PENGEMBANGAN APLIKASI BACKEND RESTAPI KRS MAHASISWA MENGGUNAKAN NODEJS

TUGAS

FARHAN MUALIF

5210411219

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN ELEKTRO

UNIVERSITAS TEKNOLOGI YOGYAKARTA

YOGYAKARTA

2020

Dalam pembuatan API untuk data KRS Mahasiswa dengan nodeJs berikut adalah langkah-langkahnya:

1. Hal pertama adalah melakukan install nodeJs, setelah melakukan penginstallan adalah membuat folder project selanjutnya lakukan inisialisasi npm dengan perintah *'npm init'*

```
Lenovo_ip3@LAPTOP-NBKRJNN3 MINGW64 /d
$ 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: krs-mahasiswa
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\package.json:

{
  "name": "krs-mahasiswa",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "directories": {
    "test": "test"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
```

2. install expressJs untuk mempermudah dalam pembuatan API dengan perintah '*npm i express'*.

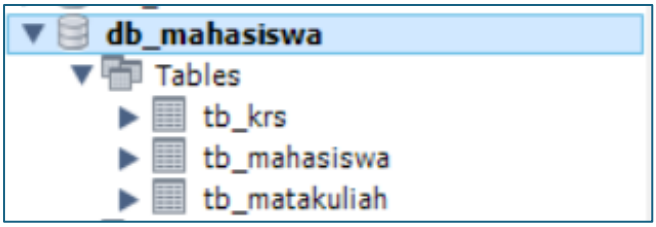
```
Lenovo_ip3@LAPTOP-NBKRJNN3 MINGW64 /d/api-krs
$ npm i express

added 64 packages in 4s

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

3. setelah melakukan installasi express, tahap selanjutnya adalah membuat file app.js
file ini akan dijalankan ketika menjalankan server.

```
const express = require("express");
const app = express();
const port = 3000;

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

Pada file tersebut berisi kode yang memanggil express dan melakukan listen dengan port 3000;

4. selanjutnya adalah mempersiapkan database dengan nama db_mahasiswa. Pada database ini terdapat tiga tabel, yaitu tabel mahasiswa, tabel matakuliah dan tabel krs.



Berikut adalah colom dari tb_mahasiswa, tb_matakuliah, dan tb_krs

Table: tb_mahasiswa

Columns:

id	int AI PK
npm	varchar(45)
nama	varchar(45)
tanggal_lahir	date
tempat_lahir	varchar(45)
agama	enum('ISLAM','KRISTEN','KONGHUCU','HINDU','BUDHA')
email	varchar(45)
alamat	varchar(45)
prodi	varchar(45)
jenis_kelamin	enum('L','P')

Table: tb_matakuliah

Columns:

id	int AI PK
kode	varchar(45)
nama	varchar(45)
sks	int
semester	varchar(45)
ruang	varchar(45)
jadwal	varchar(45)

Table: tb_krs

Columns:

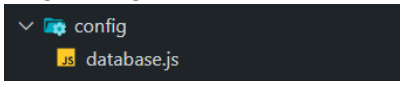
id	int AI PK
npm	varchar(45)
id_matakuliah	int

Selanjutnya adalah membuat relasi antar ketiga tabel tersebut:
Colom npm pada tb_krs berelasi dengan colom npm pada tb_mahasiswa, colom id_matakuliah berlasi dengan colom id pada tb_matakuliah.

Foreign Key Name	Referenced Table	Column	Referenced Column
krs_makul	`db_mahasiswa`.`tb_matakuliah	☐ id	
krs_mhs	`db_mahasiswa`.`tb_mahasiswa	☑ npm	npm
		☐ id_matakuliah	

Foreign Key Name	Referenced Table	Column	Referenced Column
krs_makul	`db_mahasiswa`.`tb_matakuliah	☐ id	
krs_mhs	`db_mahasiswa`.`tb_mahasiswa	☐ npm	
		☑ id_matakuliah	id

5. selanjutnya membuat koneksi database. Lakukan installasi mysql untuk nodeJs dengan perintah *'npm i mysql'* dan buat folder config dan file database.js. file ini berisi konfigurasi untuk menghubungkan ke database.



database.js

```
config > database.js > connection
You, yesterday | 1 author (You)
const mysql = require("mysql");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "Moenzel1933*",      You, yesterday • api: crud mh
  database: "db_mahasiswa",
});

connection.connect((err) => {
  if (err) {
    console.error("Error connecting to database: ", err);
    return;
  }
  console.log("Connected to database!");
});

module.exports = connection;
```

6. selajutnya adalah membuat validator, validator ini berfungsi untuk validasi ketika client mengirim request. Terdapat 3 validator yang perlu dibuat, yaitu krs-post-validator, makul-post-validator, dan mhs-post-validator.

```
const { body } = require("express-validator");

const mhsPostValidator = [
  [
    body("npm").notEmpty(),
    body("nama").notEmpty(),
    body("tanggal_lahir").notEmpty(),
    body("tempat_lahir").notEmpty(),
    body("agama").notEmpty(),
    body("email").notEmpty(),
    body("alamat").notEmpty(),
    body("prodi").notEmpty(),
    body("jenis_kelamin").notEmpty(),
  ],
];
module.exports = mhsPostValidator;
```

```
const { body } = require("express-validator");

const makulPostValidator = [
  [
    body("kode").notEmpty(),
    body("nama").notEmpty(),
    body("sks").notEmpty(),
    body("semester").notEmpty(),
    body("ruang").notEmpty(),
    body("jadwal").notEmpty(),
  ],
];
module.exports = makulPostValidator;
```

```
const { body } = require("express-validator");

const krsPostValidator = [
  [
    body("npm").notEmpty(),
    body("id_matakuliah").notEmpty()],
];
module.exports = krsPostValidator;
```

Validator ini untuk memastikan untuk request body yang dikirim harus terisi semua, jika salah satupun field tidak terisi, maka akan merespons error.

7. selanjutnya adalah membuat controller dengan cara membuat folder dengan nama controller. dan berisi fille-file contoller. Controller ini berisi logic dan proses query database. Teradapat tiga controller yang perlu di buat, yaitu krs-controller, mahasiswa-controller, dan matakuliah controller. Berikut adalah penjelasan dari masing-masing controller tersebut:
   a. mahasiswa-controller.js
      pada file ini terdapat class MahasiswaController dan class ini memiliki beberapa method. Berikut adalah masing-masing methodnya.
      i. Method index. Method ini untuk menampilkan semua data mahasiswa

```
class MahasiswaController {
  static async index(req, res) {
    connection.query("SELECT * FROM tb_mahasiswa", (err, rows) => {
      if (err) {
        console.error("Error executing query: ", err);
        return res.status(500).json({ error: "Internal Server Error" });
      }
      res.json({ data: rows });
    });
  }      You, yesterday • api: crud mhs,krs,makul
```

      ii. Method store. Method ini untuk insert ke database.

```
static async store(req, res) {
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).json({
      errors: errors.array(),
    });
  }
  let formData = {
    nama: req.body.nama,
    npm: req.body.npm,
    tanggal_lahir: req.body.tanggal_lahir,
    tempat_lahir: req.body.tempat_lahir,
    agama: req.body.agama,
    email: req.body.email,
    alamat: req.body.alamat,
    prodi: req.body.prodi,
    jenis_kelamin: req.body.jenis_kelamin,
  };
```

```
  connection.query(
    "INSERT INTO tb_mahasiswa SET ?",
    formData,
    function (err, rows) {
      //if(err) throw err
      if (err) {
        return res.status(500).json({
          status: false,
          message: "Internal Server Error",
          data: err,
        });
      } else {
        return res.status(201).json({
          status: true,
          message: "Insert Data Successfully",
          data: rows[0],
        });
      }
    }
  );
}
```

      Pada metod ini melakukan validasi dengan validator yang telah dibuat sebelumnya, setelah vallidasi lolos maka akan melakukan insert data mahasiswa ke database.
      iii. Method update.

```
static update(req, res) {
  try {
    const errors = validationResult(req);

    if (!errors.isEmpty()) {
      return res.status(422).json({
        errors: errors.array(),
      });
    }

    //id post
    let id = req.params.id;

    //data post
    let formData = {
      nama: req.body.nama,
      npm: req.body.npm,
      tanggal_lahir: req.body.tanggal_lahir,
      tempat_lahir: req.body.tempat_lahir,
      agama: req.body.agama,
      email: req.body.email,
      alamat: req.body.alamat,
      prodi: req.body.prodi,
      jenis_kelamin: req.body.jenis_kelamin,
    };
```

```
    // update query
    connection.query(
      `UPDATE tb_mahasiswa SET ? WHERE id = ${id}`,
      formData,
      function (err, rows) {
        //if(err) throw err
        if (err) {
          return res.status(500).json({
            status: false,
            message: err.message,
            name: err.name,
          });
        } else {
          return res.status(200).json({
            status: true,
            message: "Update Data Successfully!",
          });
        }
      }      You, 59 seconds ago • Uncommitted change
    );
  } catch (error) {
    return res.status(200).json({
      status: false,
      message: err.message,
      name: err.name,
    });
  }
```

Method ini digunakan untuk melakukan update data mahaisswa berdasarkan id.

iv. Method delete ini digunakan untuk menghapus data mahasiswa dari database berdasarkan id. Dikarenakan konfigurasi foreign key options dengan tb_krs diatur dengan on delete cascade dan on update cascade, maka setiap menghapus data mahasiswa, data krs akan terhapus juga.

```
static delete(req, res) {
  try {
    let id = req.params.id;
    You, 11 seconds ago • Uncommitted changes
    connection.query(
      `DELETE FROM tb_mahasiswa WHERE id = ${id}`,
      function (err, rows) {
        //if(err) throw err
        if (err) {
          return res.status(500).json({
            status: false,
            message: err.message,
            name: err.name,
          });
        } else {
          return res.status(200).json({
            status: true,
            message: "Delete Data Successfully!",
          });
        }
      }
    );
  } catch (error) {
    return res.status(200).json({
      status: false,
      message: error.message,
      name: error.name,
    });
  }
```

```
        } else {
          return res.status(200).json({
            status: true,
            message: "Delete Data Successfully!",
          });
        }
      }
    );
  } catch (error) {
    return res.status(200).json({
      status: false,
      message: error.message,
      name: error.name,
    });
  }
}
```

b. Matakuliah controller. pada file ini terdapat class MatakuliahController dan class ini memiliki beberapa method. Berikut adalah masing-masing methodnya:

i. Method index. Method ini untuk menampilkan semua data Matakuliah yang ada pada database.

```
static async index(req, res) {
  connection.query("SELECT * FROM tb_matakuliah", (err, rows) => {
    if (err) {
      console.error("Error executing query: ", err);
      return res.status(500).json({ error: "Internal Server Error" });
    }
    res.json({ data: rows });
  });
}
```

ii. Method show. Method ini digunakan untuk menampilkan data matakuliah dari database berdasarkan id.

```
static async show(req, res) {
  connection.query(
    `SELECT * FROM tb_matakuliah WHERE id=${req.params.id}`,
    (err, rows) => {
      if (err) {
        console.error("Error executing query: ", err);
        return res.status(500).json({ error: "Internal Server Error" });
      }
      res.json({ data: rows });
    }
  );
}
```

iii. Method store. Method ini untuk insert data matakuliah ke database.

```
static async store(req, res) {
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).json({
      errors: errors.array(),
    });
  }
  let formData = {
    kode: req.body.kode,
    nama: req.body.nama,
    sks: req.body.sks,
    semester: req.body.semester,
    ruang: req.body.ruang,
    jadwal: req.body.jadwal,
  };
```

```
connection.query(
  "INSERT INTO tb_matakuliah SET ?",
  formData,
  function (err, rows) {
    //if(err) throw err
    if (err) {
      return res.status(500).json({
        status: false,
        message: "Internal Server Error",
        data: err,
      });
    } else {
      return res.status(201).json({
        status: true,
        message: "Insert Data Successfully",
        data: rows[0],
      });
    }
  }
);
```

iv. Method update. Method ini digunakan untuk melakukan update data matakuliah berdasarkan id.

```
static update(req, res) {
  try {
    const errors = validationResult(req);

    if (!errors.isEmpty()) {
      return res.status(422).json({
        errors: errors.array(),
      });
    }

    //id post
    let id = req.params.id;

    //data post
    let formData = {      You, 9 minutes a
      kode: req.body.kode,
      nama: req.body.nama,
      sks: req.body.sks,
      semester: req.body.semester,
      ruang: req.body.ruang,
      jadwal: req.body.jadwal,
    };
```

```
    connection.query(
      `UPDATE tb_matakuliah SET ? WHERE id = ${id}`,
      formData,
      function (err, rows) {
        //if(err) throw err
        if (err) {
          return res.status(500).json({
            status: false,
            message: err.message,
            name: err.name,
          });
        } else {
          return res.status(200).json({
            status: true,
            message: "Update Data Successfully!",
          });
        }
      }
    );
  } catch (error) {
    return res.status(200).json({
      status: false,
      message: error.message,
      name: error.name,
    });
  }
```

v. Method delete. Method delete ini digunakan untuk menghapus data mahasiswa berdasarkan id. Dikarenakan konfigurasi foreign key options antara tb_mahasiswa dengan tb_krs diatur dengan on delete cascade dan on update cascade, maka setiap menghapus data mahasiswa, data krs akan terhapus juga.

```
static delete(req, res) {
  try {
    let id = req.params.id;

    connection.query(
      `DELETE FROM tb_matakuliah WHERE id = ${id}`,
      function (err, rows) {      You, 2 minutes ago
        //if(err) throw err
        if (err) {
          return res.status(500).json({
            status: false,
            message: err.message,
            name: err.name,
          });
        } else {
          return res.status(200).json({
            status: true,
            message: "Delete Data Successfully!",
          });
        }
      }
    );
  } catch (error) {
    return res.status(200).json({
      status: false,
      message: error.message,
      name: error.name,
    });
  }
```

c. KrsController. pada file ini terdapat class KrsController dan class ini memiliki beberapa method. Berikut adalah masing-masing methodnya:

i. Method index. Method ini untuk menampilkan semua data Matakuliah yang ada pada database.

```
static async index(req, res) {
  connection.query(
    `SELECT tb_mahasiswa.npm,tb_mahasiswa.nama as nama_mahasiswa,tb_matakuliah.kode as kode_makul,tb_matakuliah.nama as
    nama_matakuliah,tb_matakuliah.sks,tb_matakuliah.semester,tb_matakuliah.ruang,tb_matakuliah.jadwal FROM tb_krs JOIN tb_mahasiswa
    ON tb_krs.npm=tb_mahasiswa.npm JOIN tb_matakuliah ON tb_krs.id_matakuliah=tb_matakuliah.id`,
    (err, rows) => {
      if (err) {
        console.error("Error executing query: ", err);
        return res.status(500).json({ error: "Internal Server Error" });
      }
      res.json({ data: rows });
    }
  );
}
```

Pada query ini melakukan melakukan select pada tabel tb_krs dan melakukan join terhadap tb_mahasiswa dan tb_matakuliah, join antara tb_krs dengan mahasiswa beralasi dengan npm, sedangkan join antara tb_krs dengan tb_matakuliah berelasi dengan id dari tb_matakuliah.

ii. Method show. Method ini digunakan untuk menampilkan data krs dari database berdasarkan npm mahasiswa.

```
static async show(req, res) {
  console.log(req.params.npm);
  connection.query(
    `SELECT tb_mahasiswa.npm,tb_mahasiswa.nama as nama_mahasiswa,tb_matakuliah.kode as kode_makul,tb_matakuliah.nama as
    nama_matakuliah,tb_matakuliah.sks,tb_matakuliah.semester,tb_matakuliah.ruang,tb_matakuliah.jadwal FROM tb_krs JOIN tb_mahasiswa
    ON tb_krs.npm=tb_mahasiswa.npm JOIN tb_matakuliah ON tb_krs.id_matakuliah=tb_matakuliah.id WHERE tb_krs.npm=${req.params.npm}`,
    (err, rows) => {
      if (err) {
        console.error("Error executing query: ", err);
        return res.status(500).json({ error: "Internal Server Error" });
      }
      res.json({ data: rows });
    }
  );
}
```

konsep join pada method tidak jauh berdeda dengan method index, hanya saja method ini menampilkan data lebih spesifik berdasarkan npm.

iii. Method store.

```
static async store(req, res) {
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).json({
      errors: errors.array(),
    });
  }
  let formData = {
    npm: req.body.npm,
    id_matakuliah: req.body.id_matakuliah,
  };
  connection.query(
    "INSERT INTO tb_krs SET ?",
    formData,
    function (err, rows) {
      //if(err) throw err
      if (err) {
        return res.status(500).json({
          status: false,
          message: "Internal Server Error",
          data: err,
        });
      } else {
        return res.status(201).json({
          status: true,
          message: "Insert Data Successfully",
          data: rows[0],
        });
      }
    }
  );
}
```

Method ini untuk insert data krs ke database. Data yang diinsertkan adalah npm mahasiswa dan id matakuliah.

vi. Method delete. Method delete ini digunakan untuk menghapus data krs berdasarkan npm mahasiswa dan id krs. Dikarenakan konfigurasi foreign key options antara tb_krs dengan tb_mahasiswa diatur dengan on delete cascade dan on update cascade, maka setiap menghapus data krs, data krs akan terhapus juga.

```
static delete(req, res) {
  let npm = req.params.npm;
  let id = req.params.id;

  connection.query(      You, 2 days ago • api: crud mhs
    `DELETE FROM tb_krs WHERE npm = ${npm} && id=${id}`,
    function (err, rows) {
      //if(err) throw err
      if (err) {
        return res.status(500).json({
          status: false,
          message: err.message,
          name: err.name,
        });
      } else {
        return res.status(200).json({
          status: true,
          message: "Delete Data Successfully!",
        });
      }
    }
  );
}
```

vii. Method Update. Method ini digunakan untuk melakukan mengubah krs mahasiwa berdasarkan npm mahasiswa dan id krs. Data yang diubah cukup id dari matakuliah.

```
static update(req, res) {
  try {
    const errors = validationResult(req);

    if (!errors.isEmpty()) {
      return res.status(422).json({
        errors: errors.array(),
      });      You, 7 seconds ago • Uncommitte
    }

    //id post
    let npm = req.params.id;

    //data post
    let formData = {
      id_matakuliah: req.body.id_matakuliah,
    };
```

```
connection.query(
  `UPDATE tb_krs SET ? WHERE npm = ${npm}`,
  formData,
  function (err, rows) {
    //if(err) throw err
    if (err) {
      return res.status(500).json({
        status: false,
        message: err.message,
        name: err.name,
      });
    } else {
      return res.status(200).json({
        status: true,
        message: "Update Data Successfully!",
      });
    }
  }
);
```

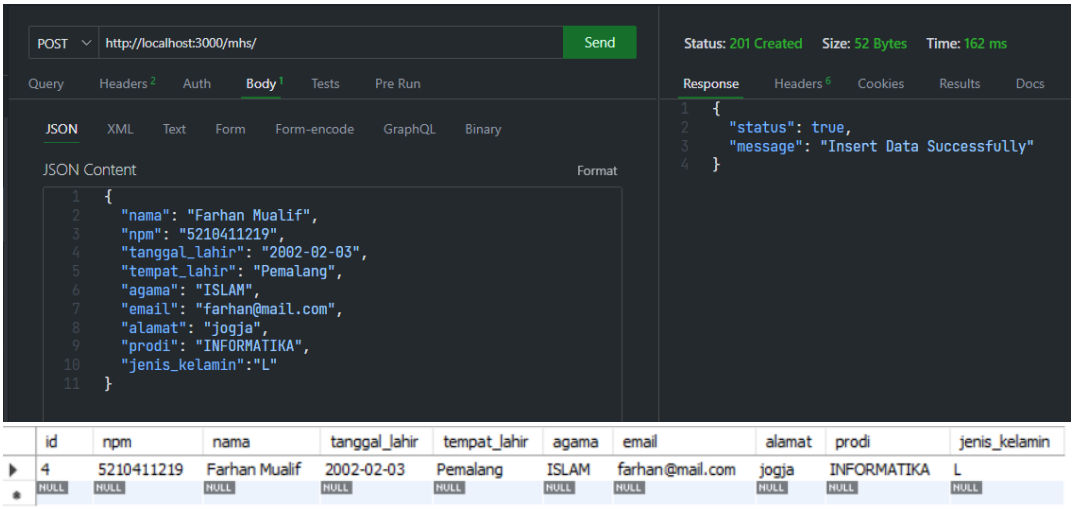
8. Selanjutnya membuat route dengan cara membuat folder router/route.js. Route ini semacam instruksi atau petunjuk agar tahu apa yang harus dilakukan ketika pengguna mengakses alamat URL tertentu.

```
You, 2 seconds ago | 1 author (You)
const router = require("express").Router();

const KrsController = require("../controller/krs-controller");
const MahasiswaController = require("../controller/mahasiswa-controller");
const MatakuliahController = require("../controller/matakuliah-controller");

const krsPostValidator = require("../validator/krs-post-validator");
const makulPostValidator = require("../validator/makul-post-validator");
const mhsPostValidator = require("../validator/mhs-post-validator");

router.get("/mhs", MahasiswaController.index);
router.delete("/mhs/(:id)", MahasiswaController.delete);
router.get("/mhs/(:id)", MahasiswaController.show);
router.post("/mhs", mhsPostValidator, MahasiswaController.store);
router.put("/mhs/(:id)", mhsPostValidator, MahasiswaController.update);

router.get("/makul", MatakuliahController.index);
router.get("/makul/(:id)", MatakuliahController.show);
router.post("/makul", makulPostValidator, MatakuliahController.store);
router.put("/makul/(:id)", makulPostValidator, MatakuliahController.update);
router.delete("/makul/(:id)", MatakuliahController.delete);

router.get("/krs", KrsController.index);      You, 2 days ago • api: crud mhs
router.post("/krs", krsPostValidator, KrsController.store);
router.get("/krs/(:npm)", KrsController.show);
router.put("/krs/(:npm)/(:id)", KrsController.update);
router.delete("/krs/(:npm)/(:id)", KrsController.delete);
```

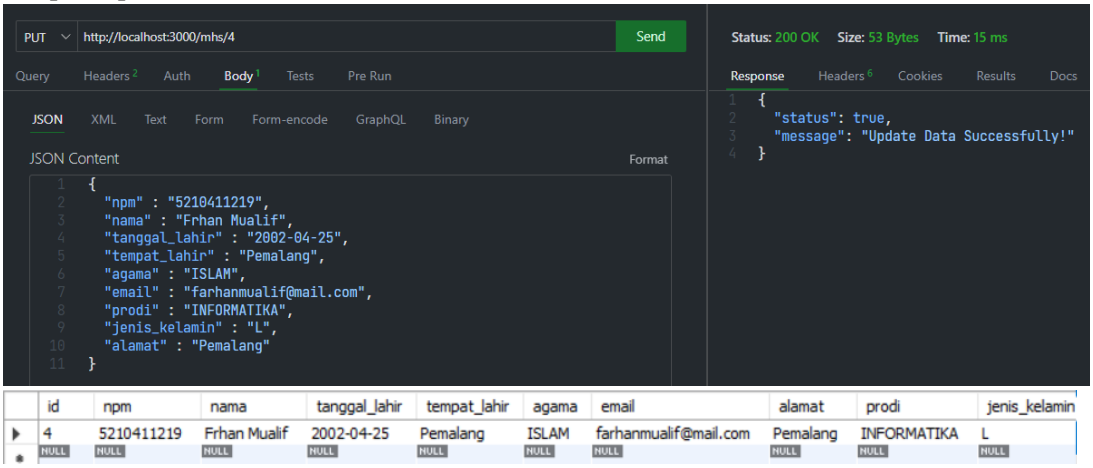
Ada beberapa route yang meyisipkan validator di parameter method. Ini berfungsi untuk validasi request yang dikirimkan oleh client, jika validasi lolos maka proses selanjutnya dapat dilakukan. Dan tiap route memaggil method class yang sudah dibuat sebelumnya.

9. Melakukan testing menggunakan extension Thunder client vscode. Berikut adalah masing-masing testing endpoint bedasarkan yang telah dibuat.

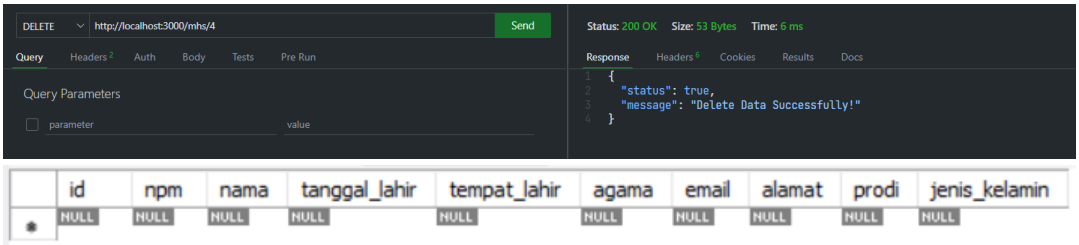
a. Endpoin post mhs atau insert mahasiswa.



b. Endpoint put mhs atau ubah mahasiswa.

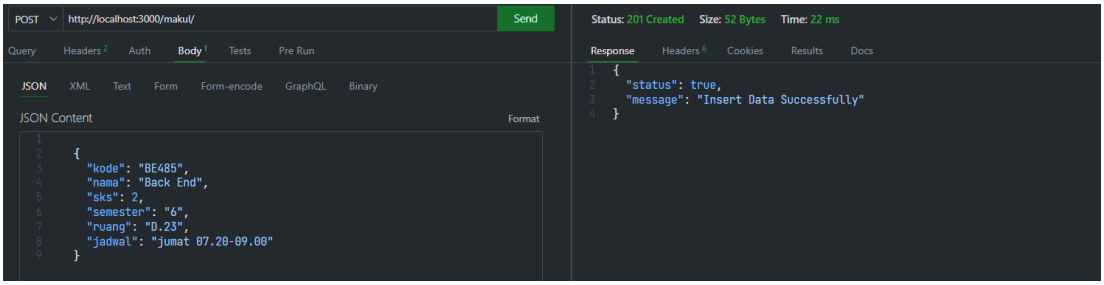


c. Endpoint hapus mhs atau ubah mahasiswa.



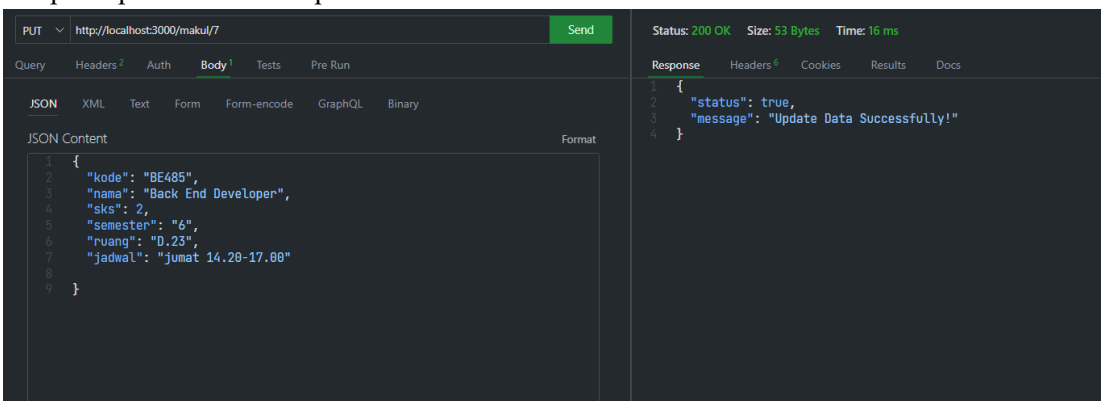
	id	npm	nama	tanggal_lahir	tempat_lahir	agama	email	alamat	prodi	jenis_kelamin
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

d. Endpoint post makul atau insert data matakuliah:



	id	kode	nama	sks	semester	ruang	jadwal
▶	7	BE485	Back End	2	6	D.23	jumat 07.20-09.00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

e. Endpoint put makul atau update data matakuliah:

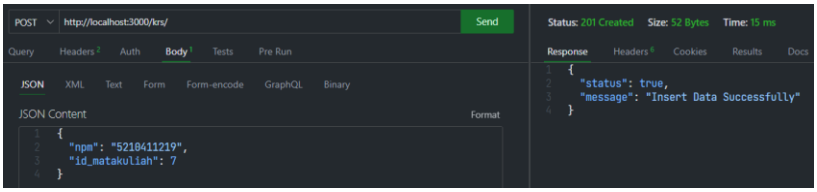


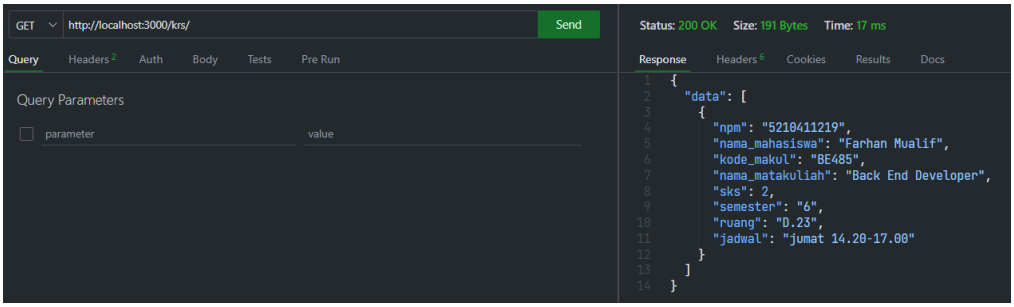
	id	kode	nama	sks	semester	ruang	jadwal
▶	7	BE485	Back End Developer	2	6	D.23	jumat 14.20-17.00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

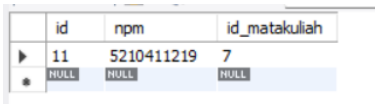
f. Endpoint post krs atau insert data krs:

	id	npm	nama	tanggal_lahir	tempat_lahir	agama	email	alamat	prodi	jenis_kelamin
▶	5	5210411219	Farhan Mualif	2002-02-03	Pemalang	ISLAM	farhan@mail.com	jogja	INFORMATIKA	L
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

	id	kode	nama	sks	semester	ruang	jadwal
	7	BE485	Back End Developer	2	6	D.23	jumat 14.20-17.00
▶	8	FE835	FrontEnd	3	6	D.24	Kamis 14.20-17.00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

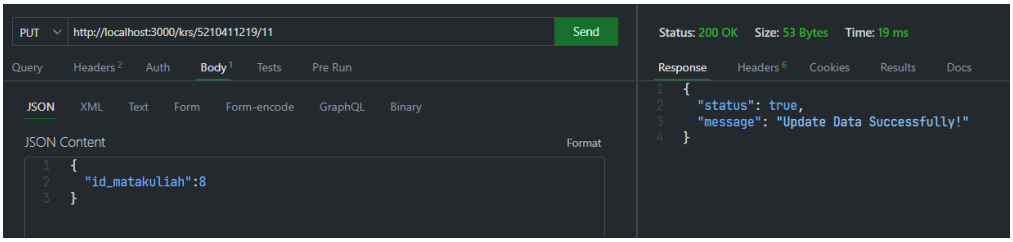


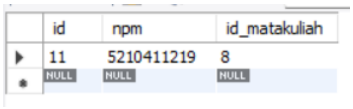




id	npm	id_matakuliah
11	5210411219	7
NULL	NULL	NULL

g. Endpoint put krs atau ubah data krs:





id	npm	id_matakuliah
11	5210411219	8
NULL	NULL	NULL

h. Endpoint get krs atau ambil data krs: