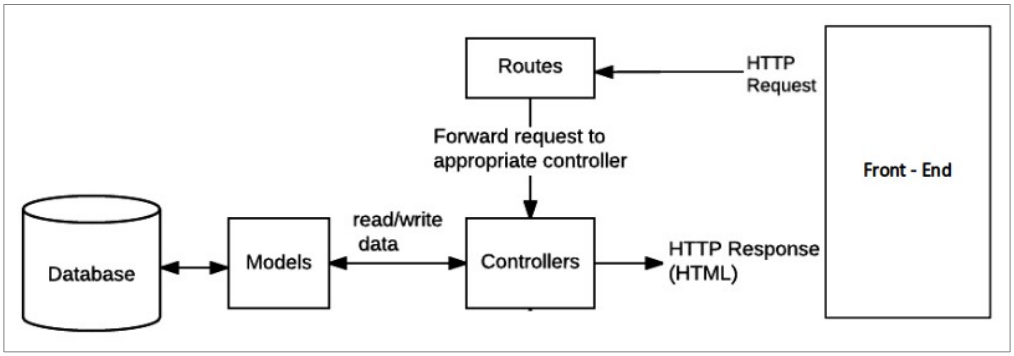
Controller

A. Pengantar

Untuk membuat aplikasi semakin baik, semakin mudah untuk di rawat dan dikelola, mudah untuk debugging pada saat development atupun manintenance, juga mudah untuk dilakukan pengembangan, perlu dibuat struktur aplikasi yang lebih tertata, dengan modul-modul yang ditujukan untuk melakukan tugas secara spesifik.

Gambar 1 menunjukkan diagram aliran data secara logik dari front-end ke back-end. Pada gambar 1 terlihat bahwa pada bagian back-end, request dikerjakan oleh tiga bagian yang bekerja secara spesifik, yaitu **Routers**, **Controller**, dan **Models**.



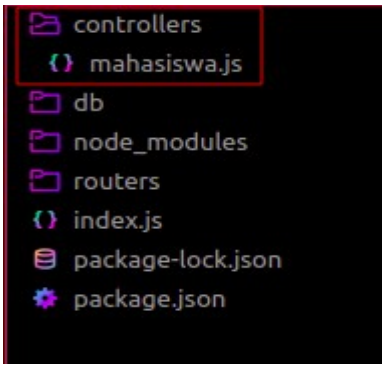
Gambar 1. Diagram alur struktur Routers, Controllers, Models

Routers bekerja untuk mengarahkan request ke controllers sesuai dengan end-point yang dituju. Controllers melakukan proses-proses logik, mendapatkan respons dari mode, dan meneruskan respons ke front-end. Sedangkan Models bertugas untuk melakukan pemodelan dan pengelolaan basis data.

Materi ini akan membahas bagaiamana memisahkan kode pada bagian routers yang telah dibuat pada pertemuan sebelumnya menjadi **routers** dan **controllers.** Sehingga pada akhir pertemuan ini akan di dapatkan tambahan module controllers yang akan membuat aplikasi menjadi lebih terstruktur dengan dan modul yang lebih spesifik daripada petemuan sebelumnya.

B. Struktur Aplikasi

Untuk keperluan diatas, maka perlu dibuat sebuah folder baru dengan ***controllers***. Di dalam folder ini dibuat file ***mahasiswa.js***, sebagaimana terlihat pada gambar 2 dan gambar 3. Gambar 2 menunjukkan struktur baru aplikasi setelah penambahan folder **controllers** dan file mahasiswa.js, sedangkan gambar 3 menunjukkan struktur awal dari file **mahasiswa.js**.



Gambar 2. Struktur aplikasi

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

module.exports = {

}
```

Gambar 3. mahasiswa.js

C. Pemisahan Kode

Gambar 4 menunjukkan function yang melakukan operasi untuk end-point ***get('/mahasiswa')*** yang berada file '***routers/mahasiswa.js***', function ini akan dipecah menjadi ***routes*** dan ***controllers***. Bagian kode yang di **blok** biru pada gambar 4 merupakan kode yang akan dipindahkan ke controllers.

```
routerMhs.get('/mahasiswa', (req,res) => {
    const qstring = "SELECT * FROM mahasiswa";
    connection.query(qstring, (err,data) => {
        if (err) {
            console.log("error: ", err);
            res.status(500).send({
                message : err.message || "Terjadi kesalahan saat get data"
            });
        }
        else res.send(data)
    });
})
```

Di pindah ke 'controllers/mahasiswa.js'

Gambar 4. Bagian kode yang akan dipisahkan

Gambar 5 menunjukkan function untuk untuk end-point ***get('/mahasiswa')*** setelah diambil bagian controllers nya.

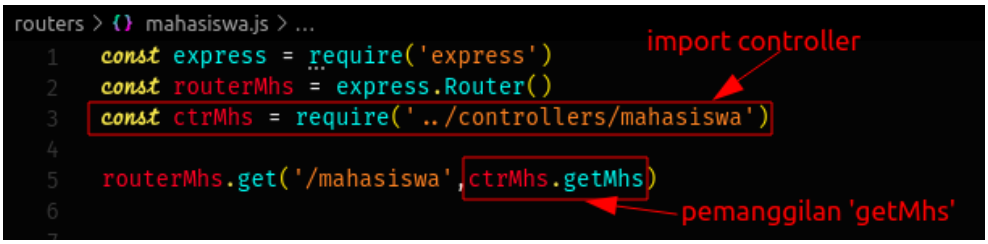
```
routerMhs.get('/mahasiswa', )
```

Gambar 5. routerMhs.get('/mahasiswa') setelah dipisahkan

Gambar 6 menunjukkan file '***controllers/mahasiswa.js***' setelah ditambahkan dengan file yang dipindahkan dari '***routers/mahasiswa.js***'. Pada gambar 6 terlihat bahwa kode yang di **blok** dengan warna biru merupakan kode yang berasal dari '***routers/mahasiswa.js***' pada end-poin **get('/mahasiswa')**. Agar proses **routing** masih bisa berjalan dengan baik, maka pada kode ini ditambahkan kode **getMhs:**, seperti terlihat dalam kotak merah dalam gambar 6.

Gambar 6. getMhs pada 'controllers/mahasiswa.js'

Pada gambar 5 dapat dilihat bahwa end-point **get('/mahasiswa')** sekarang sudh tidak lagi mempunyai kode untuk memproses data dan memberikan response ke user, karena telah dipindahkan ke controllers. Untuk itu perlu dilakukan **koneksi** antara ***routers*** dengan ***controller*** agar aplikasi masih tetap berjalan dengan baik. Untuk tujuan ini maka perlu dilakukan ***import*** controller kedalam routers dan meletakkan ***variabel*** *dari* controller yang sesuai kedalam end-poin nya masing-masing. Gambar 7 menunjukkan hasil import controller dan penambahan variabel dari controller untuk end-point **get('/mahsiswa')**.

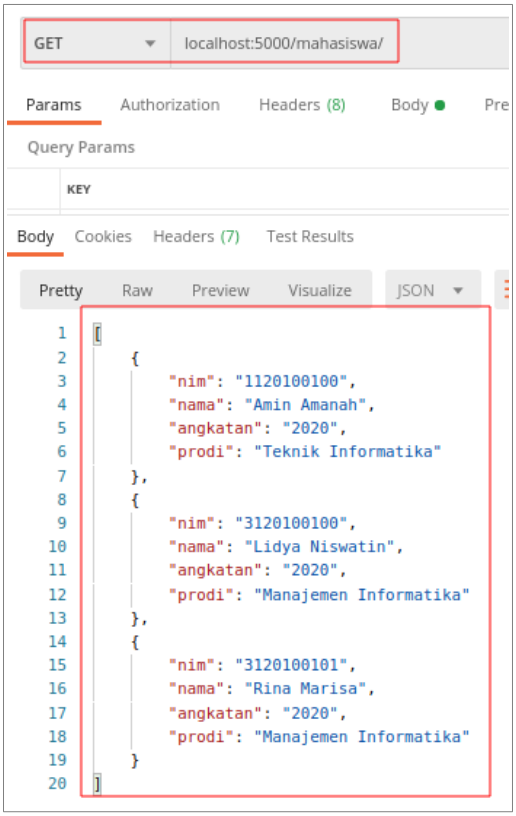


Gambar 7. koneksi controllers ke routers

C. Uji Coba dengan Postman

Gambar 8 menunjukkan hasil uji coba end-point **get('/mahasiswa')** setelah dilakukan penambahan controllers dan dilakukan pemindahan bagian kode ke controllers. Gambar 8 menunjukkan bahwa prose ini berjalan dengan baik dan aplikasi

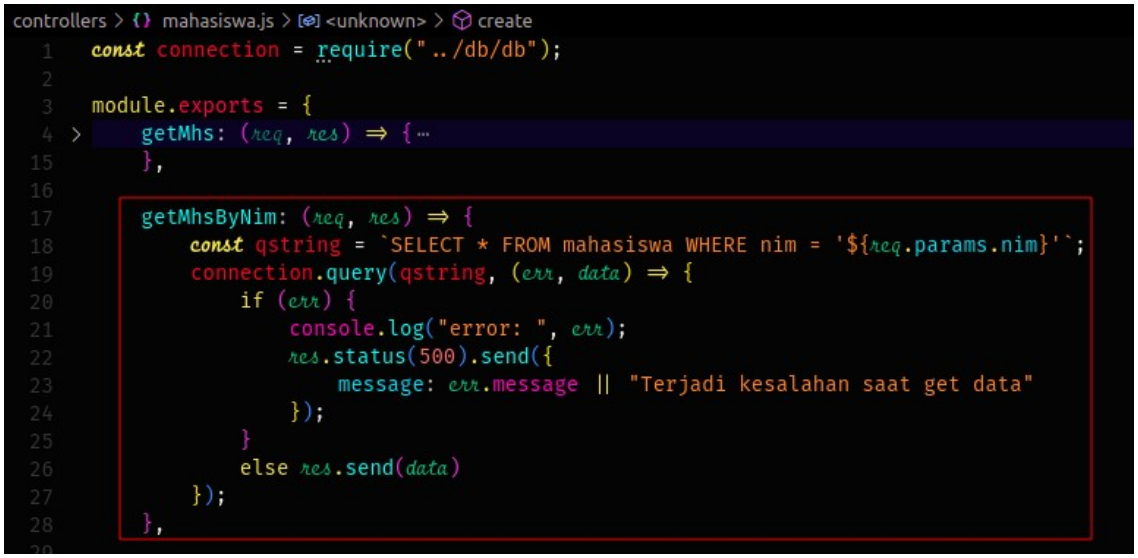
masih bisa berjalan sebagaimana seharusnya. Dengan struktur aplikasi yang lebih tertata dan sistematis.



Gambar 8. hasil uji coba pada postman

C. Pemisahan Semua End-Point

Dengan cara yang sama seperti pada bagian B, semua end-point pada **'*routers/mahasiswa.js*'** dipisahkan bagian controller nya untuk diletakkan pada **'*controllers/mahasiswa.js*'.**



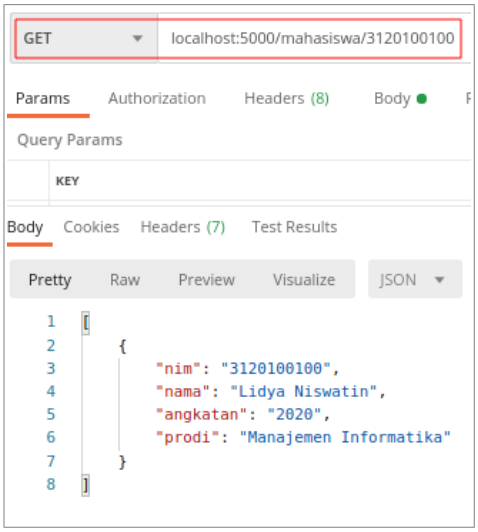
Gambar 9. getMhsByNim pada 'controllers/mahasiswa.js'

Gambar 9 menunjukkan file ‘controllers/mahasiswa.js’ setelah penambahan kode untuk melayani end-point **get(‘*mahasiswa/:nim*’).** Sedangkan gambar 10 menunjukkan file ‘routers/mahasiswa.js’ setelah pemisahan dan penambahan pemanggilan controllers. Gambar 11 menunjukkan hasil uji coba di Postman

routers > {} mahasiswa.js > ...

```
1  const express = require('express')
2  const routerMhs = express.Router()
3  const ctrMhs = require('../controllers/mahasiswa')
4
5  routerMhs.get('/mahasiswa',ctrMhs.getMhs)
6  routerMhs.get('/mahasiswa/:nim',ctrMhs.getMhsByNim)
7
```

Gambar 10. pemanggilan controller ***getMhsByNim*** *pada ‘routers/mahasiwa.js’*



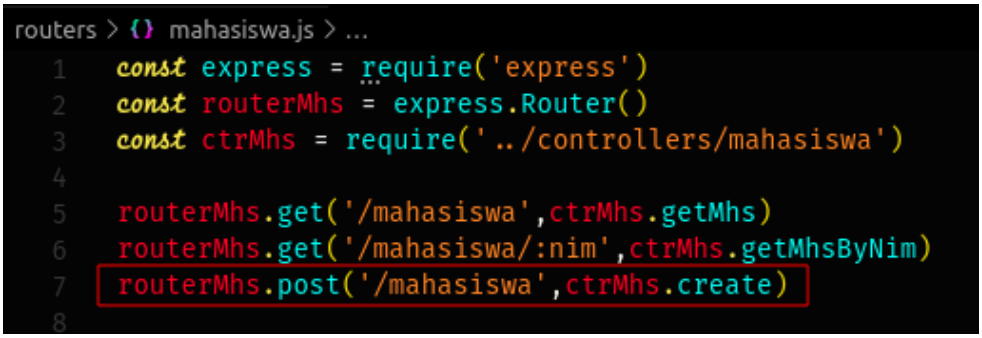
Gambar 11. Hasil uji coba end-point ***get(‘/mahasiswa/:nim’)***

controllers > {} mahasiswa.js > [@] <unknown> > update

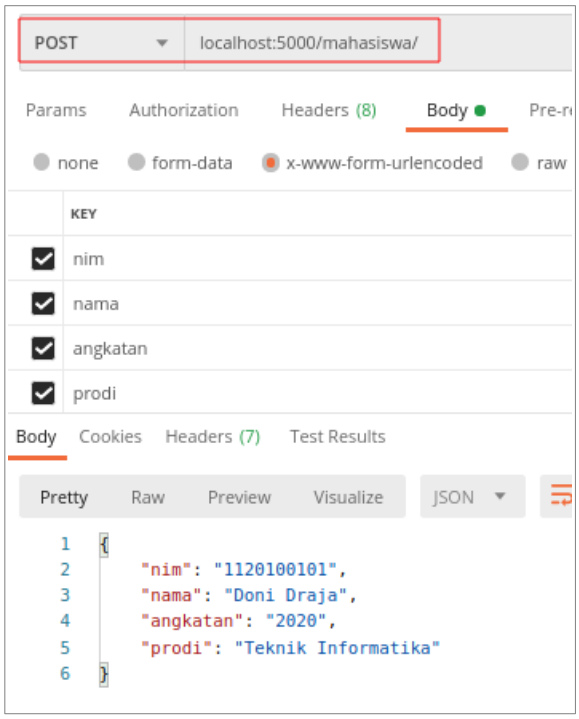
```
1  const connection = require("../db/db");
2
3  module.exports = {
4      getMhs: (req, res) => {…
15     },
16
17     getMhsByNim: (req, res) => {…
28     },
29
30     create: (req, res) => {
31         const mahasiswaBaru = req.body;
32         connection.query("INSERT INTO mahasiswa SET ?", mahasiswaBaru, (err, result) => {
33             if (err) {
34                 console.log("error: ", err);
35                 res.status(500).send({
36                     message: err.message || "Terjadi kesalahan saat insert data"
37                 });
38             }
39             else
40                 res.send(mahasiswaBaru)
41         });
42     },
```

Gambar 11. create pada ‘controllers/mahasiswa.js’

Gambar 12 menunjukkan file 'controllers/mahasiswa.js' setelah penambahan kode untuk melayani end-point **post('mahasiswa').** Sedangkan gambar 13 menunjukkan file 'routers/mahasiswa.js' setelah pemisahan dan penambahan pemanggilan controllers. Gambar 14 menunjukkan hasil uji coba di Postman.



Gambar 13. pemanggilan controller ***create*** *pada 'routers/mahasiwa.js'*



Gambar 14. Hasil uji coba end-point ***post('/mahasiswa')***

Gambar 15 menunjukkan file 'controllers/mahasiswa.js' setelah penambahan kode untuk melayani end-point ***put('mahasiswa/:nim')***. Sedangkan gambar 16 menunjukkan file 'routers/mahasiswa.js' setelah pemisahan dan penambahan pemanggilan controllers. Gambar 17 menunjukkan hasil uji coba di Postman.

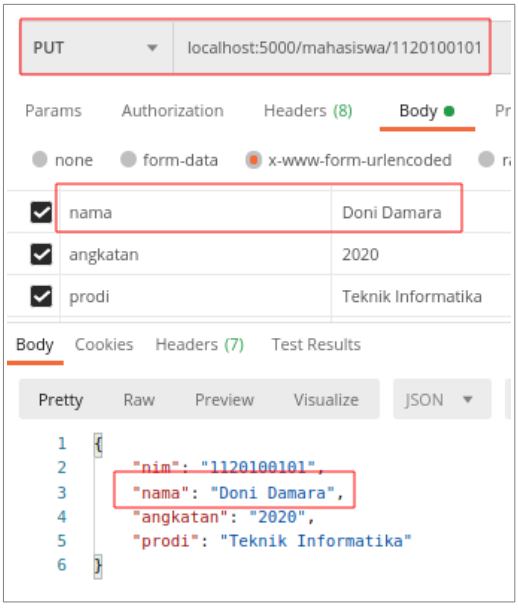
```
    create: (req, res) => {…
    },

    update: (req, res) => {
        const nim = req.params.nim;
        const mhs = req.body;
        const qstring = `UPDATE mahasiswa
                    SET nama = '${mhs.nama}', angkatan = '${mhs.angkatan}', prodi = '${mhs.prodi}'
                    WHERE nim = '${nim}'`
        connection.query(qstring, (err, data) => {
            if (err) {
                res.status(500).send({
                    message: "Error updating mahasiswa with NIM " + nim
                });
            }
            else if (data.affectedRows == 0) {
                res.status(404).send({
                    message: `Not found mahasiswa with NIM ${nim}.`
                });
            }
            else {
                console.log("updated mahasiswa: ", { nim: nim, ...mhs });
                res.send({ nim: nim, ...mhs });
            }
        })
    },
```

Gambar 15. update pada 'controllers/mahasiswa.js'

```
{} mahasiswa.js routers ●   {} mahasiswa.js controllers   {} db.js   {} i
routers > {} mahasiswa.js > ...
1   const express = require('express')
2   const routerMhs = express.Router()
3   const ctrMhs = require('../controllers/mahasiswa')
4
5   routerMhs.get('/mahasiswa',ctrMhs.getMhs)
6   routerMhs.get('/mahasiswa/:nim',ctrMhs.getMhsByNim)
7   routerMhs.post('/mahasiswa',ctrMhs.create)
8   routerMhs.put('/mahasiswa/:nim',ctrMhs.update)
9
```

Gambar 16. pemanggilan controller **update** pada 'routers/mahasiwa.js'



Gambar 17. Hasil uji coba end-point **put('/mahasiswa/:nim')**

Gambar 18 menunjukkan file 'controllers/mahasiswa.js' setelah penambahan kode untuk melayani end-point ***delete('mahasiswa/:nim')***. Sedangkan gambar 19 menunjukkan file 'routers/mahasiswa.js' setelah pemisahan dan penambahan pemanggilan controllers. Gambar 20 menunjukkan hasil uji coba di Postman.

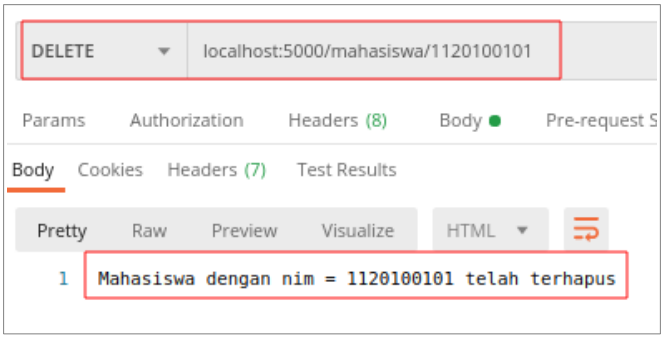
```
update: (req, res) => {…
},

delete: (req, res) => {
    const nim = req.params.nim
    const qstring = `DELETE FROM mahasiswa WHERE nim = '${nim}'`
    connection.query(qstring, (err, data) => {
        if (err) {
            res.status(500).send({
                message: "Error deleting mahasiswa with NIM " + nim
            });
        }
        else if (data.affectedRows == 0) {
            res.status(404).send({
                message: `Not found mahasiswa with NIM ${nim}.`
            });
        }
        else res.send(`Mahasiswa dengan nim = ${nim} telah terhapus`)
    });
}
```

Gambar 18. delete pada 'controllers/mahasiswa.js'

```
routers > {} mahasiswa.js > ...
1   const express = require('express')
2   const routerMhs = express.Router()
3   const ctrMhs = require('../controllers/mahasiswa')
4
5   routerMhs.get('/mahasiswa',ctrMhs.getMhs)
6   routerMhs.get('/mahasiswa/:nim',ctrMhs.getMhsByNim)
7   routerMhs.post('/mahasiswa',ctrMhs.create)
8   routerMhs.put('/mahasiswa/:nim',ctrMhs.update)
9   routerMhs.delete('/mahasiswa/:nim',ctrMhs.delete)
10
11  module.exports = routerMhs
12
```

Gambar 17. pemanggilan controller delete pada 'routers/mahasiwa.js'



Gambar 17. Hasil uji coba end-point put('/mahasiswa/:nim')

D. Kesimpulan

1. Setelah dilakukan perbaikan, struktur aplikasi menjadi lebih rapi sehingga lebih mudah untuk dilakukan perawatan.
2. Kode program pada setiap file hasil perubahan, terlihat lebih lebih simple sehingga lebih mudah dibaca.
3. Program tetap berjalan dengan baik seperti semula.

E. Tugas

Buatlah controllers untuk matakuliah dan nilai, sebagaimana yang telah dilakukan pada controller mahasiswa.