MODUL PERTEMUAN 6

PRAKTIKUM FULLSTACK DEVELOPMENT

“Pembuatan Backend Express dan Database MongoDB”

SEKOLAH TINGGI ILMU KOMPUTER PGRI BANYUWANGI

STIKOM BANYUWANGI

2024

A. Pendahuluan

Untuk modul kali ini kita kan membahas backend dengan menggunakan express.js dan database noSQL MongoDB sebagai tempat penyimpanannya. Modul ini bertujuan untuk memberikan pemahaman dasar tentang cara membuat backend menggunakan Express dan menghubungkannya dengan database MongoDB. Peserta diharapkan dapat melakukan operasi CRUD (Create, Read, Update, Delete) pada dokumen `mahasiswa` dalam koleksi `siakad` yang digunakan dalam contoh dibawah dan mengembangkan lebih lanjut.

B. Persiapan/Prasyarat

1. **Node.js dan npm** sudah terinstal di komputer Anda.
2. **MongoDB server** sudah terinstal dan berjalan di komputer Anda.
3. **MongoDB Compass** untuk mempermudah proses pembuatan database.
4. **Text editor** (misalnya VSCode) atau IDE yang mendukung JavaScript.
5. Komputer dengan sistem operasi Windows, MacOS, atau Linux.
6. Koneksi internet untuk mengunduh dependensi.

C. Langkah-langkah Praktikum

1. Bagian 1: Persiapan Lingkungan

a) Instalasi Node.js dan npm

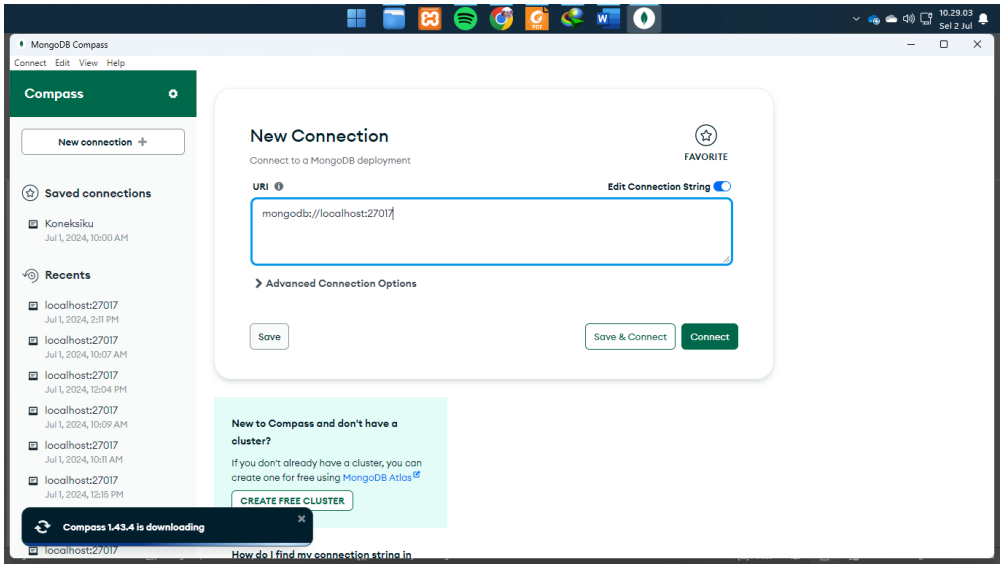
Pastikan Node.js dan npm telah terinstal. Anda dapat memeriksanya dengan perintah berikut:

Perintah pada CMD

```
node -v
npm -v
```

b) Instalasi MongoDB dan MongoDBCompass

Untuk mengecek apakah MongoDB sudah terinstall dapat melakukan koneksi MongoDBCompass ke server MongoDB dengan Alamat : mongodb://localhost:27017 kemudian klik connect



Jika sudah terkoneksi maka sudah siap untuk digunakan

2. **Bagian 2: Pembuatan Koleksi, Dokumen dan kolom**

a) Json yang digunakan sebagai acuan untuk pembuatan table adalah sebagai berikut :

json

```
       {
         "mahasiswa": [
           {
             "nim": "1122101010",
             "nama": "Ahmad Budi",
             "jurusan": "Teknik Informatika",
             "angkatan": 2020,
             "email": "ahmad.budi@example.com"
           },
           {
             "nim": "1122101011",
             "nama": "Siti Aminah",
             "jurusan": "Teknik Informatika",
             "angkatan": 2020,
             "email": "siti.aminah@example.com"
           },
           {
             "nim": "1122101012",
             "nama": "Rudi Hartono",
             "jurusan": "Teknik Informatika",
             "angkatan": 2020,
             "email": "rudi.hartono@example.com"
           },
           {
             "nim": "1122101013",
             "nama": "Dewi Sartika",
             "jurusan": "Teknik Informatika",
             "angkatan": 2020,
             "email": "dewi.sartika@example.com"
           }
         ]
}
```

b) Maka jika dibuat dalam bentuk noSQL MongoDB akan seperti berikut :

Transact SQl

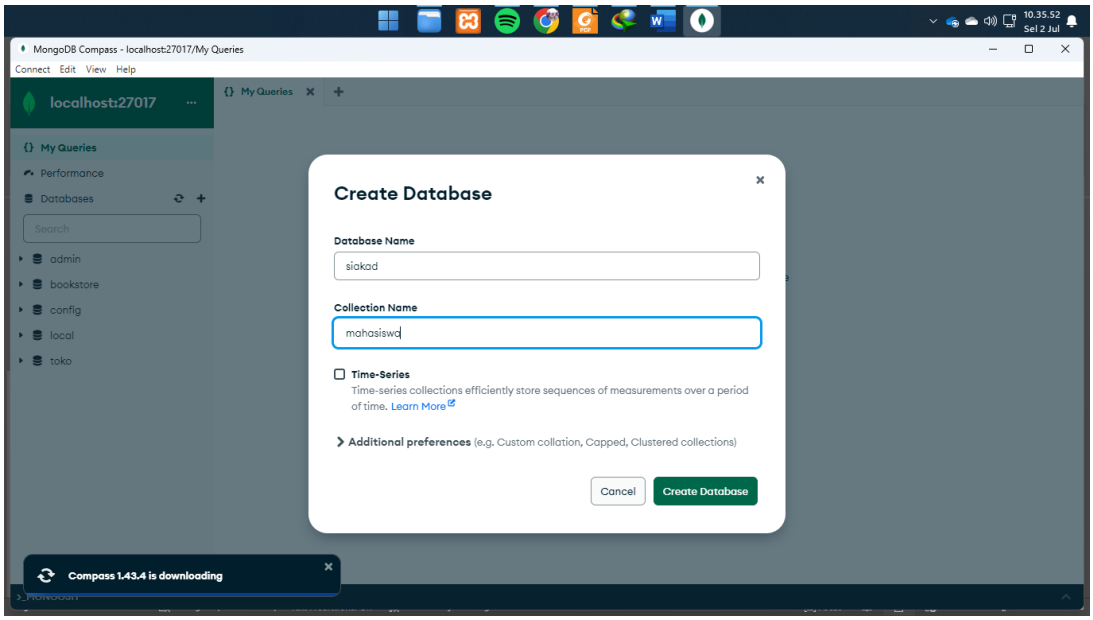
```
{
  "_id": "1122101010",
  "nama": "Ahmad Budi",
  "jurusan": "Teknik Informatika",
  "angkatan": 2020,
  "email": "ahmad.budi@example.com"
}

Nb :
Nim dapat menjadi ObjectId atau dapat pula di-set manual sebagai NIM
```

Keterangan :

1) _id: Digunakan sebagai identifier unik untuk setiap dokumen. Dalam hal ini, kita menggunakan NIM mahasiswa sebagai _id.
2) nama: Nama lengkap mahasiswa.
3) jurusan: Jurusan yang diambil oleh mahasiswa.
4) angkatan: Tahun angkatan masuk mahasiswa.
5) email: Alamat email mahasiswa.

c) Implementasi pembuatan koleksi, dokumen dan kolom

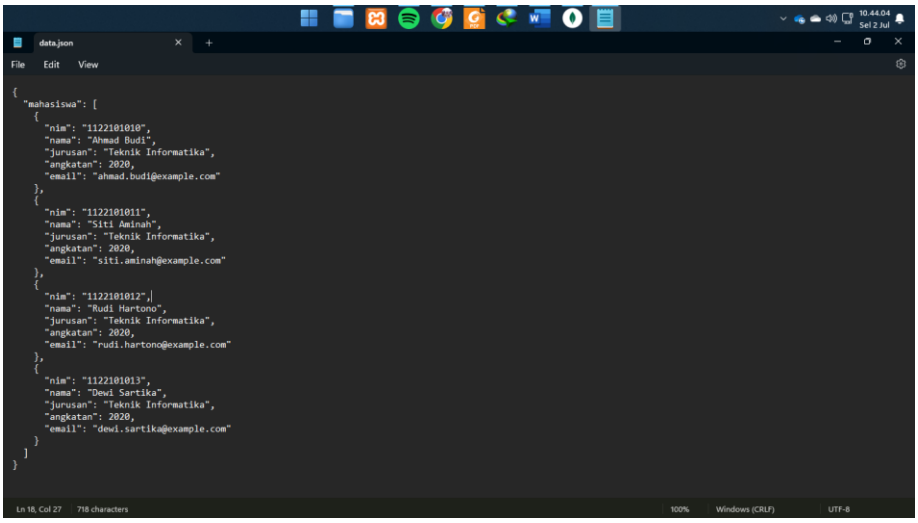
Buka MongoDBCompass pastikan sudah terkoneksi dengan baik dan klik tombol tambah database pada sisi kiri :



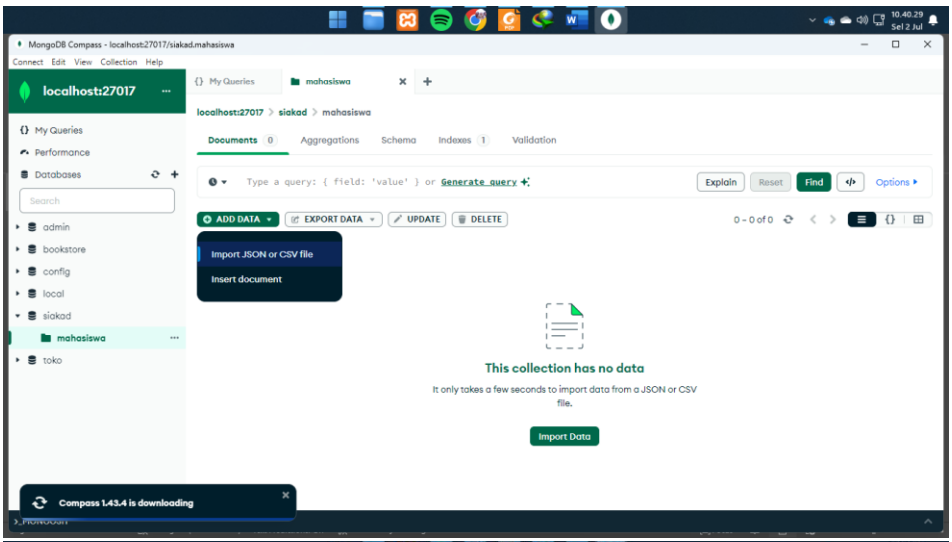
Struktur
Database name : siakad
Collection name : mahasiswa

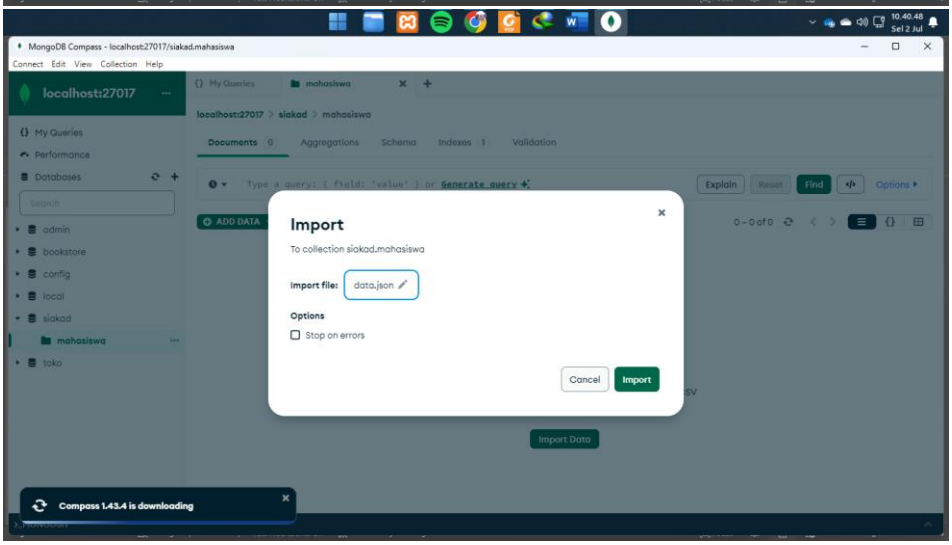
Ada dua cara yang disediakan MongoDB untuk membuat dokumen dan kolom yang disedikan oleh MongoDB yaitu import json/csv file dan insert dokumen seperti berikut :

1) Untuk cara pertama maka kalian harus mempersiapkan file berextensi .json yang isinya json yang akan dimasukkan ke mongoDB seperti dalam gambar berikut :

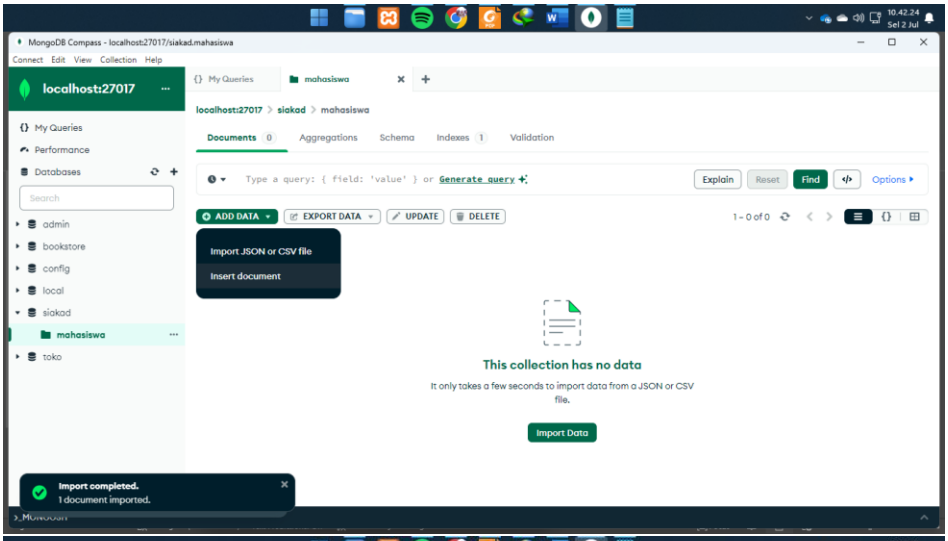


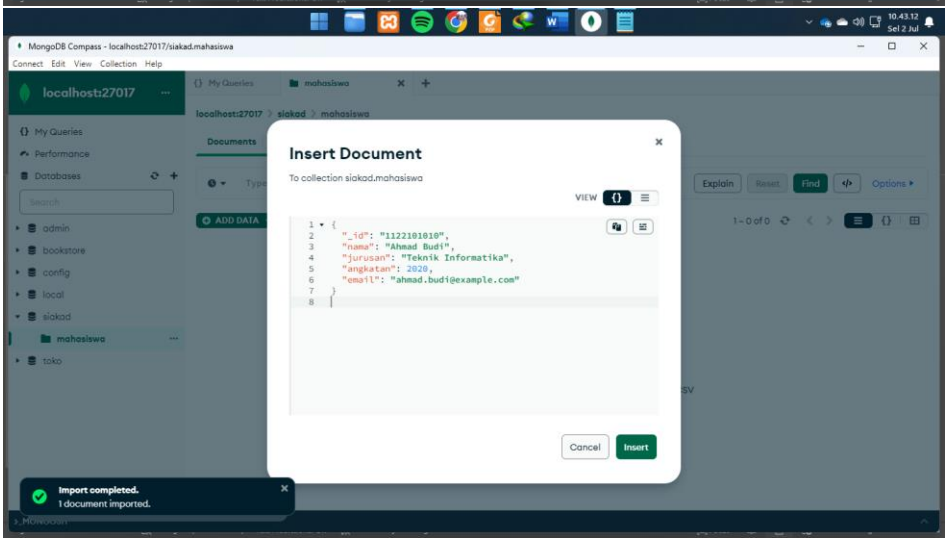
Kemuadian pada MongoDBCompass klik add data klik import json or csv file ➔ cari file json yang sudah diapkan dan klik import.

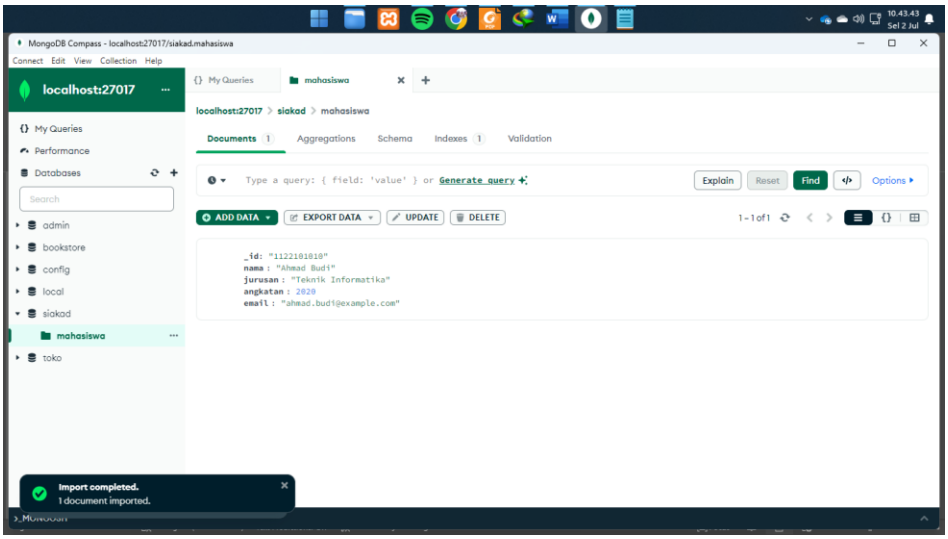




2) Untuk cara kedua klik add data klik insert document kemudian masukkan json yang ada baik satu persatu atau langsung kesemuanya. Dalam gambar saya berikan satu persatu sehingga prosesnya akan berulang.





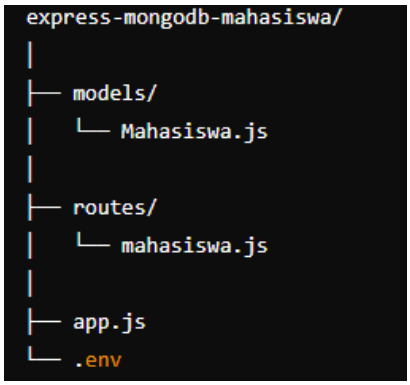


Jika kita menggunakan mongose dengan terminal maka perintahnya adalah sebagai berikut :

Perintah MongoDB dengan terminal

```
use siakad;

db.mahasiswa.insertMany([
  {
    _id: "1122101010",
    nama: "Ahmad Budi",
    jurusan: "Teknik Informatika",
    angkatan: 2020,
    email: "ahmad.budi@example.com"
  },
  {
    _id: "1122101011",
    nama: "Siti Aminah",
    jurusan: "Teknik Informatika",
    angkatan: 2020,
    email: "siti.aminah@example.com"
  },
  {
    _id: "1122101012",
    nama: "Rudi Hartono",
    jurusan: "Teknik Informatika",
    angkatan: 2020,
    email: "rudi.hartono@example.com"
  },
  {
    _id: "1122101013",
    nama: "Dewi Sartika",
    jurusan: "Teknik Informatika",
    angkatan: 2020,
    email: "dewi.sartika@example.com"
  }
]);
```

3. **Bagian 3: Pembuatan Proyek Express**
   a) Membuat direktori proyek
   b) Inisialisasi proyek Node.js dengan perintah "**npm init -y**"
   c) Instalasi dependensi "***npm install express mongoose body-parser***"
   d) Buatlah struktur project seperti berikut :



4. **Bagian 4: Implementasi Backend dengan Express**
   a) Buka file .env tambahkan kode berikut :

.env
```
MONGO_URI=mongodb://localhost:27017/siakad
PORT=3000
```

   b) Buka file `models/Mahasiswa.js` tambahkan kode berikut:

models/Mahasiswa.js
```
const mongoose = require('mongoose');

const mahasiswaSchema = new mongoose.Schema({
  _id: String,
  nama: { type: String, required: true },
  jurusan: { type: String, required: true },
  angkatan: { type: Number, required: true },
  email: { type: String, required: true }
});

module.exports = mongoose.model('Mahasiswa', mahasiswaSchema);
// DELETE: Hapus data mahasiswa
app.delete('/mahasiswa/:nim', (req, res) => {
     const { nim } = req.params;
     const sql = 'DELETE FROM mahasiswa WHERE nim = ?';
     db.query(sql, [nim], (err, result) => {
       if (err) {
         return res.status(500).json({ error: err.message });
       }
       if (result.affectedRows === 0) {
         return res.status(404).json({ message: 'Mahasiswa not
found' });
       }
       res.status(200).json({ message: 'Mahasiswa deleted
successfully' });
     });
});
```

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

c) Buka file **"routes/mahasiswa.js"** tambahkan kode berikut :

routes/mahasiswa.js

```
const express = require('express');
const router = express.Router();
const Mahasiswa = require('../models/Mahasiswa');

      // GET semua mahasiswa
      router.get('/', async (req, res) => {
        try {
          const mahasiswa = await Mahasiswa.find();
          res.json(mahasiswa);
        } catch (err) {
          res.status(500).json({ message: err.message });
        }
      });

      // GET satu mahasiswa
      router.get('/:id', getMahasiswa, (req, res) => {
        res.json(res.mahasiswa);
      });

      // POST mahasiswa baru
      router.post('/', async (req, res) => {
        const mahasiswa = new Mahasiswa({
          _id: req.body.nim,
          nama: req.body.nama,
          jurusan: req.body.jurusan,
          angkatan: req.body.angkatan,
          email: req.body.email
        });

        try {
          const newMahasiswa = await mahasiswa.save();
          res.status(201).json(newMahasiswa);
        } catch (err) {
          res.status(400).json({ message: err.message });
        }
      });

      // Middleware untuk mendapatkan mahasiswa berdasarkan ID
      async function getMahasiswa(req, res, next) {
        let mahasiswa;
        try {
          mahasiswa = await Mahasiswa.findById(req.params.id);
          if (mahasiswa == null) {
            return res.status(404).json({ message: 'Mahasiswa tidak
      ditemukan' });
          }
        } catch (err) {
```

```
            return res.status(500).json({ message: err.message });
          }

          res.mahasiswa = mahasiswa;
          next();
        }

module.exports = router;
```

d) Buka file **"app.js"** tambahkan kode berikut :

Perintah pada CMD

```
require('dotenv').config();

      const express = require('express');
      const mongoose = require('mongoose');
      const bodyParser = require('body-parser');

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

      mongoose.connect(process.env.MONGO_URI, {
        useNewUrlParser: true,
        useUnifiedTopology: true
      });

      const db = mongoose.connection;
      db.on('error', (error) => console.error(error));
      db.once('open', () => console.log('Terhubung ke MongoDB'));

      app.use(bodyParser.json());

      const mahasiswaRouter = require('./routes/mahasiswa');
      app.use('/mahasiswa', mahasiswaRouter);

app.listen(PORT, () => console.log(`Server berjalan di port
${PORT}`));
```

e) Jalankan server

Perintah pada CMD

```
node app.js
```

Nb:

- Untuk pengguna yang sudah menginstall nodemon dapat menjalankan dengan perintah : "***nodemon index.js***"
- Jangan lupa untuk cek database mongodb agar aplikasi dapat mengakses data yang ada dalam database.

5. Bagian 5: Pengujian API

a) Menggunakan Postman

Skenario

✓ **Tambah data mahasiswa (POST)**

- Endpoint: `POST http://localhost:3000/mahasiswa`

- Body (JSON):
- Inputkan json berikut :

```json
{
  "nim": "12345678",
  "nama": "Ahmad Budi",
  "jurusan": "Teknik Informatika",
  "angkatan": 2020,
  "email": "ahmad.budi@example.com"
}
```

✓ Ambil semua data mahasiswa (GET)
- Endpoint: `GET http://localhost:3000/mahasiswa`

✓ Ambil data mahasiswa berdasarkan NIM (GET)
- Endpoint: `GET http://localhost:3000/mahasiswa/id`

✓ Ubah data mahasiswa (PUT)
- Endpoint: `PUT http://localhost:3000/mahasiswa/id`
- Body (JSON):

```json
{
  "nama": "Ahmad Budi Santoso",
  "jurusan": "Teknik Informatika",
  "angkatan": 2020,
  "email": "ahmad.budi.santoso@example.com"
}
```

✓ Hapus data mahasiswa (DELETE)
- Endpoint: `DELETE http://localhost:3000/mahasiswa/id`

6. Praktikum

a. Buatlah koleksi baru dengan nama matakuliah dengan json sebagai berikut :

```json
{
  "kodeMK": "KK010105",
  "namaMK": "Praktikum Fullstack Development",
  "sks": 1,
  "semester": "4"
}
```

b. Buatkan kode untuk melakukan proses CRUD pada data diatas dibackend yang sudah dibuat.

c. Buatkan route untuk mengakses halaman tersebut.

d. Lakukan percobaan dengan postman berikan scenario seperti dalam modul diatas.

e. Buatlah laporan praktikumnya