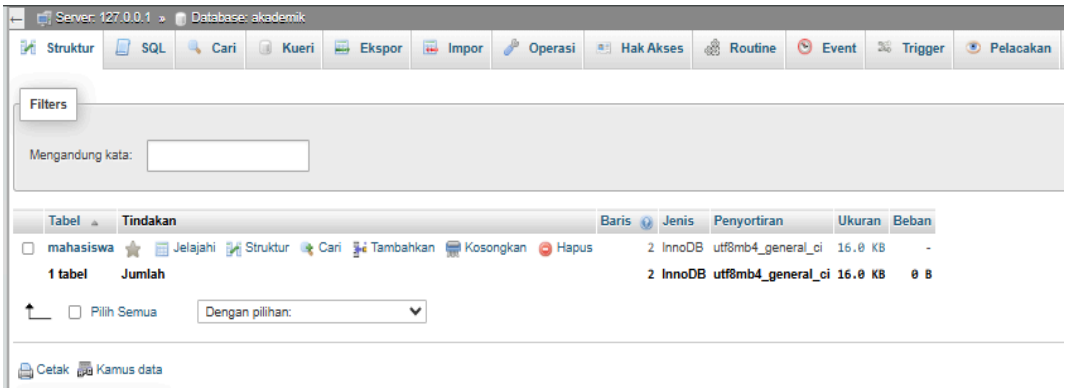
- Inisialisasi Proyek [Node.js]

```
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Praktikum13\restfulAPI> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (restfulapi) \
Sorry, name can only contain URL-friendly characters.
package name: (restfulapi)
version: (1.0.0)

description: guided modul 13
entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

About to write to C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Praktikum13\restfulAPI\package.json:

{
  "name": "restfulapi",
  "version": "1.0.0",
  "description": "guided modul 13",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

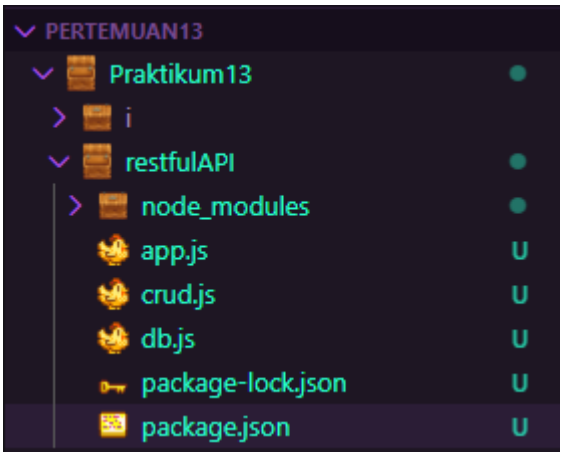
- Instalasi Dependency

```
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Praktikum13\restfulAPI> npm install express mysql

added 75 packages, and audited 76 packages in 3s

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

found 0 vulnerabilities
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Praktikum13\restfulAPI>
```

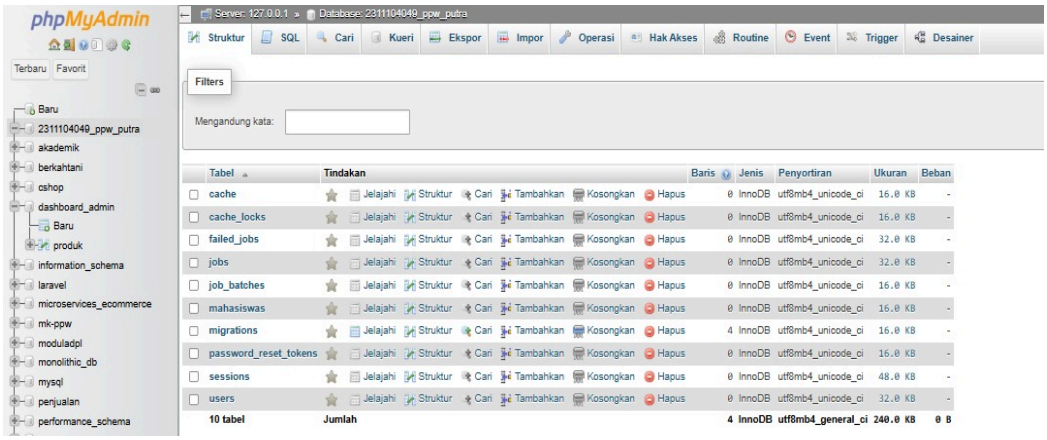
- Membuat database yang diberi nama: akademik dan tabel yang diberi nama mahasiswa



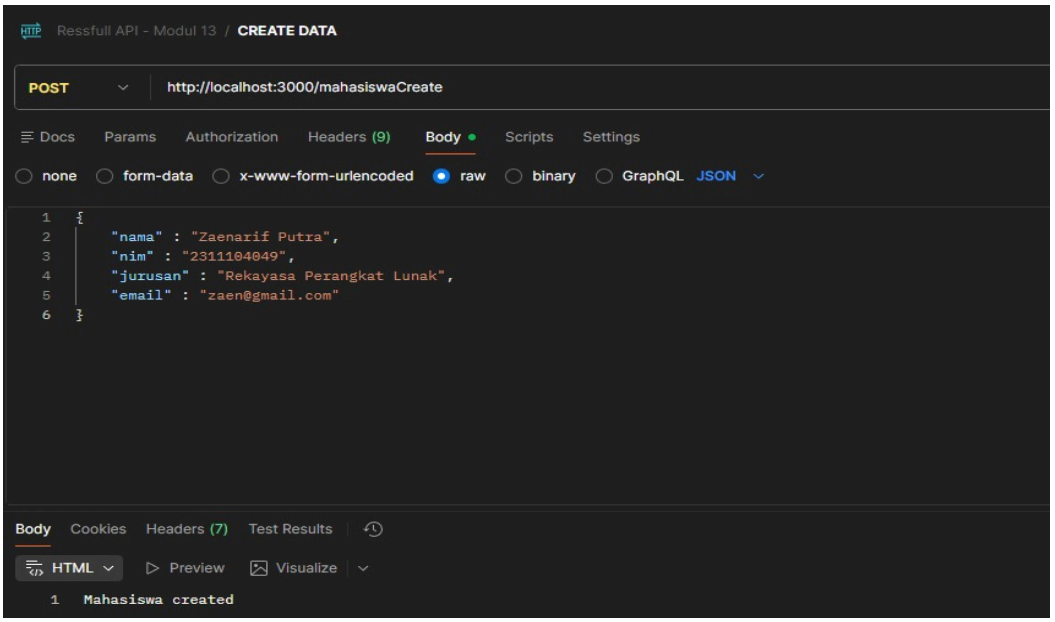
- Struktur Project guided



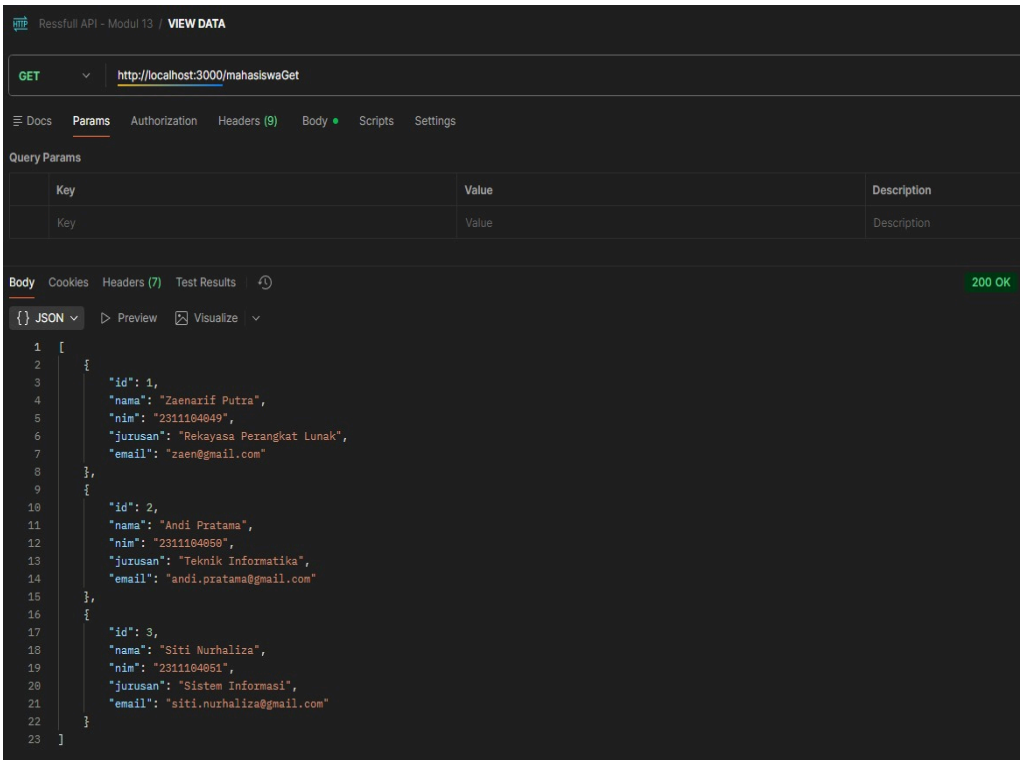
- Menyambungkan dengan menggunakan perintah php artisan migrate



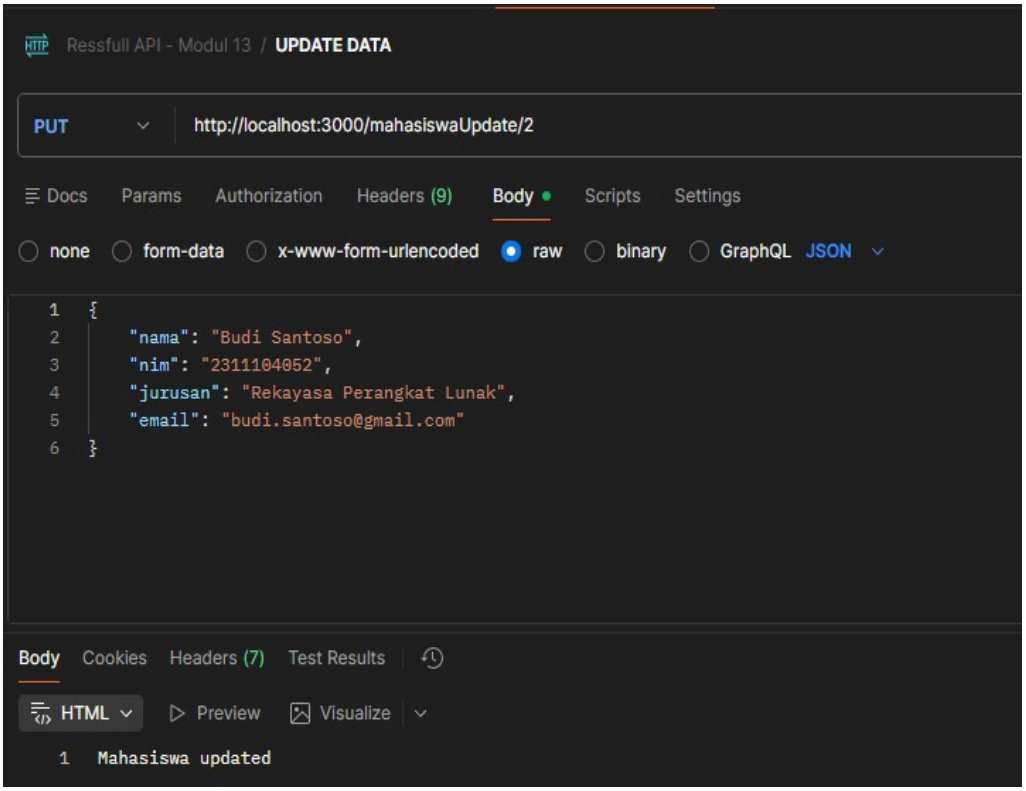
- Menjalankan dengan menggunakan postman untuk POST (Menambahkan Data)

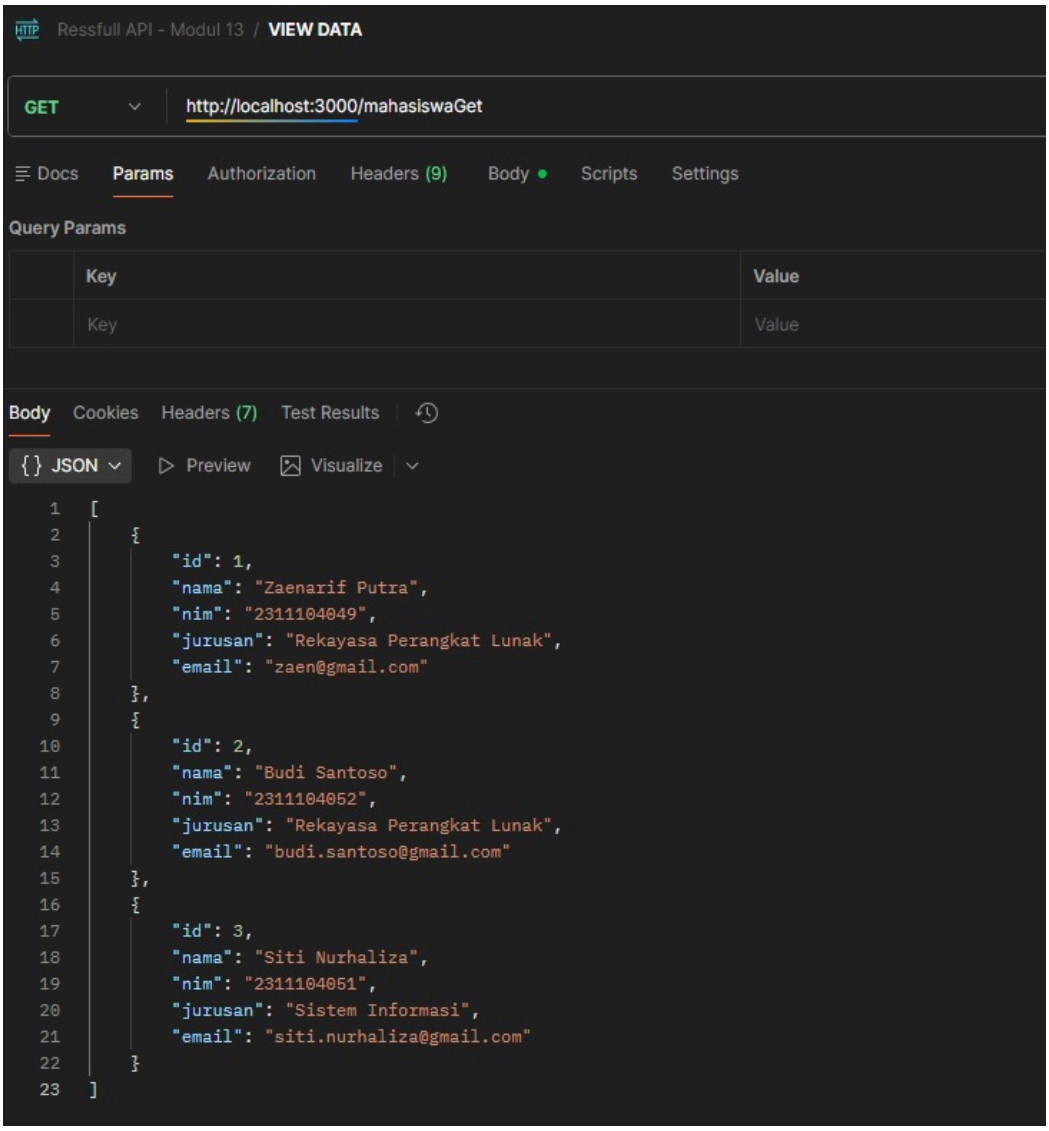


- Menjalankan GET (Melihat seluruh Data)

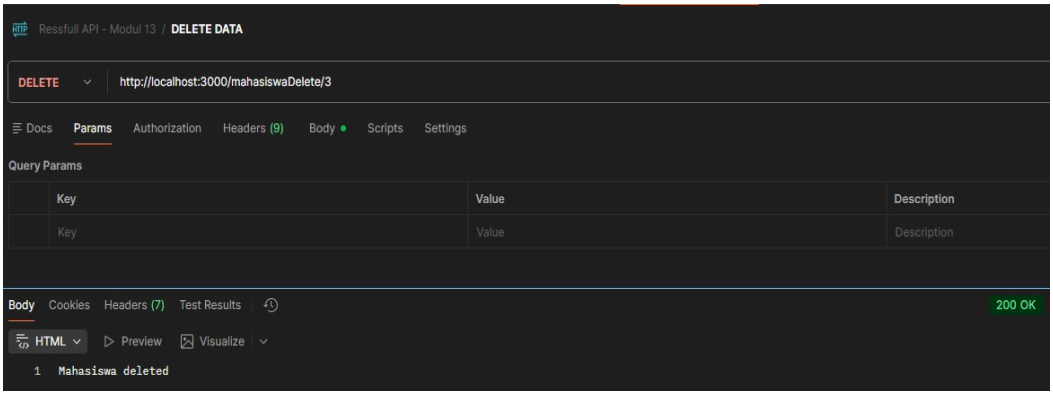


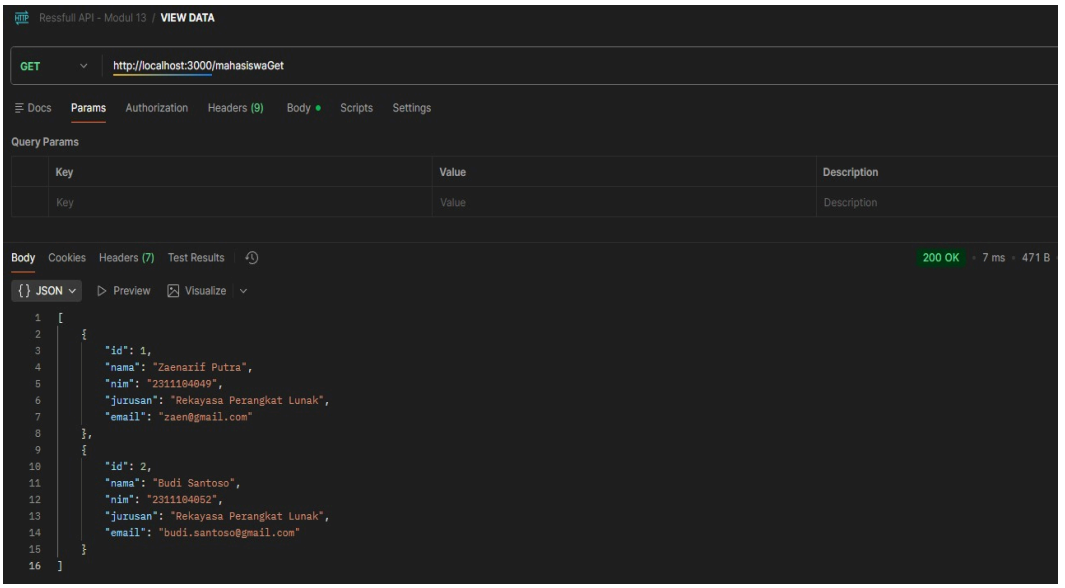
- Menjalankan PUT (Mengedit Data dengan ID: 2 & Melihat Data Setelah di Update)





- Menjalankan DEL (Menghapus sebuah data dengan menggunakan ID: 3 & melihat data apakah sudah berhasil di hapus)





c. Deksripsi Program

Program yang dibuat merupakan aplikasi RESTful API berbasis Node.js untuk mengelola data mahasiswa menggunakan Express.js sebagai framework server dan MySQL sebagai basis data. Secara alur kerja, program dimulai dari file db.js yang berfungsi membangun koneksi ke database akademik sehingga aplikasi dapat melakukan operasi penyimpanan data, kemudian crud.js berperan sebagai lapisan logika bisnis yang berisi algoritma CRUD (Create, Read, Update, Delete) menggunakan perintah SQL dengan metode query untuk memasukkan, mengambil, memperbarui, dan menghapus data mahasiswa. File app.js bertindak sebagai pusat aplikasi yang mengatur server, routing, dan alur request–response, di mana setiap endpoint HTTP (POST, GET, PUT, DELETE) menerima permintaan dari client, memproses data JSON melalui middleware express.json(), lalu meneruskannya ke fungsi yang sesuai di crud.js. Algoritma program bekerja secara berurutan: client mengirim request melalui endpoint tertentu, Express menangkap request tersebut, data diproses dan dikirim ke database melalui query MySQL, kemudian hasilnya dikembalikan ke server dan diteruskan ke client dalam bentuk respons sukses atau pesan error. Output yang dihasilkan berupa respons HTTP seperti pesan berhasil (misalnya “Mahasiswa created”, “Mahasiswa updated”) atau data mahasiswa dalam format JSON pada proses GET, sehingga aplikasi ini memudahkan pengelolaan data mahasiswa secara terstruktur, efisien, dan mudah dipahami alurnya.

2.2 Unguided

1. Latihan
    a. Source Code

config/database.js

```
const mysql = require('mysql2');
require('dotenv').config();

const pool = mysql.createPool({
    host: process.env.DB_HOST || 'localhost',
    user: process.env.DB_USER || 'root',
    password: process.env.DB_PASSWORD || '',
    database: process.env.DB_NAME || 'db_mahasiswa',
    waitForConnections: true,
    connectionLimit: 10,
    queueLimit: 0
});

// Test connection
pool.getConnection((err, connection) => {
    if (err) {
        console.error('Database Tidak Berhasil
Tersambung:', err.message);
    } else {
        console.log('Database Berhasil Tersambung');
        connection.release();
    }
});

module.exports = pool.promise()
```

controllers/mahasiswaController.js

```
const MahasiswaModel = require('../models/mahasiswaModel');

class MahasiswaController {
    // Get all students
    static async getAll(req, res) {
```

```
        try {
            const mahasiswa = await
MahasiswaModel.getAll();
            res.json({
                success: true,
                count: mahasiswa.length,
                data: mahasiswa
            });
        } catch (error) {
            res.status(500).json({
                success: false,
                error: error.message
            });
        }
    }

    // Get student by ID
    static async getById(req, res) {
        try {
            const mahasiswa = await
MahasiswaModel.getById(req.params.id);

            if (!mahasiswa) {
                return res.status(404).json({
                    success: false,
                    error: 'Mahasiswa tidak ditemukan'
                });
            }

            res.json({ success: true, data: mahasiswa });
        } catch (error) {
            res.status(500).json({
                success: false,
                error: error.message
            });
        }
    }
```

```
    // Create new student
    static async create(req, res) {
        try {
            const { nim, nama, jurusan, semester, email } =
req.body;

            // Validation
            if (!nim || !nama || !jurusan) {
                return res.status(400).json({
                    success: false,
                    error: 'NIM, Nama, dan Jurusan wajib
diisi'
                });
            }

            const newMahasiswa = await
MahasiswaModel.create(req.body);
            res.status(201).json({
                success: true,
                message: 'Mahasiswa berhasil ditambahkan',
                data: newMahasiswa
            });
        } catch (error) {
            res.status(500).json({
                success: false,
                error: error.message
            });
        }
    }

    // Update student
    static async update(req, res) {
        try {
            const { id } = req.params;
            const { nim, nama, jurusan, semester, email } =
req.body;
```

```
            // Validation
            if (!nim || !nama || !jurusan) {
                return res.status(400).json({
                    success: false,
                    error: 'NIM, Nama, dan Jurusan wajib
diisi'
                });
            }

            const updatedMahasiswa = await
MahasiswaModel.update(id, req.body);
            res.json({
                success: true,
                message: 'Mahasiswa berhasil diperbarui',
                data: updatedMahasiswa
            });
        } catch (error) {
            res.status(500).json({
                success: false,
                error: error.message
            });
        }
    }

    // Delete student
    static async delete(req, res) {
        try {
            await MahasiswaModel.delete(req.params.id);
            res.json({
                success: true,
                message: 'Mahasiswa berhasil dihapus'
            });
        } catch (error) {
            res.status(500).json({
                success: false,
                error: error.message
```

```
            });
        }
    }

    // Search students
    static async search(req, res) {
        try {
            const { keyword } = req.query;
            const results = await
MahasiswaModel.search(keyword);
            res.json({
                success: true,
                keyword: keyword,
                count: results.length,
                data: results
            });
        } catch (error) {
            res.status(500).json({
                success: false,
                error: error.message
            });
        }
    }
}

module.exports = MahasiswaController;
```

models/mahasiswaModel.js

```
const db = require('../config/database');

class MahasiswaModel {
    // Get all students
    static async getAll() {
        const [rows] = await db.query('SELECT * FROM
mahasiswa ORDER BY id DESC');
        return rows;
    }
```

```
    // Get student by ID
    static async getById(id) {
        const [rows] = await db.query('SELECT * FROM
mahasiswa WHERE id = ?', [id]);
        return rows[0];
    }

    // Create new student
    static async create(data) {
        const { nim, nama, jurusan, semester, email } =
data;
        const [result] = await db.query(
            'INSERT INTO mahasiswa (nim, nama, jurusan,
semester, email) VALUES (?, ?, ?, ?, ?)',
            [nim, nama, jurusan, semester, email]
        );
        return { id: result.insertId, ...data };
    }

    // Update student
    static async update(id, data) {
        const { nim, nama, jurusan, semester, email } =
data;
        await db.query(
            'UPDATE mahasiswa SET nim = ?, nama = ?,
jurusan = ?, semester = ?, email = ? WHERE id = ?',
            [nim, nama, jurusan, semester, email, id]
        );
        return { id, ...data };
    }

    // Delete student
    static async delete(id) {
        await db.query('DELETE FROM mahasiswa WHERE id =
?', [id]);
        return true;
```

```
    }

    // Search students
    static async search(keyword) {
        const [rows] = await db.query(
            'SELECT * FROM mahasiswa WHERE nama LIKE ? OR
nim LIKE ? OR jurusan LIKE ?',
            [`%${keyword}%`, `%${keyword}%`,
`%${keyword}%`]
        );
        return rows;
    }
}

module.exports = MahasiswaModel;
```

public/js/app.js

```
// Konfigurasi API
const API_BASE = '/api/mahasiswa';

// Format JSON untuk tampilan
function formatJSON(data) {
    return JSON.stringify(data, null, 2);
}

// Tampilkan response di box
function showResponse(data) {
    const responseBox =
document.getElementById('apiResponse');
    responseBox.innerHTML =
`<pre>${formatJSON(data)}</pre>`;
}

// Update data table
function updateTable(data) {
    const tableBody = document.getElementById('tableBody');
    const totalData = document.getElementById('totalData');
```

```
    if (!data || data.length === 0) {
        tableBody.innerHTML = `
            <tr>
                <td colspan="7" class="text-center
text-muted">
                    Tidak ada data mahasiswa
                </td>
            </tr>
        `;
        totalData.textContent = '0';
        return;
    }

    let html = '';
    data.forEach(student => {
        html += `
            <tr>
                <td>${student.id}</td>
                <td><strong>${student.nim}</strong></td>
                <td>${student.nama}</td>
                <td>${student.jurusan}</td>
                <td>${student.semester || '-'}</td>
                <td>${student.email || '-'}</td>
                <td>
                    <button class="btn btn-sm btn-info"
onclick="getStudentById(${student.id})">
                        👁️ Lihat
                    </button>
                    <button class="btn btn-sm btn-danger"
onclick="confirmDelete(${student.id})">
                        🗑️ Hapus
                    </button>
                </td>
            </tr>
        `;
    });
```

```
    tableBody.innerHTML = html;
    totalData.textContent = data.length;
}

// Ambil semua data mahasiswa
async function getAllStudents() {
    try {
        const response = await fetch(API_BASE);
        const data = await response.json();

        showResponse(data);
        if (data.success && data.data) {
            updateTable(data.data);
        }

        return data;
    } catch (error) {
        showResponse({ error: 'Gagal mengambil data: ' +
error.message });
    }
}

// Ambil data berdasarkan ID
async function getById() {
    const id = document.getElementById('searchId').value;
    if (!id) {
        alert('Masukkan ID terlebih dahulu');
        return;
    }

    try {
        const response = await fetch(`${API_BASE}/${id}`);
        const data = await response.json();
        showResponse(data);
    } catch (error) {
        showResponse({ error: 'Gagal mengambil data: ' +
```

```
error.message });
    }
}

// Cari mahasiswa
async function searchStudents() {
    const keyword =
document.getElementById('searchKeyword').value;
    if (!keyword) {
        alert('Masukkan kata kunci pencarian');
        return;
    }

    try {
        const response = await
fetch(`${API_BASE}/search/all?keyword=${encodeURIComponent(
keyword)}`);
        const data = await response.json();
        showResponse(data);
        if (data.success && data.data) {
            updateTable(data.data);
        }
    } catch (error) {
        showResponse({ error: 'Gagal mencari: ' +
error.message });
    }
}

// Tampilkan form tambah
function showAddForm() {
    const modal = new
bootstrap.Modal(document.getElementById('addModal'));
    modal.show();
}

// Submit form tambah
async function submitAddForm() {
```

```
    const data = {
        nim: document.getElementById('addNim').value,
        nama: document.getElementById('addNama').value,
        jurusan:
document.getElementById('addJurusan').value,
        semester:
document.getElementById('addSemester').value || null,
        email: document.getElementById('addEmail').value ||
null
    };

    // Validasi
    if (!data.nim || !data.nama || !data.jurusan) {
        alert('NIM, Nama, dan Jurusan wajib diisi!');
        return;
    }

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

        const result = await response.json();
        showResponse(result);

        if (result.success) {
            // Tutup modal
            const modal =
bootstrap.Modal.getInstance(document.getElementById('addMod
al'));
            modal.hide();

            // Reset form
```

```
            document.getElementById('addForm').reset();

            // Refresh data
            getAllStudents();
        }
    } catch (error) {
        showResponse({ error: 'Gagal menambah data: ' +
error.message });
    }
}

// Hapus data
async function deleteStudent() {
    const id = document.getElementById('deleteId').value;
    if (!id) {
        alert('Masukkan ID terlebih dahulu');
        return;
    }

    if (!confirm(`Apakah Anda yakin ingin menghapus data
dengan ID ${id}?`)) {
        return;
    }

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

        const result = await response.json();
        showResponse(result);

        if (result.success) {
            // Refresh data
            getAllStudents();
            document.getElementById('deleteId').value = '';
        }
```

```
    } catch (error) {
        showResponse({ error: 'Gagal menghapus data: ' +
error.message });
    }
}

// Konfirmasi hapus dari tabel
function confirmDelete(id) {
    if (confirm(`Hapus mahasiswa dengan ID ${id}?`)) {
        fetch(`${API_BASE}/${id}`, {
            method: 'DELETE'
        })
        .then(response => response.json())
        .then(result => {
            showResponse(result);
            if (result.success) {
                getAllStudents();
            }
        })
        .catch(error => {
            showResponse({ error: 'Gagal menghapus: ' +
error.message });
        });
    }
}

// Ambil data single student
async function getStudentById(id) {
    try {
        const response = await fetch(`${API_BASE}/${id}`);
        const data = await response.json();
        showResponse(data);
    } catch (error) {
        showResponse({ error: 'Gagal mengambil data: ' +
error.message });
    }
}
```

```
// Test API connection
async function testAPI() {
    try {
        const response = await fetch('/api/test');
        const data = await response.json();
        showResponse(data);
    } catch (error) {
        showResponse({ error: 'API tidak terhubung: ' +
error.message });
    }
}

// Tambah data contoh
async function addSampleData() {
    const sampleData = {
        nim: '2021' + Math.floor(Math.random() *
10000).toString().padStart(4, '0'),
        nama: 'Mahasiswa Contoh',
        jurusan: 'Teknik Informatika',
        semester: Math.floor(Math.random() * 8) + 1,
        email: `contoh${Math.floor(Math.random() *
1000)}@example.com`
    };

    try {
        const response = await fetch(API_BASE, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(sampleData)
        });

        const result = await response.json();
        showResponse(result);
```

```
        if (result.success) {
            getAllStudents();
        }
    } catch (error) {
        showResponse({ error: 'Gagal menambah data contoh: ' + error.message });
    }
}

// Inisialisasi saat halaman dimuat
document.addEventListener('DOMContentLoaded', function() {
    // Load data awal
    getAllStudents();
    testAPI();

    // Update port info
    const serverPort = window.location.port || '3000';
    document.getElementById('serverPort').textContent = serverPort;
});
```

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>Aplikasi Data Mahasiswa</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body { padding: 20px; background-color: #f8f9fa; }
        .container { max-width: 1200px; }
        .card { margin-bottom: 20px; }
        .status-circle {
```

```
            width: 10px; height: 10px; border-radius: 50%;
            display: inline-block; margin-right: 5px;
        }
        .status-active { background-color: #28a745; }
        .endpoint-card { cursor: pointer; transition:
transform 0.2s; }
        .endpoint-card:hover { transform: translateY(-2px);
}
        .response-box {
            background-color: #f8f9fa;
            border: 1px solid #dee2e6;
            border-radius: 5px;
            padding: 15px;
            max-height: 300px;
            overflow-y: auto;
            font-family: monospace;
            font-size: 14px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="text-center mb-4">
            <h1 class="text-primary">📚 Aplikasi Data
Mahasiswa</h1>
            <p class="lead">Node.js + Express.js + MySQL +
REST API</p>
            <div class="d-flex justify-content-center
align-items-center mb-3">
                <span class="status-circle
status-active"></span>
                <small class="text-muted"
id="apiStatus">API Connected</small>
            </div>
        </div>

        <div class="row">
```

```
            <!-- Info Panel -->
            <div class="col-md-4">
                <div class="card">
                    <div class="card-header bg-info
text-white">
                        <h5 class="mb-0">📊 Informasi
Sistem</h5>
                    </div>
                    <div class="card-body">
                        <p><strong>Status Server:</strong>
<span class="text-success">● Berjalan</span></p>
                        <p><strong>Database:</strong> <span
id="dbName">db_mahasiswa</span></p>
                        <p><strong>Total Data:</strong>
<span id="totalData">-</span> mahasiswa</p>
                        <p><strong>Port:</strong> <span
id="serverPort">3000</span></p>
                        <hr>
                        <h6>📋 Instruksi:</h6>
                        <ol class="small">
                            <li>Test API di
Postman/Insomnia</li>
                            <li>Gunakan endpoint di
samping</li>
                            <li>Data akan tampil di
bawah</li>
                        </ol>
                    </div>
                </div>

                <!-- Quick Actions -->
                <div class="card">
                    <div class="card-header bg-primary
text-white">
                        <h5 class="mb-0">⚡ Quick
Actions</h5>
                    </div>
```

```
                    <div class="card-body">
                        <button class="btn btn-success
btn-sm w-100 mb-2" onclick="getAllStudents()">
                            🔄 Refresh Data
                        </button>
                        <button class="btn btn-primary
btn-sm w-100 mb-2" onclick="testAPI()">
                            🧪 Test API Connection
                        </button>
                        <button class="btn btn-warning
btn-sm w-100" onclick="addSampleData()">
                            ➕ Tambah Data Contoh
                        </button>
                    </div>
                </div>
            </div>

            <!-- API Endpoints -->
            <div class="col-md-8">
                <div class="card">
                    <div class="card-header bg-success
text-white">
                        <h5 class="mb-0">🔗 REST API
Endpoints</h5>
                    </div>
                    <div class="card-body">
                        <div class="row">
                            <!-- GET All -->
                            <div class="col-md-6 mb-3">
                                <div class="card
endpoint-card border-success" onclick="getAllStudents()">
                                    <div class="card-body">
                                        <span class="badge
bg-success">GET</span>

<h6>/api/mahasiswa</h6>
                                        <p class="small
```

```
text-muted">Ambil semua data mahasiswa</p>
                                    </div>
                                </div>
                            </div>

                            <!-- POST -->
                            <div class="col-md-6 mb-3">
                                <div class="card
endpoint-card border-primary" onclick="showAddForm()">
                                    <div class="card-body">
                                        <span class="badge
bg-primary">POST</span>

<h6>/api/mahasiswa</h6>
                                        <p class="small
text-muted">Tambah data mahasiswa baru</p>
                                    </div>
                                </div>
                            </div>

                            <!-- GET by ID -->
                            <div class="col-md-6 mb-3">
                                <div class="card
endpoint-card border-info">
                                    <div class="card-body">
                                        <span class="badge
bg-info">GET</span>

<h6>/api/mahasiswa/:id</h6>
                                        <p class="small
text-muted">Ambil data berdasarkan ID</p>
                                        <div
class="input-group input-group-sm">
                                            <input
type="number" id="searchId" class="form-control"
placeholder="ID" min="1">
                                            <button
```

```
class="btn btn-info" onclick="getById()">Cari</button>
                                        </div>
                                    </div>
                                </div>

                                <!-- PUT -->
                                <div class="col-md-6 mb-3">
                                    <div class="card
endpoint-card border-warning">
                                        <div class="card-body">
                                            <span class="badge
bg-warning">PUT</span>

<h6>/api/mahasiswa/:id</h6>
                                            <p class="small
text-muted">Update data mahasiswa</p>
                                            <div
class="input-group input-group-sm">
                                                <input
type="number" id="editIdInput" class="form-control"
placeholder="ID" min="1">
                                                <button
class="btn btn-warning"
onclick="showEditFormById()">Edit</button>
                                            </div>
                                        </div>
                                    </div>
                                </div>

                                <!-- DELETE -->
                                <div class="col-md-6 mb-3">
                                    <div class="card
endpoint-card border-danger">
                                        <div class="card-body">
                                            <span class="badge
bg-danger">DELETE</span>
```

```
<h6>/api/mahasiswa/:id</h6>
                                        <p class="small
text-muted">Hapus data mahasiswa</p>
                                        <div
class="input-group input-group-sm">
                                            <input
type="number" id="deleteId" class="form-control"
placeholder="ID" min="1">
                                            <button
class="btn btn-danger"
onclick="deleteStudent()">Hapus</button>
                                        </div>
                                    </div>
                                </div>

                                <!-- SEARCH -->
                                <div class="col-md-6 mb-3">
                                    <div class="card
endpoint-card border-secondary">
                                        <div class="card-body">
                                            <span class="badge
bg-secondary">GET</span>

<h6>/api/mahasiswa/search/all</h6>
                                        <p class="small
text-muted">Cari mahasiswa</p>
                                        <div
class="input-group input-group-sm">
                                            <input
type="text" id="searchKeyword" class="form-control"
placeholder="Kata kunci">
                                            <button
class="btn btn-secondary"
onclick="searchStudents()">Cari</button>
                                        </div>
```

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

                    <!-- Response Section -->
                    <div class="card mt-3">
                        <div class="card-header bg-dark
text-white">
                            <h5 class="mb-0">📤 API
Response</h5>
                        </div>
                        <div class="card-body">
                            <div class="response-box"
id="apiResponse">
                                <em>Response akan muncul di
sini...</em>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Data Table -->
            <div class="card mt-4">
                <div class="card-header bg-secondary text-white
d-flex justify-content-between align-items-center">
                    <h5 class="mb-0">👥 Data Mahasiswa</h5>
                    <button class="btn btn-light btn-sm"
onclick="getAllStudents()">
                        <span id="refreshIcon">🔄</span>
Refresh
                    </button>
                </div>
                <div class="card-body">
```

```
                <div class="table-responsive">
                    <table class="table table-hover"
id="dataTable">
                        <thead>
                            <tr>
                                <th>ID</th>
                                <th>NIM</th>
                                <th>Nama</th>
                                <th>Jurusan</th>
                                <th>Semester</th>
                                <th>Email</th>
                                <th>Aksi</th>
                            </tr>
                        </thead>
                        <tbody id="tableBody">
                            <tr>
                                <td colspan="7"
class="text-center">
                                    <div
class="spinner-border spinner-border-sm text-primary"
role="status">
                                        <span
class="visually-hidden">Loading...</span>
                                    </div>
                                    Memuat data...
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>

        <!-- Footer -->
        <footer class="text-center mt-4 text-muted">
            <p>© 2024 Aplikasi Data Mahasiswa - Node.js +
Express + MySQL</p>
```

```
        </footer>
    </div>

    <!-- Modal Add Form -->
    <div class="modal fade" id="addModal" tabindex="-1">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title">➕ Tambah Data
Mahasiswa</h5>
                    <button type="button" class="btn-close"
data-bs-dismiss="modal"></button>
                </div>
                <div class="modal-body">
                    <form id="addForm">
                        <div class="mb-3">
                            <label class="form-label">NIM
*</label>
                            <input type="text"
class="form-control" id="addNim" required>
                        </div>
                        <div class="mb-3">
                            <label class="form-label">Nama
Lengkap *</label>
                            <input type="text"
class="form-control" id="addNama" required>
                        </div>
                        <div class="mb-3">
                            <label
class="form-label">Jurusan *</label>
                            <select class="form-select"
id="addJurusan" required>
                                <option value="">Pilih
Jurusan</option>
                                <option value="Teknik
Informatika">Teknik Informatika</option>
                                <option value="Sistem
```

```
Informasi">Sistem Informasi</option>
                                    <option value="Teknik
Komputer">Teknik Komputer</option>
                                    <option value="Manajemen
Informatika">Manajemen Informatika</option>
                                    <option value="Teknik
Elektro">Teknik Elektro</option>
                                </select>
                            </div>
                            <div class="row">
                                <div class="col-md-6 mb-3">
                                    <label
class="form-label">Semester</label>
                                    <input type="number"
class="form-control" id="addSemester" min="1" max="14">
                                </div>
                                <div class="col-md-6 mb-3">
                                    <label
class="form-label">Email</label>
                                    <input type="email"
class="form-control" id="addEmail">
                                </div>
                            </div>
                        </form>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn
btn-secondary" data-bs-dismiss="modal">Batal</button>
                        <button type="button" class="btn
btn-primary" onclick="submitAddForm()">Simpan</button>
                    </div>
                </div>
            </div>
        </div>

        <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">
                <form id="editForm">
                    <input type="hidden" id="editId">
                    <div class="mb-3">
                        <label class="form-label">NIM
*</label>
                        <input type="text"
class="form-control" id="editNim" required>
                    </div>
                    <div class="mb-3">
                        <label class="form-label">Nama
Lengkap *</label>
                        <input type="text"
class="form-control" id="editNama" required>
                    </div>
                    <div class="mb-3">
                        <label class="form-label">Jurusan
*</label>
                        <select class="form-select"
id="editJurusan" required>
                            <option value="">Pilih
Jurusan</option>
                            <option value="Teknik
Informatika">Teknik Informatika</option>
                            <option value="Sistem
Informasi">Sistem Informasi</option>
                            <option value="Teknik
Komputer">Teknik Komputer</option>
                            <option value="Manajemen
Informatika">Manajemen Informatika</option>
```

```
                            <option value="Teknik
Elektro">Teknik Elektro</option>
                        </select>
                    </div>
                    <div class="row">
                        <div class="col-md-6 mb-3">
                            <label
class="form-label">Semester</label>
                            <input type="number"
class="form-control" id="editSemester" min="1" max="14">
                        </div>
                        <div class="col-md-6 mb-3">
                            <label
class="form-label">Email</label>
                            <input type="email"
class="form-control" id="editEmail">
                        </div>
                    </div>
                </form>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn
btn-secondary" data-bs-dismiss="modal">Batal</button>
                <button type="button" class="btn
btn-primary" onclick="submitEditForm()">Update
Data</button>
            </div>
        </div>
    </div>
    </div>

    <!-- Scripts -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/b
ootstrap.bundle.min.js"></script>
    <script src="js/app.js"></script>
</body>
```

```
</html>
```

routes/mahasiswaRoutes.js

```
const express = require('express');
const router = express.Router();
const MahasiswaController =
require('../controllers/mahasiswaController');

// GET all students
router.get('/', MahasiswaController.getAll);

// GET student by ID
router.get('/:id', MahasiswaController.getById);

// POST create new student
router.post('/', MahasiswaController.create);

// PUT update student
router.put('/:id', MahasiswaController.update);

// DELETE student
router.delete('/:id', MahasiswaController.delete);

// GET search students
router.get('/search/all', MahasiswaController.search);

module.exports = router;
```

app.js

```
const express = require('express');
const cors = require('cors');
require('dotenv').config();

const mahasiswaRoutes =
require('./routes/mahasiswaRoutes');
```

```
const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static('src/public'));

// Routes
app.use('/api/mahasiswa', mahasiswaRoutes);

// Test route
app.get('/api/test', (req, res) => {
    res.json({
        success: true,
        message: 'API Mahasiswa is running!',
        endpoints: {
            getAll: 'GET /api/mahasiswa',
            getById: 'GET /api/mahasiswa/:id',
            create: 'POST /api/mahasiswa',
            update: 'PUT /api/mahasiswa/:id',
            delete: 'DELETE /api/mahasiswa/:id',
            search: 'GET
/api/mahasiswa/search/all?keyword=...'
        }
    });
});

// Home route
app.get('/', (req, res) => {
    res.sendFile(__dirname + '/public/index.html');
});

// 404 handler
app.use((req, res) => {
    res.status(404).json({
```

```
        success: false,
        error: 'Endpoint tidak ditemukan'
    });
});

// Error handler
app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).json({
        success: false,
        error: 'Terjadi kesalahan server'
    });
});

// Start server
app.listen(PORT, () => {
    console.log('='.repeat(50));
    console.log(`SERVER BERJALAN DI:
http://localhost:${PORT}`);
    console.log(`DATABASE: ${process.env.DB_NAME}`);
    console.log('='.repeat(50));
    console.log('ENDPOINTS:');
    console.log(`   Halaman Web:
http://localhost:${PORT}`);
    console.log(`   Test API:
http://localhost:${PORT}/api/test`);
    console.log(`   Data Mahasiswa:
http://localhost:${PORT}/api/mahasiswa`);
    console.log('='.repeat(50));
});
```

package.json

```
{
  "name": "aplikasi-mahasiswa",
  "version": "1.0.0",
  "description": "Tugas Unguided modul13",
  "main": "src/app.js",
```

```
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node src/app.js",
    "dev": "nodemon src/app.js",
    "setup-db": "node setup-database.js"
  },
  "author": "Zaenarif Putra 'Ainurdin",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^17.2.3",
    "express": "^5.2.1",
    "mysql2": "^3.16.0"
  },
  "devDependencies": {
    "nodemon": "^3.1.11"
  }
}
```

setup-database.js

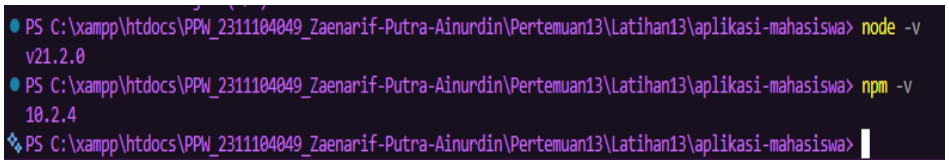
```
const mysql = require('mysql2');
require('dotenv').config();

const connection = mysql.createConnection({
    host: process.env.DB_HOST || 'localhost',
    user: process.env.DB_USER || 'root',
    password: process.env.DB_PASSWORD || ''
});

async function setupDatabase() {
    try {
        console.log('Setting up database...');

        // Create database
        await connection.promise().query(`CREATE DATABASE
IF NOT EXISTS \`${process.env.DB_NAME ||
'db_mahasiswa'}\``);
```

```
        console.log('Database created');

        // Use database
        await connection.promise().query(`USE
\`${process.env.DB_NAME || 'db_mahasiswa'}\``);

        // Create table
        const createTableSQL = `
            CREATE TABLE IF NOT EXISTS mahasiswa (
                id INT PRIMARY KEY AUTO_INCREMENT,
                nim VARCHAR(20) UNIQUE NOT NULL,
                nama VARCHAR(100) NOT NULL,
                jurusan VARCHAR(50) NOT NULL,
                semester INT,
                email VARCHAR(100),
                created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
            )
        `;

        await connection.promise().query(createTableSQL);
        console.log('Table created');

        // Insert sample data
        const sampleData = [
            ['20210001', 'Ahmad Rizki', 'Teknik
Informatika', 5, 'ahmad@example.com'],
            ['20210002', 'Siti Nurhaliza', 'Sistem
Informasi', 3, 'siti@example.com'],
            ['20210003', 'Budi Santoso', 'Teknik Komputer',
7, 'budi@example.com'],
            ['20210004', 'Dewi Lestari', 'Manajemen
Informatika', 4, 'dewi@example.com'],
            ['20210005', 'Rina Wijaya', 'Teknik Elektro',
6, 'rina@example.com']
        ];
```

```
        for (const data of sampleData) {
            await connection.promise().query(
                'INSERT IGNORE INTO mahasiswa (nim, nama,
jurusan, semester, email) VALUES (?, ?, ?, ?, ?)',
                data
            );
        }

        console.log('Sample data inserted');
        console.log('Database setup completed!');

    } catch (error) {
        console.error('Error:', error.message);
    } finally {
        connection.end();
    }
}

setupDatabase();
```

b. Screenshoot Output

1. Setup dan check versi dari node & npm



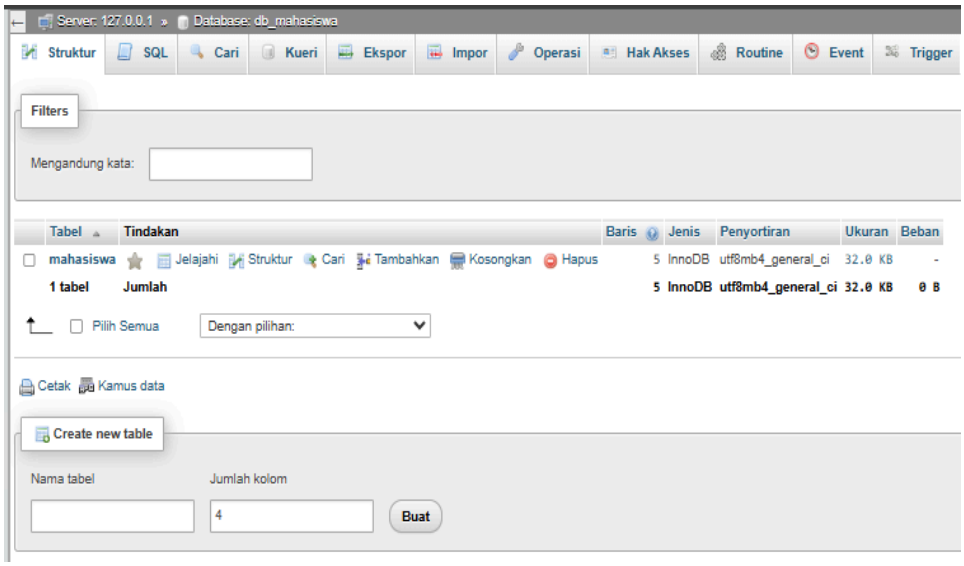
2. Setup inisialisasi node.js

```
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Latihan13\aplikasi-mahasiswa> npm init
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (aplikasi-mahasiswa)
version: (1.0.0)
description: Tugas Unguided modul13
entry point: (index.js)
test command:
git repository:
keywords:
author: Zaenarif Putra 'Ainurdin
license: (ISC)
About to write to C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Latihan13\aplikasi-mahasiswa\package.json:

{
  "name": "aplikasi-mahasiswa",
  "version": "1.0.0",
  "description": "Tugas Unguided modul13",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Zaenarif Putra 'Ainurdin",
  "license": "ISC"
}


Is this OK? (yes) yes
```

3. Setup instalasi untuk npm install express mysql2 cors dotenv & npm install -D nodemon

```
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Latihan13\aplikasi-mahasiswa> npm install express mysql2 cors dotenv
>> npm install -D nodemon

added 78 packages, and audited 79 packages in 3s

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

found 0 vulnerabilities

added 27 packages, and audited 106 packages in 3s

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

found 0 vulnerabilities
```

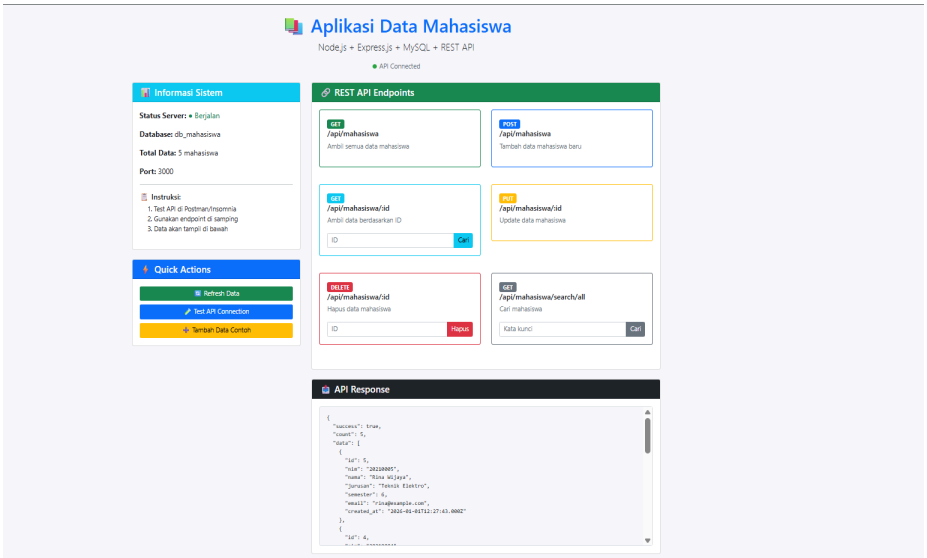
4. Package.json yang dihasilkan dan setup syntax sudah ada pada source code yang kemudian sebelum memasukkan syntax setup db terlebih dahulu



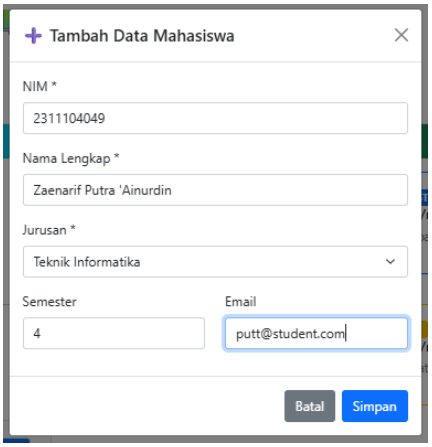
5. Menjalankan npm run dev untuk menjalankan semuanya seperti berikut:

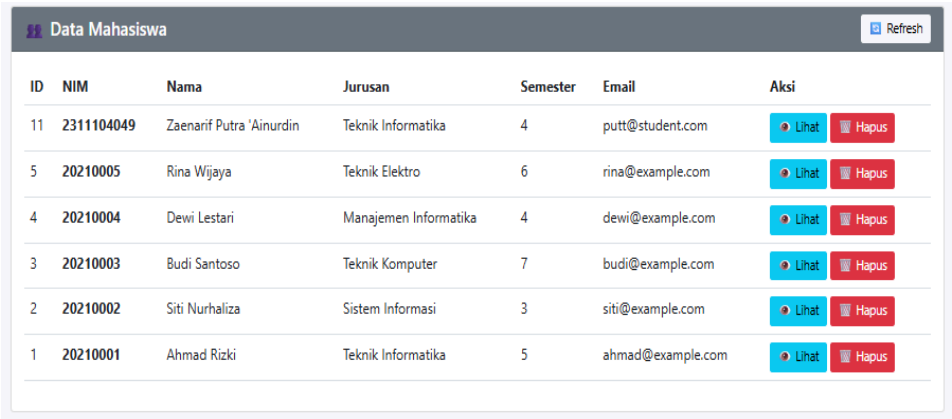
```
PS C:\xampp\htdocs\PPW_2311104049_Zaenarif-Putra-Ainurdin\Pertemuan13\Latihan13\aplikasi-mahasiswa> npm run dev

> aplikasi-mahasiswa@1.0.0 dev
> nodemon src/app.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/app.js`
[dotenv@17.2.3] injecting env (0) from .env -- tip: 🔐 encrypt with Dotenvx: https://dotenvx.com
[dotenv@17.2.3] injecting env (0) from .env -- tip: 👥 sync secrets across teammates & machines: https://dotenvx.com/ops
================================================
🚀 SERVER BERJALAN DI: http://localhost:3000
📊 DATABASE: undefined
================================================
📌 ENDPOINTS:
   Halaman Web: http://localhost:3000
   Test API: http://localhost:3000/api/test
   Data Mahasiswa: http://localhost:3000/api/mahasiswa
================================================
✅ Database Berhasil Tersambung
```

6. Kemudian berikut tampilan restfulApi sederhana:

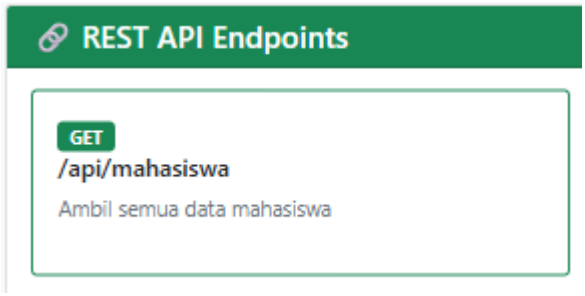


7. Testing menggunakan web POST (Menambahkan Data)



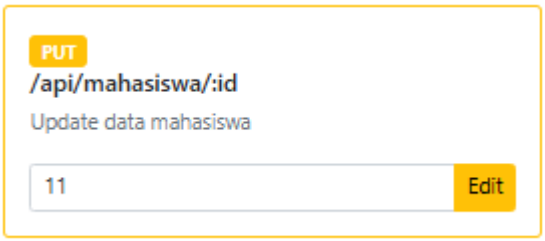


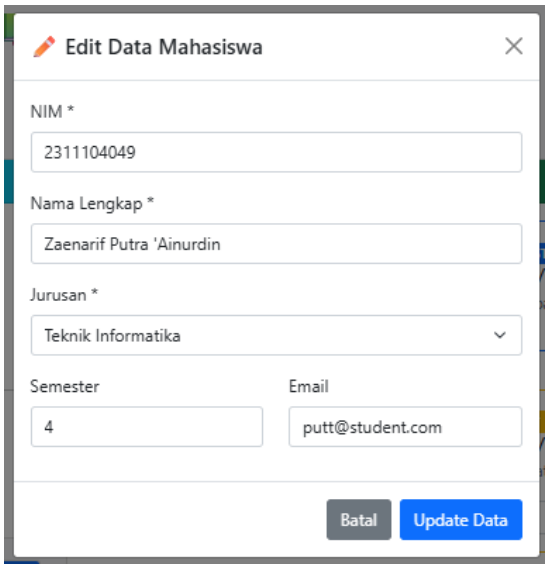
8. Testing menggunakan web GET (Melihat Data Yang Sudah Di Tambahkan), dengan menekan tombol button GET

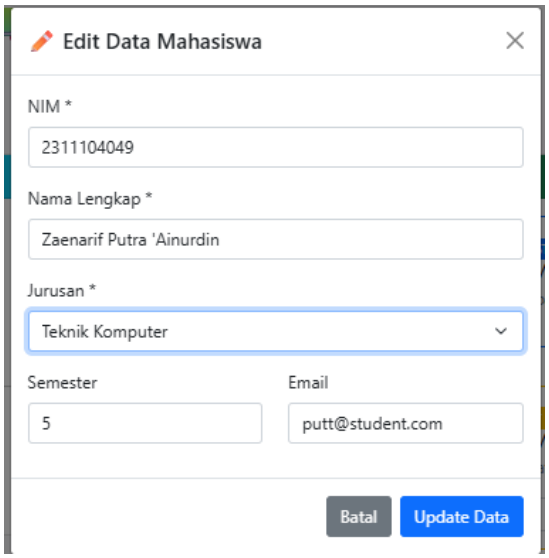


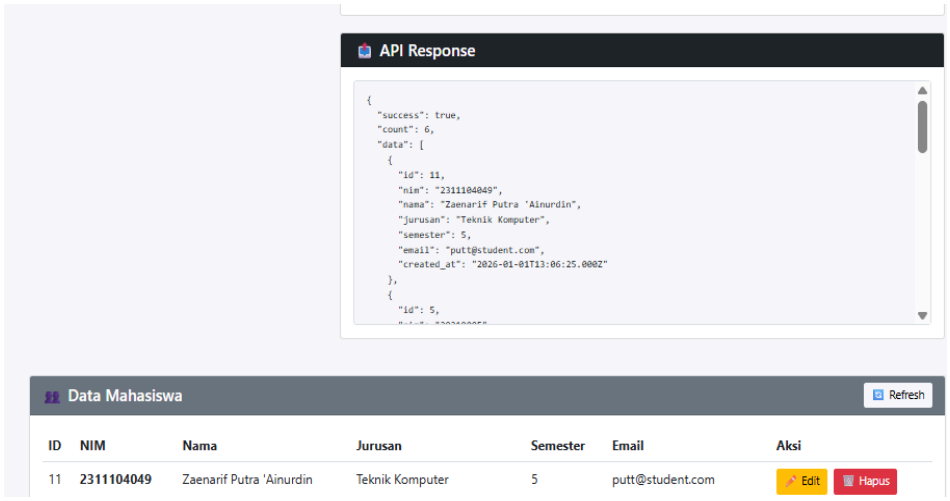

```
{
  "success": true,
  "count": 6,
  "data": [
    {
      "id": 11,
      "nim": "2311104049",
      "nama": "Zaenarif Putra 'Ainurdin",
      "jurusan": "Teknik Informatika",
      "semester": 4,
      "email": "putt@student.com",
      "created_at": "2026-01-01T13:06:25.000Z"
    },
```


9. Testing menggunakan web PUT (Mengupdate Data Yang Sebelumnya Sudah Ditambahkan Dengan Mengganti Jurusan Dan Semester dengan menggunakan id)

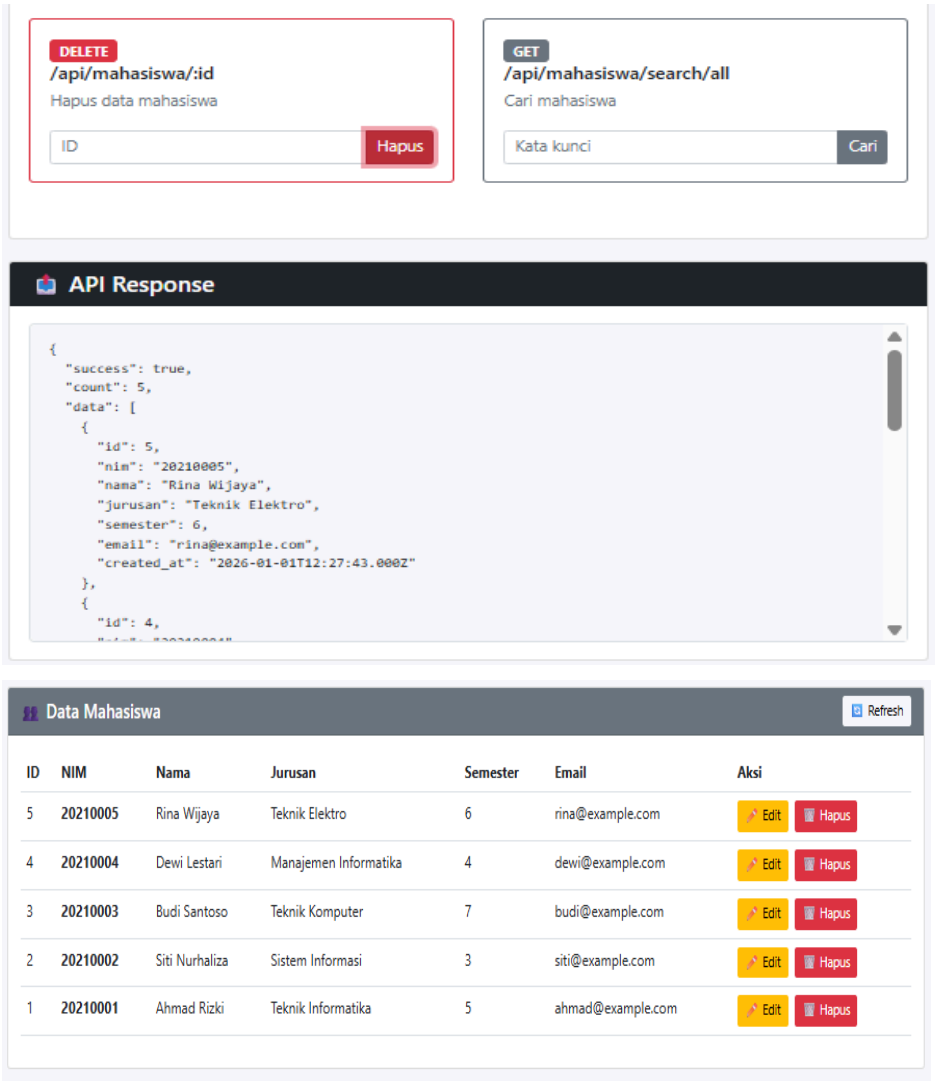



Edit Data Mahasiswa
NIM *
2311104049
Nama Lengkap *
Zaenarif Putra 'Ainurdin
Jurusan *
Teknik Informatika
Semester
4
Email
putt@student.com
Batal
Update Data


Edit Data Mahasiswa
NIM *
2311104049
Nama Lengkap *
Zaenarif Putra 'Ainurdin
Jurusan *
Teknik Komputer
Semester
5
Email
putt@student.com
Batal
Update Data


API Response
{
"success": true,
"count": 6,
"data": [
{
"id": 11,
"nim": "2311104049",
"nama": "Zaenarif Putra 'Ainurdin",
"jurusan": "Teknik Komputer",
"semester": 5,
"email": "putt@student.com",
"created_at": "2026-01-01T13:06:25.000Z"
},
{
"id": 5,
Data Mahasiswa
Refresh
ID
NIM
Nama
Jurusan
Semester
Email
Aksi
11
2311104049
Zaenarif Putra 'Ainurdin
Teknik Komputer
5
putt@student.com
Edit
Hapus

10. Testing menggunakan web DEL (Menghapus dengan menggunakan id yang sebelumnya sudah di update)



c. Deksripsi Program

Aplikasi pengelolaan data mahasiswa ini dibangun menggunakan Node.js dengan framework Express.js dan database MySQL, mengimplementasikan arsitektur Model-View-Controller (MVC) yang terstruktur. Program ini menyediakan RESTful API lengkap dengan operasi CRUD (Create, Read, Update, Delete) untuk mengelola data mahasiswa, serta antarmuka web interaktif yang memungkinkan pengguna mengakses fitur-fitur tersebut melalui browser. Alur kerja aplikasi dimulai dari request HTTP yang diterima oleh server Express.js, kemudian dialirkan melalui router ke

controller yang sesuai, controller akan memproses logika bisnis dan berinteraksi dengan model untuk mengakses database MySQL, setelah data diproses, response dikembalikan dalam format JSON untuk API atau ditampilkan sebagai halaman web dengan data real-time. Setiap komponen bekerja secara modular: app.js sebagai entry point server yang mengatur middleware dan routing, database.js meng-handle koneksi ke MySQL, mahasiswaModel.js berisi query database untuk operasi data, mahasiswaController.js mengatur logika bisnis dan response, mahasiswaRoutes.js mendefinisikan endpoint API, sedangkan frontend terdiri dari index.html untuk tampilan UI dan app.js untuk interaksi client-side dengan API. Output yang dihasilkan adalah sistem pengelolaan data yang efisien dengan dashboard real-time, tabel data mahasiswa yang bisa diperbarui secara dinamis, dan API yang dapat diuji melalui Postman, memungkinkan pengguna untuk melakukan penambahan, pengeditan, pencarian, dan penghapusan data mahasiswa dengan mudah melalui antarmuka web yang responsif.

BAB III

KESIMPULAN & SARAN

3.1 Kesimpulan

Berdasarkan pembahasan yang telah dilakukan, dapat disimpulkan bahwa Node.js merupakan solusi efektif untuk pengembangan aplikasi web modern yang membutuhkan performa tinggi dan skalabilitas. Kombinasi Node.js, Express.js, dan RESTful API memungkinkan pengembangan backend yang terstruktur, fleksibel, dan mudah dikembangkan lebih lanjut. Implementasi operasi CRUD menggunakan RESTful API juga memberikan kemudahan dalam pengelolaan data serta meningkatkan efisiensi komunikasi antara client dan server, sehingga sangat sesuai untuk pengembangan sistem informasi berbasis web.

3.2 Saran

Untuk pengembangan selanjutnya, disarankan agar aplikasi Node.js dilengkapi dengan fitur keamanan seperti autentikasi dan otorisasi guna melindungi data dari

akses yang tidak sah. Selain itu, penggunaan ORM atau query builder seperti Sequelize dapat dipertimbangkan untuk meningkatkan keamanan dan kemudahan pengelolaan database. Pengujian API secara menyeluruh serta penerapan arsitektur yang lebih modular juga dianjurkan agar aplikasi menjadi lebih stabil, mudah dipelihara, dan siap digunakan dalam skala yang lebih besar.