

LAPORAN PRAKTIKUM

FULLSTACK DEVELOPMENT

“Pembuatan Backend Express Dan Database”

Disusun untuk Memenuhi Matakuliah Praktikum fullstack Development

Dibimbing oleh Bapak M. Taufiq



Oleh:

Lucky Candra Permana

1122102032

PROGRAM STUDI S1 TEKNIK INFORMATIKA
SEKOLAH TINGGI ILMU KOMPUTER PGRI BANYUWANGI
2022

FULLSTACK DEVELOPMENT

Tujuan

Setelah mempelajari bab ini diharapkan mahasiswa akan mampu :

1. mempraktekkan pembuatan json yang dikonfersikan ke dalam tabel, dokumen, dan kolom koleksi;
2. mempraktekkan pembuatan database, koleksi, dokumen, dan kolom pada mongodb
3. melakukan pengujian backend yang dibuat dengan menggunakan express.js dan mengoneksikan ke mongodb.

DASAR TEORI

Jelaskan dasar teori yang digunakan untuk materi ini:

1. Node.js

Node.js adalah platform JavaScript yang dibangun di atas mesin JavaScript V8 dari Chrome untuk menjalankan JavaScript di sisi server. Ini memungkinkan pengembang untuk menulis kode JavaScript di sisi server dan mengelola logika aplikasi serta interaksi dengan database.

2. Express

Express adalah framework aplikasi web yang berjalan di atas Node.js, menyediakan fitur-fitur yang mempermudah pengembangan aplikasi web seperti routing, middleware, dan pengelolaan permintaan dan respons HTTP. Dengan Express, Anda dapat dengan mudah membuat endpoint API RESTful untuk mengakses dan memanipulasi data.

3. MongoDB

MongoDB adalah basis data NoSQL yang berorientasi dokumen, yang berarti data disimpan dalam dokumen JSON yang fleksibel. MongoDB sangat cocok untuk pengembangan aplikasi yang memerlukan skema data yang dinamis dan sering berubah.

4. Mongoose

Mongoose adalah pustaka ODM (Object Data Modeling) untuk MongoDB dan Node.js, yang menyediakan lingkungan yang kuat untuk mendefinisikan skema, memvalidasi data, dan melakukan operasi CRUD terhadap basis data MongoDB menggunakan JavaScript.

Latihan Praktikum 1
Pemrograman Berbasis Jaringan

Nama Program : Menampilkan kalimat

Bahasa Pemrograman :

Compiler : node js

Script program :

Pada app.js

```
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;

// Middleware
app.use(bodyParser.json());
```

```

// Setup koneksi ke MongoDB menggunakan Mongoose
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'));

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

// Gunakan router untuk matakuliah
const matakuliahRouter = require('./routes/matakuliah');
app.use('/matakuliah', matakuliahRouter);

// Middleware untuk menangani error jika endpoint tidak ditemukan
app.use((req, res, next) => {
  const error = new Error('Endpoint tidak ditemukan');
  error.status = 404;
  next(error);
});

// Middleware untuk menangani error lainnya
app.use((err, req, res, next) => {
  res.status(err.status || 500);
  res.json({
    error: {
      message: err.message
    }
  });
});

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

```

Pada folder models/Matakuliah.js

```

const mongoose = require('mongoose');

const matakuliahSchema = new mongoose.Schema({
  kodeMK: { type: String, required: true },
  namaMK: { type: String, required: true },
  sks: { type: Number, required: true },
  semester: { type: String, required: true }
});

```

```
// Export model Matakuliah
module.exports = mongoose.model('Matakuliah', matakuliahSchema);
```

pada folder routes/mahasiswa.js

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

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

// GET satu matakuliah berdasarkan ID
router.get('/:id', getMatakuliah, (req, res) => {
  res.json(res.matakuliah);
});

// POST matakuliah baru
router.post('/', async (req, res) => {
  const matakuliah = new Matakuliah({
    kodeMK: req.body.kodeMK,
    namaMK: req.body.namaMK,
    sks: req.body.sks,
    semester: req.body.semester
  });

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

// PUT (update) matakuliah berdasarkan ID
router.put('/:id', getMatakuliah, async (req, res) => {
  if (req.body.kodeMK !== null) {
    res.matakuliah.kodeMK = req.body.kodeMK;
  }
  if (req.body.namaMK !== null) {
    res.matakuliah.namaMK = req.body.namaMK;
  }
  if (req.body.sks !== null) {

```

```

    res.matakuliah.sks = req.body.sks;
  }
  if (req.body.semester !== null) {
    res.matakuliah.semester = req.body.semester;
  }

  try {
    const updatedMatakuliah = await res.matakuliah.save();
    res.json(updatedMatakuliah);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// DELETE matakuliah berdasarkan ID
router.delete('/:id', getMatakuliah, async (req, res) => {
  try {
    await res.matakuliah.remove();
    res.json({ message: 'Matakuliah berhasil dihapus' });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

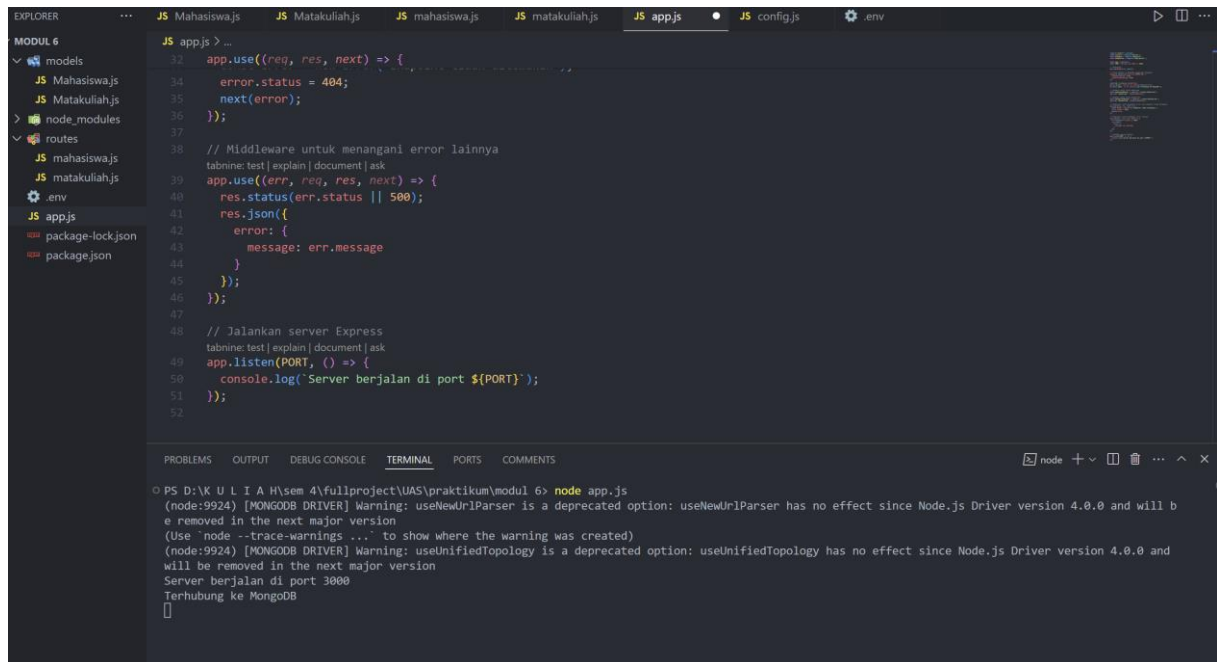
// Middleware untuk mendapatkan matakuliah berdasarkan ID
async function getMatakuliah(req, res, next) {
  let matakuliah;
  try {
    matakuliah = await Matakuliah.findById(req.params.id);
    if (matakuliah == null) {
      return res.status(404).json({ message: 'Matakuliah tidak ditemukan' });
    }
  } catch (err) {
    return res.status(500).json({ message: err.message });
  }

  res.matakuliah = matakuliah;
  next();
}

module.exports = router;

```

Output Program :

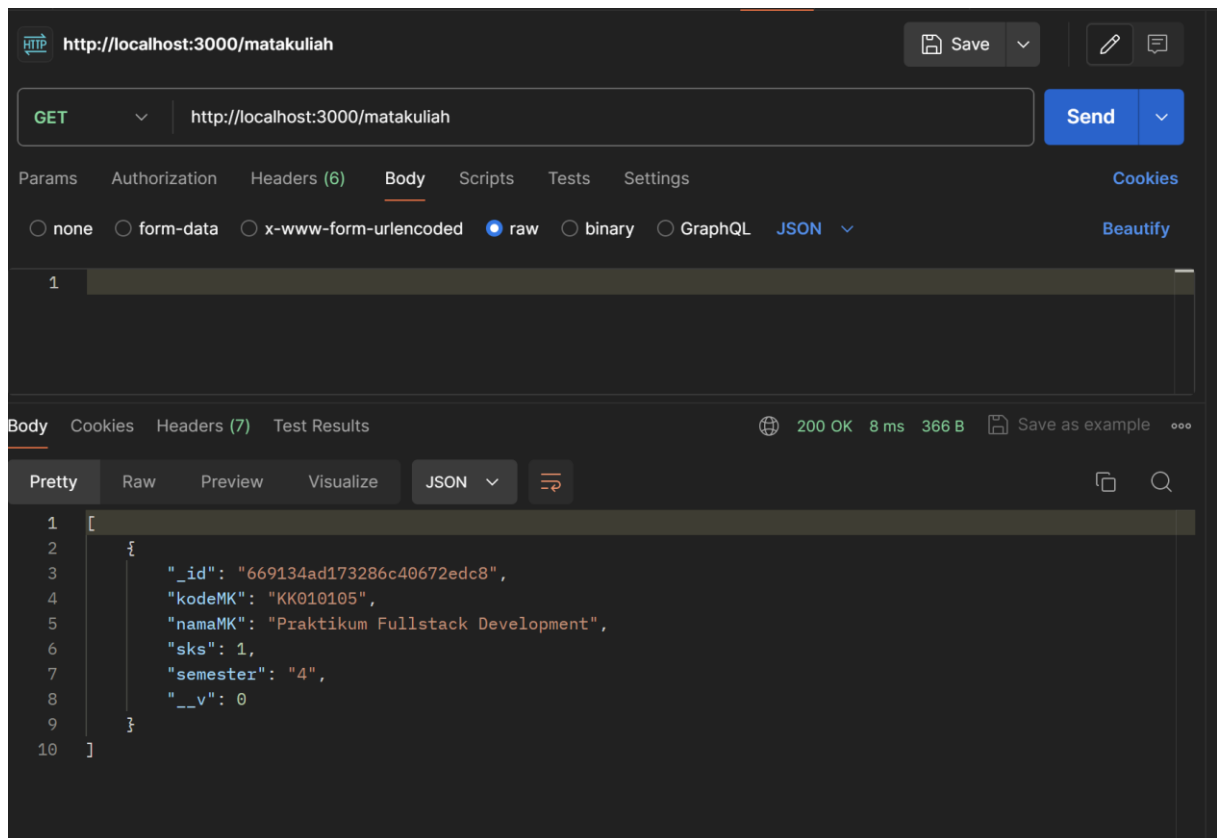


```
EXPLORER JS Mahasiswajs JS Matakuliahjs JS mahasiswajs JS matakuliahjs JS appjs JS config.js .env
MODUL 6
  JS Mahasiswajs
  JS Matakuliahjs
  node_modules
  routes
    JS mahasiswajs
    JS matakuliahjs
  .env
  JS appjs
  package-lock.json
  package.json

JS appjs > ...
32 app.use((req, res, next) => {
33   // Middleware untuk menangani error lainnya
34   error.status = 404;
35   next(error);
36 });
37
38 // Middleware untuk menangani error lainnya
39 app.use((err, req, res, next) => {
40   res.status(err.status || 500);
41   res.json({
42     error: {
43       message: err.message
44     }
45   });
46 });
47
48 // Jalankan server Express
49 app.listen(PORT, () => {
50   console.log('Server berjalan di port ${PORT}');
51 });
52
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS D:\K U L I A H\sem 4\fullproject\UAS\praktikum\modul 6> node app.js
(node:9924) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(node:9924) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Server berjalan di port 3000
Terhubung ke MongoDB
[]
```



Algoritma :

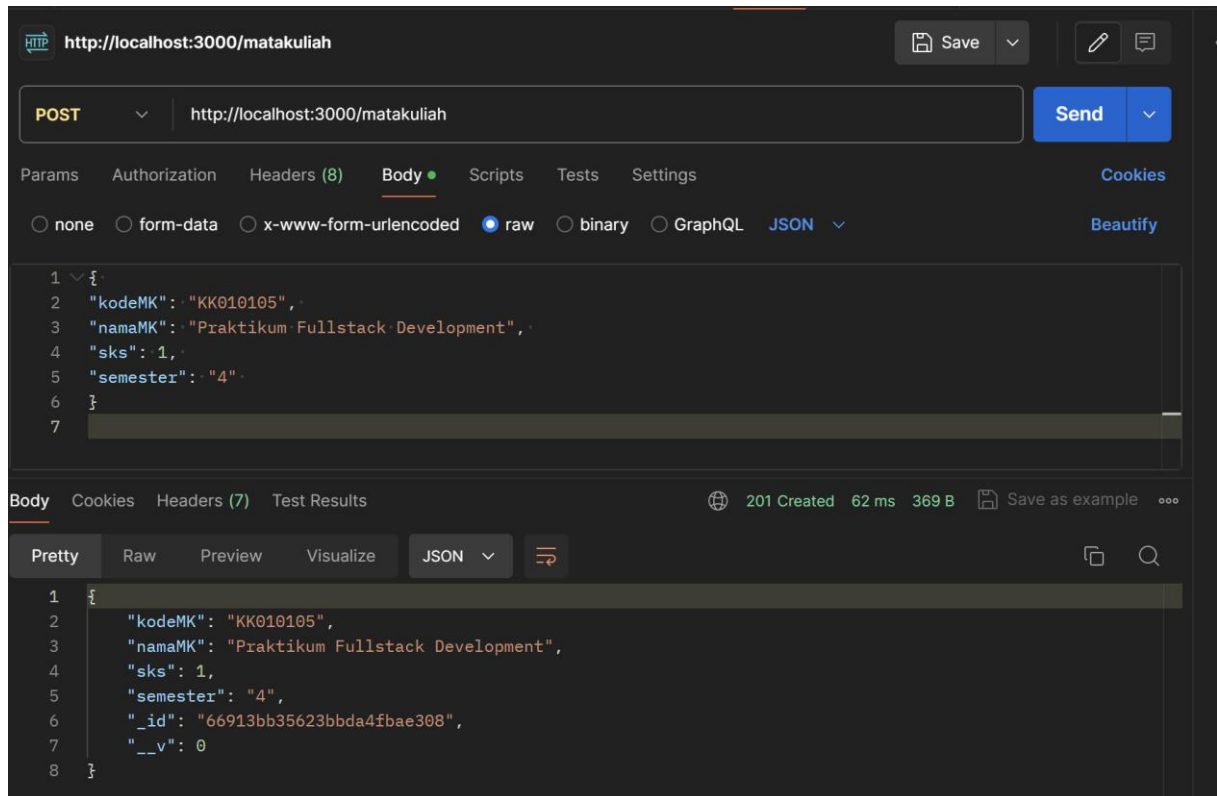
Pertama Program membuat Environment Configuration, kemudian Inisialisasi Modul dan Middleware, kemudian melakukan Koneksi ke MongoDB, Mengatur Middleware, Menggunakan Router, Menjalankan Server Express

Pengujian API :

a) Menggunakan postman

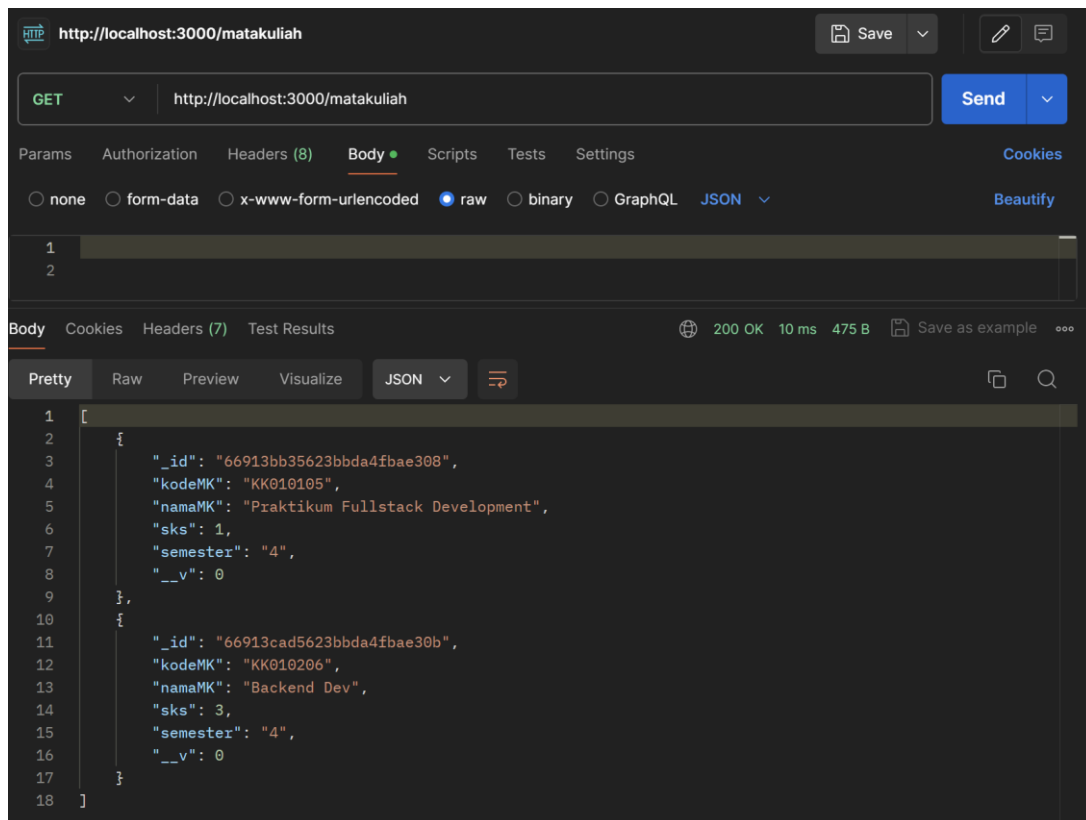
Scenario:

- **Tambah data mahasiswa**
 - Endpoint: `POST http://localhost:3000/matakuliah`
 - Body (JSON)



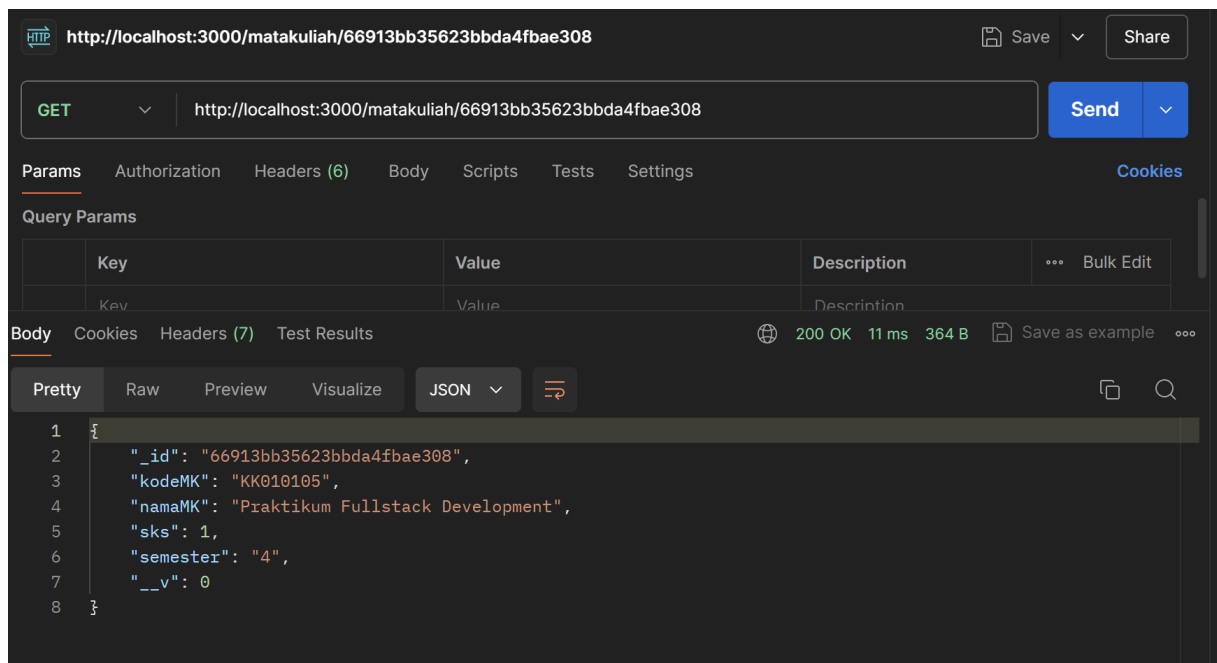
b) Ambil semua data mahasiswa (GET)

- Endpoint: `GET http://localhost:3000/matakuliah`



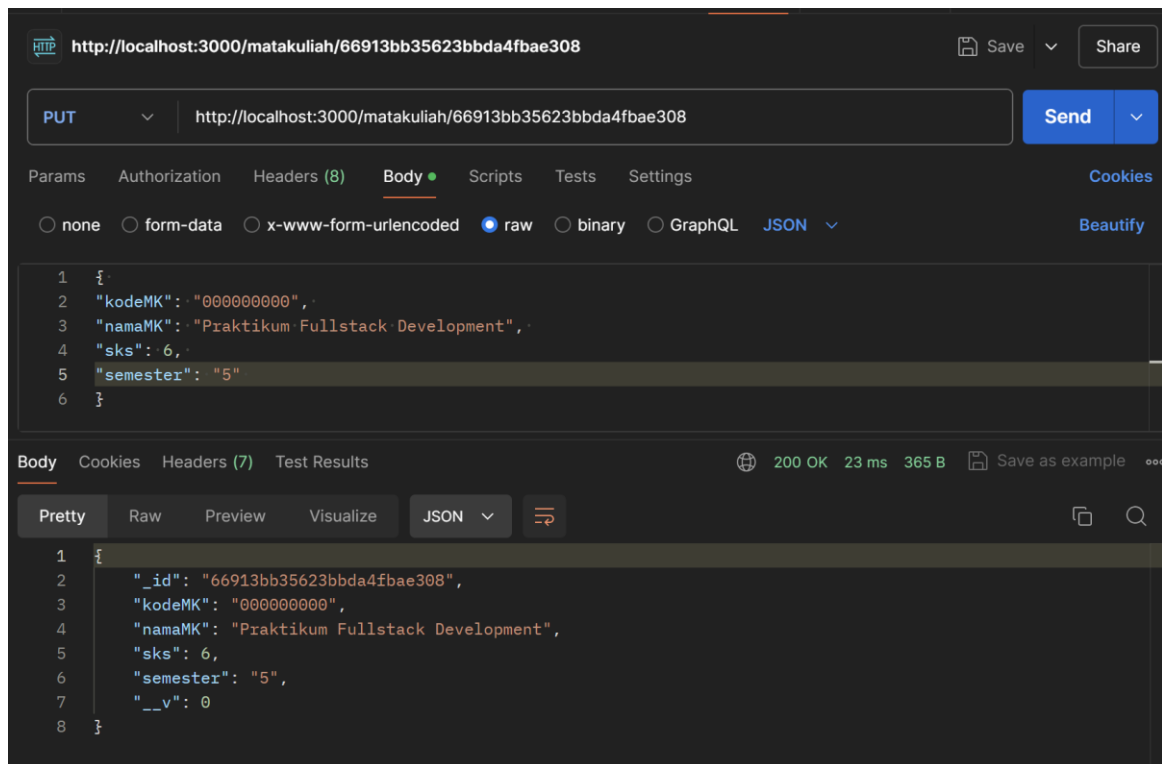
c) Ambil data matakuliah berdasarkan ID (GET)

- Endpoint: `GET http://localhost:3000/matakuliah/id`



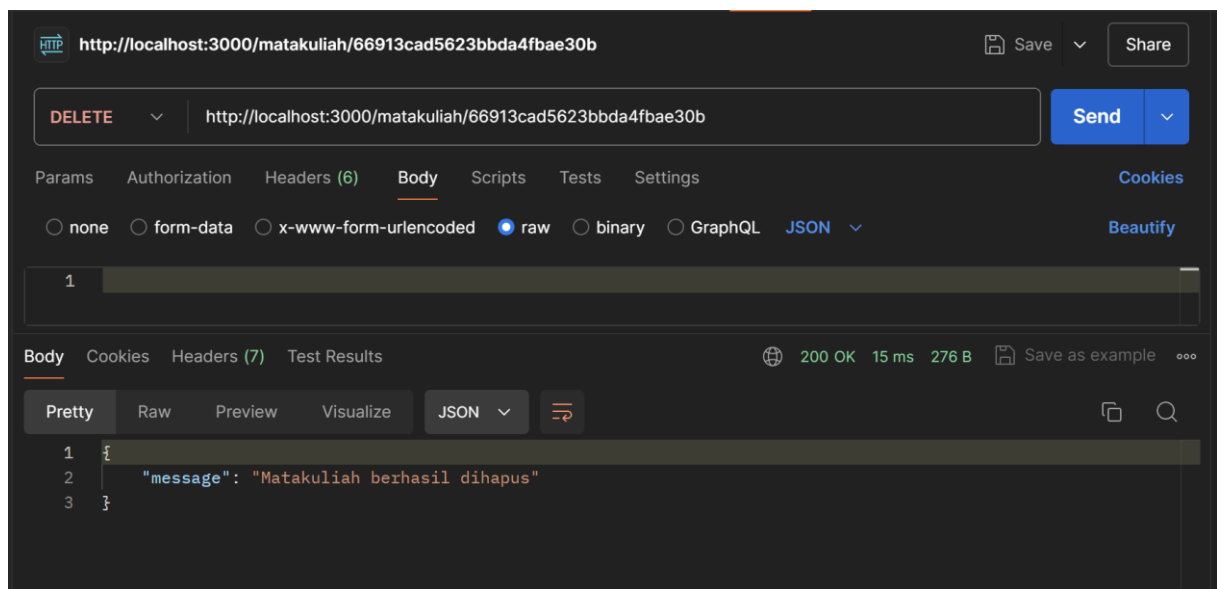
d) Ubah data matakuliah (PUT)

- Endpoint: `PUT http://localhost:3000/matakuliah/id`
- Body (JSON)
- Inputan:



e) Hapus data matakuliah (DELETE)

- Endpoint: `DELETE http://localhost:3000/matakuliah/id``



NB. Halaman ini dibuat sebanyak soal yang diberikan

KESIMPULAN

1. Implementasi CRUD di atas memungkinkan aplikasi untuk melakukan operasi dasar terhadap koleksi matakuliah dalam basis data MongoDB. Dengan menggunakan Express untuk mengatur rute dan middleware, serta Mongoose untuk berinteraksi dengan MongoDB, aplikasi dapat membuat, membaca, memperbarui, dan menghapus data matakuliah dengan respons yang sesuai terhadap permintaan HTTP yang diterima. Dengan memanfaatkan fungsi async/await dalam JavaScript, operasi-operasi ini dilakukan secara asynchronous untuk meningkatkan efisiensi dan responsivitas aplikasi.