

**LAPORAN PRAKTIKUM**  
**PERANCANGAN DAN PEMROGRAMAN WEB**

**MODUL 13**

**(Node.js: Pengenalan)**



Oleh:

(Faishal Arif Setiawan) (2311104066)

**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.

Node.js beroperasi dalam satu *thread* utama, tetapi karena mendukung *asynchronous programming*, server dapat menangani ribuan permintaan simultan tanpa harus membuat banyak *thread* seperti dalam model multi-threaded tradisional. Hal ini membuat Node.js sangat efisien untuk aplikasi yang memerlukan operasi I/O tinggi, seperti aplikasi web real-time dan API.

Node.js didukung oleh ekosistem *package* yang sangat besar yang dikenal sebagai NPM (**Node Package Manager**). NPM memudahkan pengembang untuk mengelola dependensi proyek dan menggunakan modul yang sudah tersedia dalam komunitas. Ribuan modul yang dikembangkan oleh komunitas dapat diinstal dan digunakan dengan cepat, mempercepat pengembangan aplikasi. Node.js memiliki keunggulan unik karena digunakan oleh banyak pengembang *front-end* yang menulis JavaScript untuk browser kini dapat menulis kode di sisi server tanpa perlu mempelajari bahasa yang sama sekali berbeda.

#### B. Tujuan

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

## **BAB II**

### **HASIL**

#### **A. GUIDED (Praktikum Terbimbing)**

Implementasi CRUD:

File ini akan berfungsi sebagai penghubung antara db.js yang berisi konfigurasi koneksi *database* dan app.js yang menangani logika aplikasi. Di dalam **crud.js**, mahasiswa akan menuliskan fungsi-fungsi untuk menjalankan operasi CRUD (Create, Read, Update, Delete) yang berinteraksi langsung dengan *database*. File ini akan membantu memisahkan logika bisnis dari koneksi database, sehingga membuat kode lebih terstruktur dan mudah dikelola.

```
1 const connection = require("./db");
2
3 // Create
4 function createMahasiswa(nama, nim, jurusan, email, callback) {
5     const query =
6         "INSERT INTO mahasiswa (nama, nim, jurusan, email) VALUES (?, ?, ?, ?)";
7     connection.query(query, [nama, nim, jurusan, email], (error, results) => {
8         if (error) {
9             callback(error, null);
10        } else {
11            callback(null, results);
12        }
13    });
14 }
15
16 // Read - Get all mahasiswa
17 function getAllMahasiswa(callback) {
18     const query = "SELECT * FROM mahasiswa";
19     connection.query(query, (error, results) => {
20         if (error) {
21             callback(error, null);
22         } else {
23             callback(null, results);
24         }
25     });
26 }
27
28 // Read - Get mahasiswa by ID
29 function getMahasiswaById(id, callback) {
30     const query = "SELECT * FROM mahasiswa WHERE id = ?";
31     connection.query(query, [id], (error, results) => {
32         if (error) {
33             callback(error, null);
34         } else {
35             callback(null, results);
36         }
37     });
38 }
39
40 // Update
41 function updateMahasiswa(id, nama, nim, jurusan, email, callback) {
42     const query =
43         "UPDATE mahasiswa SET nama = ?, nim = ?, jurusan = ?, email = ? WHERE id = ?";
44     connection.query(query, [nama, nim, jurusan, email, id], (error, results) => {
45         if (error) {
46             callback(error, null);
47         } else {
48             callback(null, results);
49         }
50     });
51 }
52
53 // Delete
54 function deleteMahasiswa(id, callback) {
55     const query = "DELETE FROM mahasiswa WHERE id = ?";
56     connection.query(query, [id], (error, results) => {
57         if (error) {
58             callback(error, null);
59         } else {
60             callback(null, results);
61         }
62     });
63 }
64
65 module.exports = {
66     createMahasiswa,
67     getAllMahasiswa,
68     getMahasiswaById,
69     updateMahasiswa,
70     deleteMahasiswa
71 };
72
```

Membuat File app.js:

File ini akan menjadi pusat dari aplikasi Node.js yang mengatur semua *route* dan middleware menggunakan Express.js. Di dalam **app.js**, mahasiswa akan mendefinisikan *route* untuk operasi CRUD, mengintegrasikan logika dari crud.js, dan mengatur server untuk mendengarkan permintaan dari klien.



```
1 //app.js
2 const express = require("express");
3 const dbOperations = require("./crud");
4 const app = express();
5 const port = 3000;
6 app.use(express.json());
7 // Endpoint untuk menambahkan data
8 app.post("/mahasiswaCreate", (req, res) => {
9   const { nama, nim, jurusan, email } = req.body;
10  dbOperations.createMahasiswa(nama, nim, jurusan, email, (error) => {
11    if (error) {
12      return res.status(500).send("Error creating");
13    }
14    res.status(201).send("Mahasiswa created");
15  });
16 });
17 // Endpoint untuk mendapatkan semua data
18 app.get("/mahasiswaGet", (req, res) => {
19   dbOperations.getAllMahasiswa((error, users) => {
20     if (error) {
21       return res.status(500).send("Error fetching users");
22     }
23     res.json(users);
24   });
25 });
26 // Endpoint untuk memperbarui data
27 app.put("/mahasiswaUpdate/:id", (req, res) => {
28   const { id } = req.params;
29   const { nama, nim, jurusan, email } = req.body;
30   dbOperations.updateMahasiswa(id, nama, nim, jurusan, email, (error) => {
31     if (error) {
32       return res.status(500).send("Error updating");
33     }
34     res.send("Mahasiswa updated");
35   });
36 });
37 // Endpoint untuk menghapus data
38 app.delete("/mahasiswaDelete/:id", (req, res) => {
39   const { id } = req.params;
40   dbOperations.deleteMahasiswa(id, (error) => {
41     if (error) {
42       return res.status(500).send("Error deleting");
43     }
44     res.send("Mahasiswa deleted");
45   });
46 });
47 // Jalankan server
48 app.listen(port, () => {
49   console.log(`Server running on http://localhost:${port}`);
50 });
```

Memulai server aplikasi:

Eksekusi perintah "node app.js" ini akan memulai server aplikasi dan membuatnya siap untuk menerima permintaan dari klien.

```
PS C:\PPW_2311104066_FaishalArifSetiawan\Pertemuan 13\Praktikum\restfulAPI> node app.js
Server running on http://localhost:3000
Connected to MySQL database
```

Melakukan operasi CRUD menggunakan Postman:

Post:

The screenshot shows a Postman interface with a POST request to `http://localhost:3000/mahasiswaCreate`. The request body is set to raw JSON with the following data:

```
1 {  
2   "nama": "Vanesa Pricillia",  
3   "nim": "2311104066",  
4   "jurusan": "Software Engineering",  
5   "email": "vanesa@gmail.com"  
6 }
```

The response status is 201 Created, with a response time of 9 ms and a response size of 250 B. The response body contains the message "Mahasiswa created".

Get:

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

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

Body  raw  binary  GraphQL  JSON

```
1 {
2   "nama": "Vanesa Pricillia",
3   "nim": "2311104066",
4   "jurusan": "Software Engineering",
5   "email": "vanesa@gmail.com"
6 }
```

Send Cookies Beautify

Body Cookies Headers (7) Test Results

200 OK 11 ms 918 B

HTML Preview Visualization

```
1 [{"id":1,"nama":"Faishal Arif Setiawan","nim":"2311104066","jurusan":"Teknik Informatika","email":"faishal@gmail.com"}, {"id":2,"nama":"Ahmad Rizki","nim":"2311104001","jurusan":"Sistem Informasi","email":"ahmad.rizki@gmail.com"}, {"id":3,"nama":"Siti Nuzhaliza","nim":"2311104002","jurusan":"Teknik Informatika","email":"siti.nur@gmail.com"}, {"id":4,"nama":"Budi Santoso","nim":"2311104003","jurusan":"Sistem Informasi","email":"budi.santoso@gmail.com"}, {"id":5,"nama":"Dewi Lestari","nim":"2311104004","jurusan":"Teknik Informatika","email":"dewi.lectari@gmail.com"}, {"id":6,"nama":"Vanesa Pricillia","nim":"2311104066","jurusan":"Software Engineering","email":"vanesa@gmail.com"}]
```

## Update:

PUT <http://localhost:3000/mahasiswaUpdate/10>

Body  raw  binary  GraphQL  JSON

```
1 {
2   "nama": "Giselma Firmansyah",
3   "nim": "2311104099",
4   "jurusan": "Software Engineering",
5   "email": "Gisellma@gmail.com"
6 }
```

Send Cookies Beautify

Body Cookies Headers (7) Test Results

200 OK 6 ms 245 B

HTML Preview Visualization

```
1 Mahasiswa updated
```

## Delete:

The screenshot shows a REST client interface with the following details:

- Method: DELETE
- URL: <http://localhost:3000/mahasiswaDelete/16>
- Headers: (13)
- Body: (empty)
- Query Params:

Key	Value	Description
Key	Value	Description

- Cookies: (empty)
- Status: 200 OK
- Time: 8 ms
- Size: 245 B
- Body content: 1 Mahasiswa deleted

## B.Unguided

### 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.

**Catatan:** Sertakan penjelasan tahapan pelaksanaan aplikasi secara sistematis.

### Jawaban:

Buat View Public/index.html:

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Aplikasi Pengelolaan Data Mahasiswa</title>
7     <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10    <div class="container">
11        <!-- Header -->
12        <header>
13            <h1> Sistem Pengelolaan Data Mahasiswa</h1>
14            <p>Aplikasi berbasis Node.js, Express, dan MySQL</p>
15        </header>
16
17        <!-- Form Input -->
18        <div class="form-container">
19            <h2 id="form-title">Tambah Mahasiswa Baru</h2>
20            <form id="mahasiswaForm">
21                <input type="hidden" id="mahasiswaId">
22
23                <div class="form-group">
24                    <label for="nama">Nama Lengkap:</label>
25                    <input type="text" id="nama" placeholder="Masukkan nama lengkap" required>
26                </div>
27
28                <div class="form-group">
29                    <label for="nim">NIM:</label>
30                    <input type="text" id="nim" placeholder="Masukkan NIM" required>
31                </div>
32
33                <div class="form-group">
34                    <label for="jurusan">Jurusan:</label>
35                    <select id="jurusan" required>
36                        <option value="">-- Pilih Jurusan --</option>
37                        <option value="Teknik Informatika">Teknik Informatika</option>
38                        <option value="Sistem Informasi">Sistem Informasi</option>
39                        <option value="Teknik Komputer">Teknik Komputer</option>
40                        <option value="Sistem Komputer">Sistem Komputer</option>
41                    </select>
42                </div>
43
44                <div class="form-group">
45                    <label for="email">Email:</label>
46                    <input type="email" id="email" placeholder="Masukkan email" required>
47                </div>
48
49                <div class="form-buttons">
50                    <button type="submit" class="btn btn-primary" id="submitBtn">
51                        + Tambah Mahasiswa
52                    </button>
53                    <button type="button" class="btn btn-secondary" id="cancelBtn" style="display: none;">
54                        X Batal
55                    </button>
56                </div>
57            </form>
58        </div>
59
60        <!-- Tabel Data Mahasiswa -->
61        <div class="table-container">
62            <div class="table-header">
63                <h2> Daftar Mahasiswa</h2>
64                <button class="btn btn-refresh" onclick="loadMahasiswa()> Refresh</button>
65            </div>
66
67            <div class="table-wrapper">
68                <table id="mahasiswaTable">
69                    <thead>
70                        <tr>
71                            <th>No</th>
72                            <th>Nama</th>
73                            <th>NIM</th>
74                            <th>Jurusan</th>
75                            <th>Email</th>
76                            <th>Aksi</th>
77                        </tr>
78                    </thead>
79                    <tbody id="mahasiswaTableBody">
80                        <tr>
81                            <td colspan="6" class="loading">Memuat data...</td>
82                        </tr>
83                    </tbody>
84                </table>
85            </div>
86        </div>
87
88        <!-- Footer -->
89        <footer>
90            <p>© 2026 Faishal Arif Setiawan - 2311104066 | Pertemuan 13</p>
91        </footer>
92    </div>
93
94    <!-- Toast Notification -->
95    <div id="toast"></div>
96
97    <script src="script.js"></script>
98 </body>
99 </html>
100
```

Public/style.css:

```
 1  // Copyright 2014 The Go Authors. All rights reserved.
 2  // Use of this source code is governed by a BSD-style
 3  // license that can be found in the LICENSE file.
 4
 5  package main
 6
 7  import (
 8      "fmt"
 9      "log"
10      "os"
11      "path/filepath"
12      "sync"
13      "time"
14      "unsafe"
15  )
16
17  const (
18      maxDepth = 10
19      maxTime  = 10 * time.Second
20  )
21
22  type tree struct {
23      name     string
24      children map[string]*tree
25      parent   *tree
26  }
27
28  func (t *tree) add(c *tree) {
29      t.children[c.name] = c
30      c.parent = t
31  }
32
33  func (t *tree) walk(f func(string)) {
34      f(t.name)
35      for _, v := range t.children {
36          v.walk(f)
37      }
38  }
39
40  func (t *tree) print() {
41      if t.parent != nil {
42          t.parent.print()
43      }
44      log.Println(t.name)
45  }
46
47  func (t *tree) printDepth(d int) {
48      if d > maxDepth {
49          return
50      }
51      if t.parent != nil {
52          t.parent.printDepth(d + 1)
53      }
54      log.Println(t.name)
55  }
56
57  func (t *tree) printTime(d time.Duration) {
58      if d > maxTime {
59          return
60      }
61      if t.parent != nil {
62          t.parent.printTime(d + time.Second)
63      }
64      log.Println(t.name)
65  }
66
67  func (t *tree) printDepthTime(d int, dTime time.Duration) {
68      if d > maxDepth || dTime > maxTime {
69          return
70      }
71      if t.parent != nil {
72          t.parent.printDepthTime(d + 1, dTime+time.Second)
73      }
74      log.Println(t.name)
75  }
76
77  func (t *tree) printDepthTimeDepth(d int, dTime time.Duration, dDepth int) {
78      if d > maxDepth || dTime > maxTime {
79          return
80      }
81      if t.parent != nil {
82          t.parent.printDepthTimeDepth(d + 1, dTime+time.Second, dDepth + 1)
83      }
84      log.Println(t.name)
85  }
86
87  func (t *tree) printDepthTimeDepthDepth(d int, dTime time.Duration, dDepth int, dDepthTime time.Duration) {
88      if d > maxDepth || dTime > maxTime {
89          return
90      }
91      if t.parent != nil {
92          t.parent.printDepthTimeDepthDepth(d + 1, dTime+time.Second, dDepth + 1, dDepthTime+time.Second)
93      }
94      log.Println(t.name)
95  }
96
97  func (t *tree) printDepthTimeDepthDepthDepth(d int, dTime time.Duration, dDepth int, dDepthTime time.Duration, dDepthDepth int) {
98      if d > maxDepth || dTime > maxTime {
99          return
100     }
101     if t.parent != nil {
102         t.parent.printDepthTimeDepthDepthDepth(d + 1, dTime+time.Second, dDepth + 1, dDepthTime+time.Second, dDepthDepth + 1)
103     }
104     log.Println(t.name)
105 }
```

Buat File db.js

```
const mysql = require('mysql2');

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

connection.connect((err) => {
  if (err) {
    console.error('✖ Error connecting to database:', err.message);
    return;
  }
  console.log('✓ Connected to MySQL database: akademik_latihan');
});

module.exports = connection;
```

Buat file crud.js

```
const connection = require('./db');

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) {
      callback(error, null);
    } else {
      callback(null, results);
    }
  });
}

function getAllMahasiswa(callback) {
  const query = 'SELECT * FROM mahasiswa ORDER BY id DESC';
  connection.query(query, (error, results) => {
    if (error) {
      callback(error, null);
    } else {
      callback(null, results);
    }
  });
}
```

```
});  
}  
  
function getMahasiswaById(id, callback) {  
    const query = 'SELECT * FROM mahasiswa WHERE id = ?';  
    connection.query(query, [id], (error, results) => {  
        if (error) {  
            callback(error, null);  
        } else {  
            callback(null, results[0]);  
        }  
    });
}  
  
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) {  
            callback(error, null);  
        } else {  
            callback(null, results);  
        }  
    });
}  
  
function deleteMahasiswa(id, callback) {  
    const query = 'DELETE FROM mahasiswa WHERE id = ?';  
    connection.query(query, [id], (error, results) => {  
        if (error) {  
            callback(error, null);  
        } else {  
            callback(null, results);  
        }  
    });
}  
  
module.exports = {  
    createMahasiswa,  
    getAllMahasiswa,  
    getMahasiswaById,  
    updateMahasiswa,  
    deleteMahasiswa  
};
```

Buat file app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const path = require('path');
const dbOperations = require('./crud');

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

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

app.use(express.static(path.join(__dirname, 'public')));

app.get('/api/mahasiswa', (req, res) => {
  dbOperations.getAllMahasiswa((error, results) => {
    if (error) {
      return res.status(500).json({
        success: false,
        message: 'Error fetching data',
        error: error.message
      });
    }
    res.json({
      success: true,
      data: results
    });
  });
});

app.get('/api/mahasiswa/:id', (req, res) => {
  const { id } = req.params;
  dbOperations.getMahasiswaById(id, (error, result) => {
    if (error) {
      return res.status(500).json({
        success: false,
        message: 'Error fetching data',
        error: error.message
      });
    }
    if (!result) {
      return res.status(404).json({
        success: false,
        message: 'Mahasiswa tidak ditemukan'
      });
    }
  });
});
```

```
    });
}

res.json({
  success: true,
  data: result
});

});

});

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

  // Validasi input
  if (!nama || !nim || !jurusan || !email) {
    return res.status(400).json({
      success: false,
      message: 'Semua field harus diisi!'
    });
  }

  dbOperations.createMahasiswa(nama, nim, jurusan, email, (error, result) => {
    if (error) {
      return res.status(500).json({
        success: false,
        message: 'Error creating data',
        error: error.message
      });
    }
    res.status(201).json({
      success: true,
      message: 'Mahasiswa berhasil ditambahkan',
      id: result.insertId
    });
  });
});

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

  if (!nama || !nim || !jurusan || !email) {
    return res.status(400).json({
      success: false,
      message: 'Semua field harus diisi!'
    });
  }
})
```

```
dbOperations.updateMahasiswa(id, nama, nim, jurusan, email, (error, result) => {
  if (error) {
    return res.status(500).json({
      success: false,
      message: 'Error updating data',
      error: error.message
    });
  }
  if (result.affectedRows === 0) {
    return res.status(404).json({
      success: false,
      message: 'Mahasiswa tidak ditemukan'
    });
  }
  res.json({
    success: true,
    message: 'Data mahasiswa berhasil diupdate'
  });
});

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

  dbOperations.deleteMahasiswa(id, (error, result) => {
    if (error) {
      return res.status(500).json({
        success: false,
        message: 'Error deleting data',
        error: error.message
      });
    }
    if (result.affectedRows === 0) {
      return res.status(404).json({
        success: false,
        message: 'Mahasiswa tidak ditemukan'
      });
    }
    res.json({
      success: true,
      message: 'Data mahasiswa berhasil dihapus'
    });
  });
});

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});
```

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

Output:



## Daftar Mahasiswa

Refresh

No	Nama	NIM	Jurusan
----	------	-----	---------

1	Vanesa Pricillya	231104090	Teknik Komputer
---	------------------	-----------	-----------------

2	Dewi Lestari	2311104004	Teknik Informatika
---	--------------	------------	--------------------

3	Budi Santoso	2311104003	Sistem Informasi
---	--------------	------------	------------------

4	Siti Nurhaliza	2311104002	Teknik Informatika
---	----------------	------------	--------------------

## Tambah Mahasiswa Baru

Nama Lengkap:

NIM:

Jurusan:

Email:

 Tambah Mahasiswa

Penjelasan:

Aplikasi ini adalah sistem untuk mengelola data mahasiswa secara digital yang bisa diakses melalui browser web. Aplikasi ini memungkinkan kita untuk menambah data mahasiswa baru, melihat daftar semua mahasiswa, mengubah data yang sudah ada, dan menghapus data mahasiswa yang tidak diperlukan lagi. Semua data mahasiswa tersimpan dalam database MySQL, sehingga data tetap aman dan tidak

hilang meskipun aplikasi ditutup. Aplikasi ini dibangun menggunakan teknologi Node.js untuk backend dan HTML/CSS/JavaScript untuk frontend, sehingga user bisa berinteraksi dengan data melalui tampilan web yang menarik dan mudah digunakan.

Aplikasi ini bekerja dengan konsep client-server, dimana browser sebagai client berkomunikasi dengan server yang dijalankan di komputer kita. Ketika kita membuka halaman web di browser, server akan mengirimkan file HTML, CSS, dan JavaScript yang membentuk tampilan halaman. Setelah halaman terbuka, setiap kali kita melakukan aksi seperti mengklik tombol "Tambah Mahasiswa" atau "Edit", JavaScript di browser akan mengirim request ke server melalui API. Server kemudian memproses request tersebut, berkomunikasi dengan database MySQL untuk menyimpan atau mengambil data, lalu mengirim response balik ke browser. Browser menerima response tersebut dan menampilkan hasilnya kepada user, misalnya menampilkan notifikasi "Data berhasil ditambahkan" atau memperbarui tabel data mahasiswa.

## BAB III

### PENUTUP

#### A. Kesimpulan

Pada praktikum Pertemuan 13 ini, Dengan menggunakan RESTful API berbasis Node.js, mahasiswa dapat membangun aplikasi web yang efisien dan scalable. RESTful API memungkinkan komunikasi yang fleksibel antara klien dan server melalui protokol HTTP, serta mendukung operasi CRUD untuk pengelolaan data. Node.js, dengan keunggulannya dalam penanganan asynchronous dan event-driven, memungkinkan server untuk menangani banyak permintaan secara bersamaan tanpa memerlukan banyak sumber daya. Kombinasi RESTful API dan Node.js ini memberikan fondasi yang kuat untuk pengembangan aplikasi

web modern yang responsif, mudah diintegrasikan, dan dapat dikembangkan lebih lanjut sesuai kebutuhan.