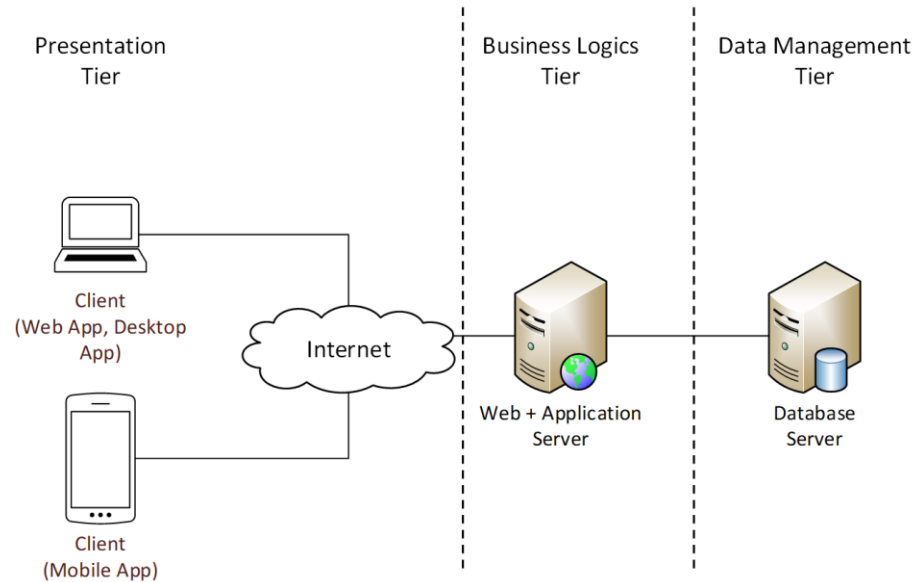


Backend

Pengantar

1. Three Tier Architecture



Gambar 1. 3 Tier Architecture

Arsitektur yang biasa digunakan untuk membangun software atau solusi atau sistem adalah 3 tier seperti pada Gambar 1 dimana sistem dibagi menjadi 3 bagian atau lapisan yaitu:

- Data management adalah bagian ini bertujuan mengelola data yang umumnya terdapat database server.
- Business logic adalah bagian yang bertujuan memberikan layanan sesuai dengan domain masalah. Pada bagian terdapat fungsi-fungsi yang mengakses data pada tier data management dan logika. Tujuan dari bagian ini adalah:
 - o Agar aplikasi tidak mengakses langsung data pada database.
 - o Agar aplikasi-aplikasi yang menggunakan bahasa pemrograman yang berbeda dapat mengakses data dengan cara yang sama.

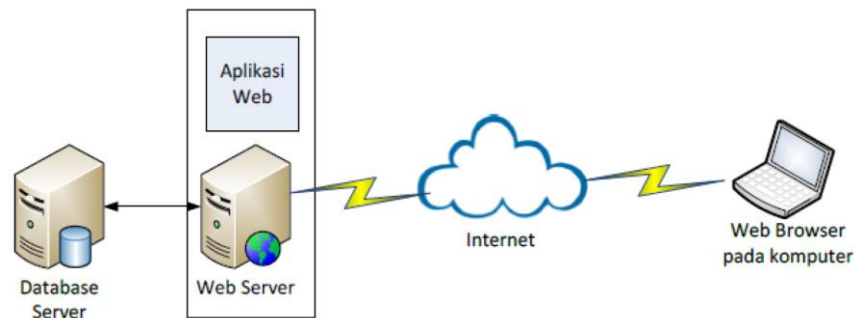
Bagian ini juga dapat disebut sebagai middleware. Pada perkembangannya, bagian ini dapat dibuat untuk mengakses resource lain seperti mengakses fungsi pada software atau mengontrol hardware.

- Presentation adalah bagian antarmuka yang digunakan oleh pengguna. Pada bagian ini terdapat form yang bisa digunakan oleh pengguna untuk input data, kontrol untuk memanggil dan menggunakan fungsi-fungsi pada tier business logics. Bagian ini dapat dibangun dengan teknologi web, desktop atau mobile.

2. Web Service

Teknologi yang digunakan sebagai business logic tier atau middleware menggunakan aplikasi web. Hal ini disebabkan karena aplikasi ini mempunyai sifat untuk dapat diakses di mana saja dan kapan saja.

Sebagai pengantar bagaimana aplikasi web bekerja. Aplikasi web yang berada pada web server akan dapat diakses oleh komputer yang digunakan oleh pengguna dengan web browser seperti Internet Explorer. Komunikasi yang terjadi antara web browser pada komputer dengan web server adalah dengan dua proses, yaitu request dan response.



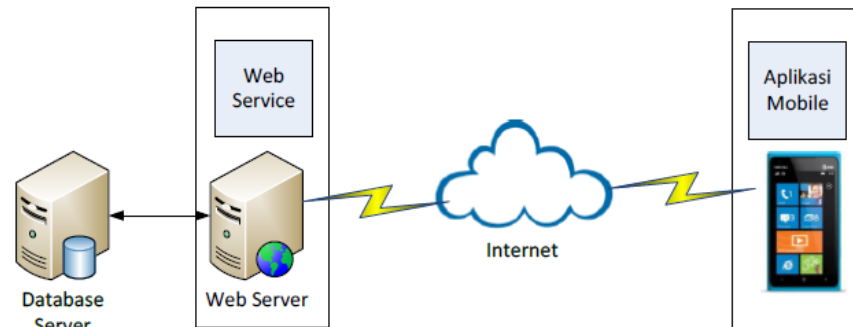
Gambar 2. Aplikasi web.

Request adalah proses yang dilakukan oleh web browser pada komputer ke web server, kemudian permintaan tersebut akan diolah oleh web server dan hasilnya akan dikirimkan ke web browser yang merupakan proses response dari web server. Hasil proses response ini lah yang akan kita lihat dalam bentuk halaman web pada web browser. Misal user ingin melihat data mahasiswa pada web browser, maka yang akan dilakukan adalah akan ada proses request ke web server. Permintaan tersebut akan diproses pada web server, karena ada kebutuhan akan data mahasiswa maka aplikasi web pada sisi server akan mengambil data mahasiswa tersebut pada database kemudian hasilnya akan dikirimkan ke web browser sebagai proses response.

Dari paparan tersebut maka didapat gambaran jika aplikasi web itu adalah aplikasi yang disimpan di server, bagi pengguna yang ingin menggunakannya tidak perlu menginstall apapun di komputer miliknya. Pengguna cukup menginstall web browser, kemudian pengguna cukup mengetikkan alamat dari web server pada web browser tersebut, dan selanjutnya proses yang terjadi seperti yang telah dipaparkan di atas.

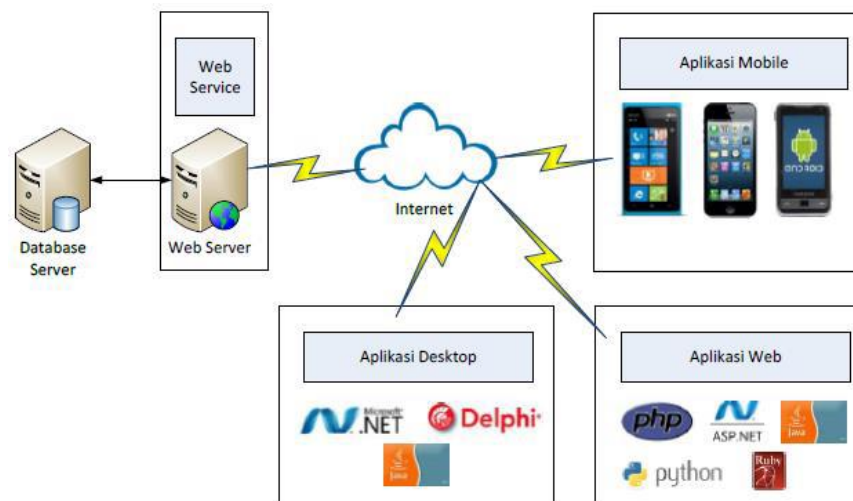
Bagaimana kira-kira cara kerja aplikasi mobile (bukan : web browser pada perangkat mobile) jika ingin melakukan hal di atas? Aplikasi mobile memang sangat beragam, ada game, ada aplikasi tanpa perlu koneksi internet atau aplikasi yang terhubung jaringan internet untuk bisa digunakan, seperti aplikasi mobile twitter atau facebook.

Pada Gambar 3 memberikan ilustrasi cara kerja aplikasi mobile yang membutuhkan koneksi internet.



Gambar 3. Aplikasi mobile.

Dari gambar di atas dapat dilihat letak dari aplikasi mobile yang berada pada perangkat mobile itu sendiri, ini artinya aplikasi mobile harus diinstall pada perangkat mobile tersebut. Misalnya aplikasi mobile yang diinstall adalah aplikasi mobile Facebook, maka dipastikan akan ada halaman yang berfungsi untuk login atau halaman untuk menampilkan status-status yang ditulis. Pada proses login akan dilakukan pengecekan username dan password yang dimasukkan untuk dicocokkan dengan data yang ada di database. Begitu juga jika pada halaman ingin ditampilkan daftar status yang telah di tulis pada wall, maka perlu ada proses untuk mengambil data tersebut pada database. Tetapi jika dilihat pada gambar di atas, proses pengambilan data tersebut tentunya tidak bisa dilakukan langsung dari aplikasi mobile ke database server. Pada gambar terdapat layanan berbasis web (web service) yang berada pada web server, pada layanan tersebut terdapat fungsi-fungsi yang bertugas untuk melakukan hal-hal penting termasuk fungsi untuk mengakses data pada database server atau fungsi untuk otentikasi user untuk proses login. Maka aplikasi mobile cukup mengakses fungsi-fungsi pada layanan tersebut untuk melakukan proses-proses tersebut



Gambar 4. Web service.

Keberadaan web service sebagai perantara seperti gambar di atas mempunyai beberapa manfaat, diantaranya adalah:

- Layanan ini dapat lebih mengamankan data pada database server, hal ini dikarena database tidak langsung diakses oleh aplikasi.

- Layanan ini bukan hanya dapat diakses oleh aplikasi mobile, tapi juga dapat diakses oleh aplikasi jenis lain seperti aplikasi web atau aplikasi desktop. Selain itu layanan berbasis web ini akan dapat diakses dan dimanfaatkan pada berbagai platform atau teknologi, artinya layanan ini bisa diakses oleh aplikasi yang dibangun dengan Node JS, .NET, Java, PHP, Python, Ruby, dan lain-lain.

Dari paparan di atas maka dapat digambarkan peran dari web service seperti pada Gambar 4.

3. REST API

Teknologi web yang dapat digunakan untuk membuat layanan web (web service) ini salah satunya adalah REST API yang akan dijelaskan pada hands on labs ini. REST atau Representation State Transfer adalah salah satu gaya arsitektur yang memanfaatkan protokol HTTP. Penjelasan hal ini akan diberikan pada bagian Instruksi saat telah diberikan contoh kode dari REST API.

REST API yang akan dipelajari menggunakan bahasa pemrograman JavaScript (Node JS). Sedangkan pada data management tier akan digunakan database server MySQL atau MariaDB.

Instruksi

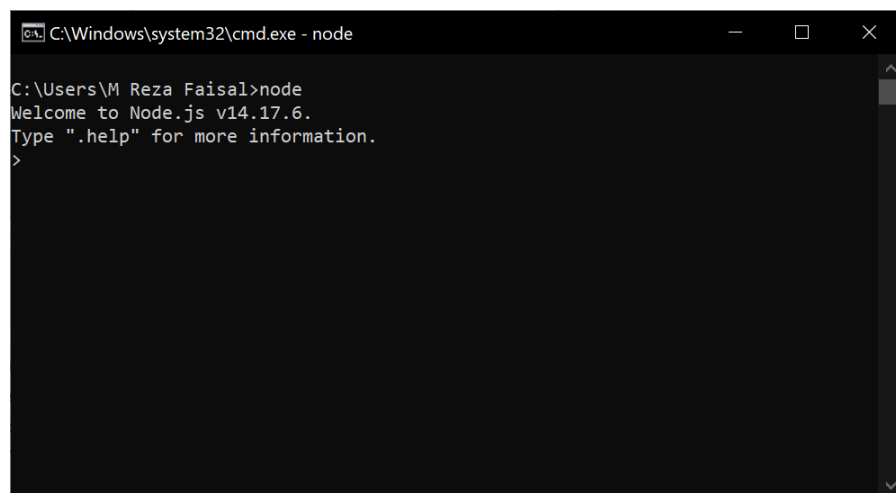
1. Node.js

Sebelum memulai hands on labs ini pastikan Node JS. Node.js adalah runtime environment untuk JavaScript yang bersifat open-source dan cross-platform. Dengan Node.js kita dapat menjalankan kode JavaScript di mana pun, tidak hanya terbatas pada lingkungan browser.

Jika belum menginstall Node.js maka file installernya dapat didownload pada link berikut <https://nodejs.org/en/>. Setelah proses instalasi selesai maka perlu dilakukan pengujian apakah file executable dari Node.js dapat dijalankan pada command prompt. Pertama jalankan command prompt kemudian ketikkan perintah ini.

```
node
```

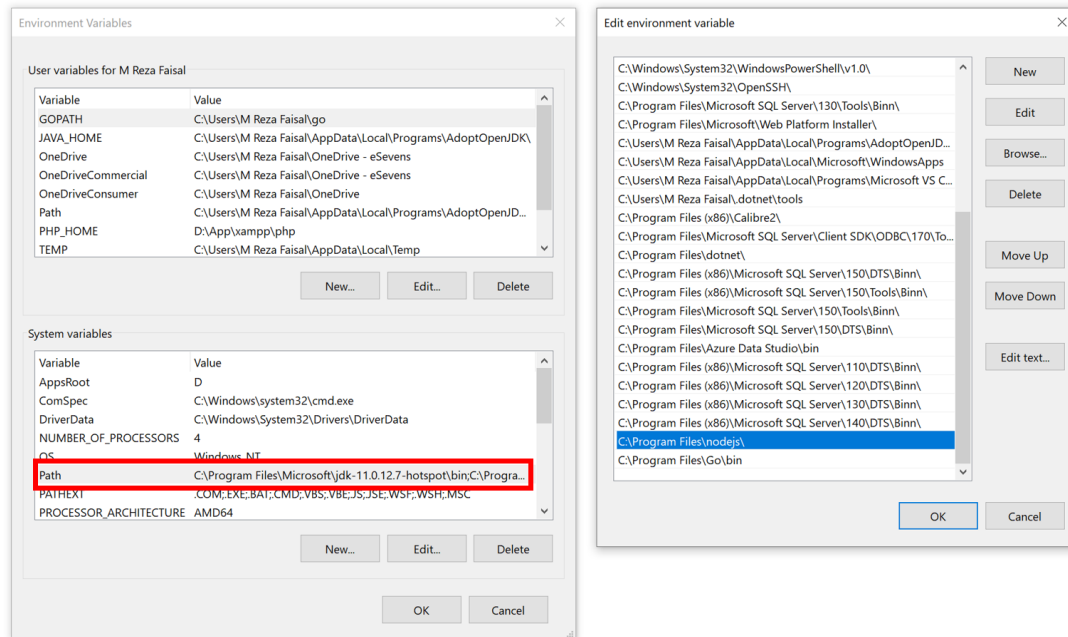
Dan hasilnya dapat dilihat pada Gambar 5.



```
C:\Windows\system32\cmd.exe - node
C:\Users\M Reza Faisal>node
Welcome to Node.js v14.17.6.
Type ".help" for more information.
>
```

Gambar 5. Node.js.

Jika langkah di atas tidak berhasil maka pastikan pada System variables, bagian Path (kotak merah pada Gambar 6) telah berisi nilai lokasi atau folder instalasi Node.js seperti yang terlihat pada kotak biru.



Gambar 6. Path pada System variables.

Selain itu juga perlu diuji apakah perintah untuk menjalankan Node Package Manager dengan mengetikkan perintah berikut pada command prompt.

```
npm
```

Jika berhasil masih maka hasilnya dapat dilihat seperti pada Gambar 7.

```
C:\Windows\system32\cmd.exe

C:\Users\M Reza Faisal>npm

Usage: npm <command>

where <command> is one of:
  access, adduser, audit, bin, bugs, c, cache, ci, cit,
  clean-install, clean-install-test, completion, config,
  create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
  edit, explore, fund, get, help, help-search, hook, i, init,
  install, install-ci-test, install-test, it, link, list, ln,
  login, logout, ls, org, outdated, owner, pack, ping, prefix,
  profile, prune, publish, rb, rebuild, repo, restart, root,
  run, run-script, s, se, search, set, shrinkwrap, star,
  stars, start, stop, t, team, test, token, tst, un,
  uninstall, unpublish, unstar, up, update, v, version, view,
  whoami

npm <command> -h  quick help on <command>
npm -l           display full usage info
npm help <term>  search for help on <term>
npm help npm     involved overview

Specify configs in the ini-formatted file:
  C:\Users\M Reza Faisal\.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@6.14.15 C:\Program Files\nodejs\node_modules\npm
```

Gambar 7. npm.

2. Express

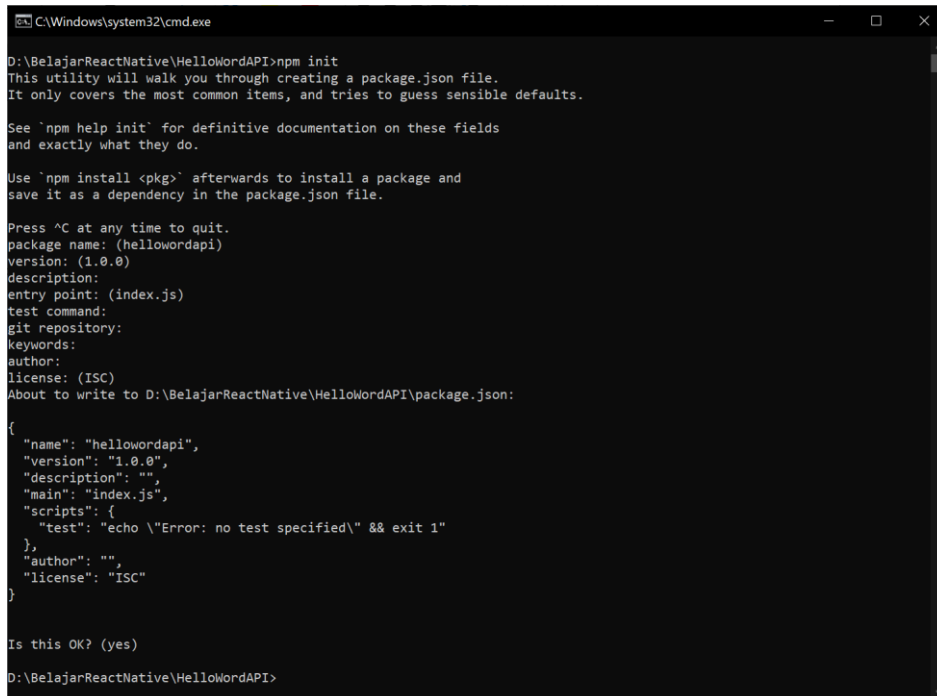
Seperti yang telah Node.js adalah fondasi dalam membangun aplikasi. Pada hands on labs sebelumnya telah diperkenalkan React Native yang merupakan framework untuk membangun aplikasi mobile. Selain itu juga dikembangkan framework lain untuk keperluan yang lain. Sebagai contoh adalah framework JS untuk mengembangkan aplikasi web pada umumnya dan REST API pada khususnya. Salah satu framework yang dapat digunakan untuk membuat REST API dengan mudah adalah Express (<https://expressjs.com/>).

Framework ini perlu ditambahkan pada project. Berikut adalah langkah-langkah untuk membuat project dan menambahkan framework Express kedalamnya. Langkah pertama adalah membuat folder project. Misal buat folder dengan nama HelloWorldAPI. Pada percobaan ini dibuat folder ini di dalam folder D:\BelajarReactNative.

Langkah berikutnya adalah membuat file package.json dengan menggunakan secara otomatis dengan menggunakan perintah berikut saat berada di dalam folder HelloWorldAPI.

```
npm init
```

Ikuti isian yang telah disediakan dengan menekan tombol Enter untuk setiap pertanyaan, seperti yang terlihat pada Gambar 8.



```
C:\Windows\system32\cmd.exe
D:\BelajarReactNative\HelloWorldAPI>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: (helloworldapi)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\BelajarReactNative\HelloWorldAPI\package.json:

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

Is this OK? (yes)
y
D:\BelajarReactNative\HelloWorldAPI>
```

Gambar 8. Membuat package.json.

Berikut adalah isi dari file package.json. Jika ingin mengganti nilai dari masing-masing atribut pada file ini dapat dilakukan dengan menggunakan text editor.

Kode 1. package.json

```
package.json
```

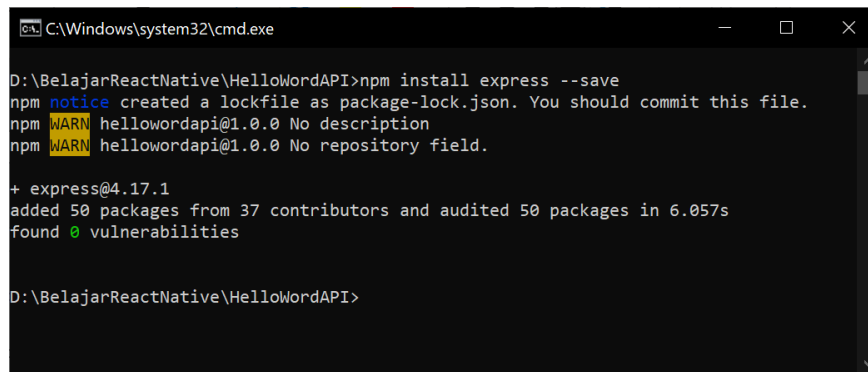
```
1 {
```

```
2  "name": "helloworldapi",
3  "version": "1.0.0",
4  "description": "",
5  "main": "index.js",
6  "scripts": {
7    "test": "echo \"Error: no test specified\" && exit 1"
8  },
9  "author": "",
10 "license": "ISC"
11 }
```

Langkah selanjutnya menambahkan framework Express ke dalam folder HelloWorldAPI dengan mengetikkan perintah berikut.

```
npm install express --save
```

Proses dan hasil dari perintah tersebut dapat dilihat pada Gambar 9.



```
C:\Windows\system32\cmd.exe
D:\BelajarReactNative\HelloWorldAPI>npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN helloworldapi@1.0.0 No description
npm WARN helloworldapi@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 6.057s
found 0 vulnerabilities

D:\BelajarReactNative\HelloWorldAPI>
```

Gambar 9. Install framework Express pada project.

Hasil dari perintah di atas dapat dilihat jika pada folder HelloWorldAPI ditambah folder node_modules yang didalamnya berisi folder express yang merupakan package dari framework Express. Selain itu juga ditambahkan package-package pendukung framework ini di dalam folder node_modules.

Langkah selanjutnya adalah membuat kode program yang merupakan implementasi dari framework ini. Nama file utama yang mesti dibuat adalah index.js, hal ini sesuai dengan file package.json baris ke-5. Maka tambahkan file index.js ke dalam folder HelloWorldAPI dan tambahkan kode seperti kode di bawah ini. Penjelasan tentang kode ini akan diberikan pada sub bab selanjutnya.

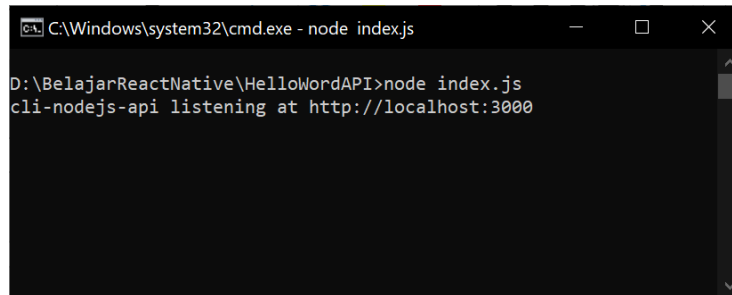
Kode 2. index.js

```
index.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.get('/', (req, res) => {
6    res.send('Hello World!')
7  });
8
9  app.listen(port, () => {
10   console.log(`cli-nodejs-api listening at http://localhost:${port}`)
11 });
```

Selanjutnya pada command prompt ketikan perintah berikut.

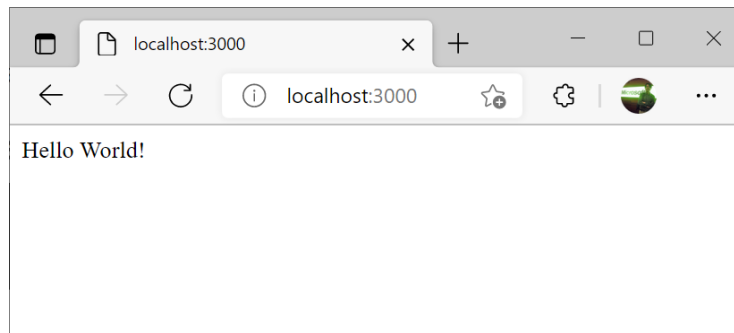
```
node index.js
```

Jika langkah-langkah sebelumnya sukses dilakukan maka dapat dilihat hasil seperti pada Gambar 10.



Gambar 10. Menjalankan index.js.

Untuk mengujinya secara cepat dan mudah dapat dilakukan dengan membuka web browser kemudian ketikan alamat <http://localhost:3000> hasilnya dapat dilihat seperti pada Gambar 11.



Gambar 11. Mengakses Web API dengan web browser.

Install package pendukung yang nantinya akan digunakan pada sub bab berikutnya. Dan hasilnya dapat dilihat pada Gambar 12.

Kode 3. Instalasi package atau modul pendukung.

```
npm install cors --save  
npm install body-parser --save  
npm install mysql --save
```



```
C:\Windows\system32\cmd.exe
D:\BelajarReactNative\HelloWordAPI>npm install cors --save
npm WARN helloworldapi@1.0.0 No description
npm WARN helloworldapi@1.0.0 No repository field.

+ cors@2.8.5
added 2 packages from 2 contributors and audited 52 packages in 1.691s
found 0 vulnerabilities

D:\BelajarReactNative\HelloWordAPI>npm install body-parser --save
npm WARN helloworldapi@1.0.0 No description
npm WARN helloworldapi@1.0.0 No repository field.

+ body-parser@1.19.0
updated 1 package and audited 53 packages in 1.634s
found 0 vulnerabilities

D:\BelajarReactNative\HelloWordAPI>npm install mysql --save
npm WARN helloworldapi@1.0.0 No description
npm WARN helloworldapi@1.0.0 No repository field.

+ mysql@2.18.1
added 9 packages from 14 contributors and audited 61 packages in 2.071s
found 0 vulnerabilities

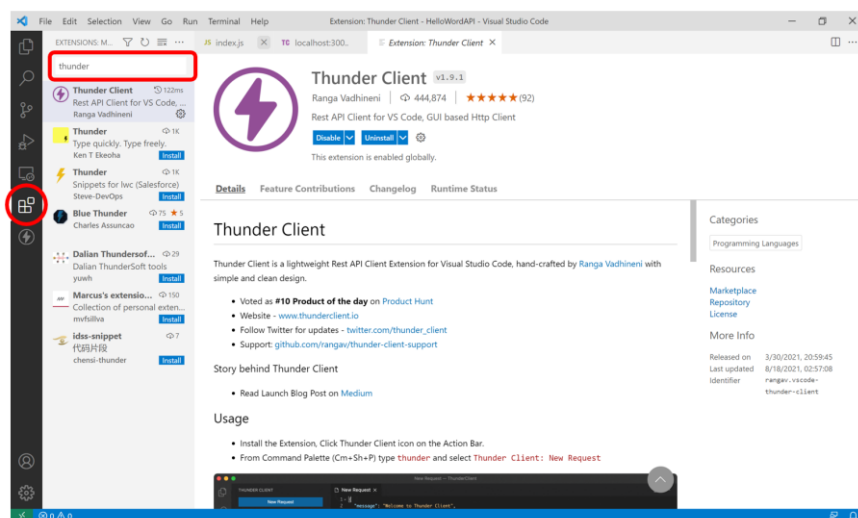
D:\BelajarReactNative\HelloWordAPI>
```

Gambar 12. Instalasi package pendukung.

3. REST API Client

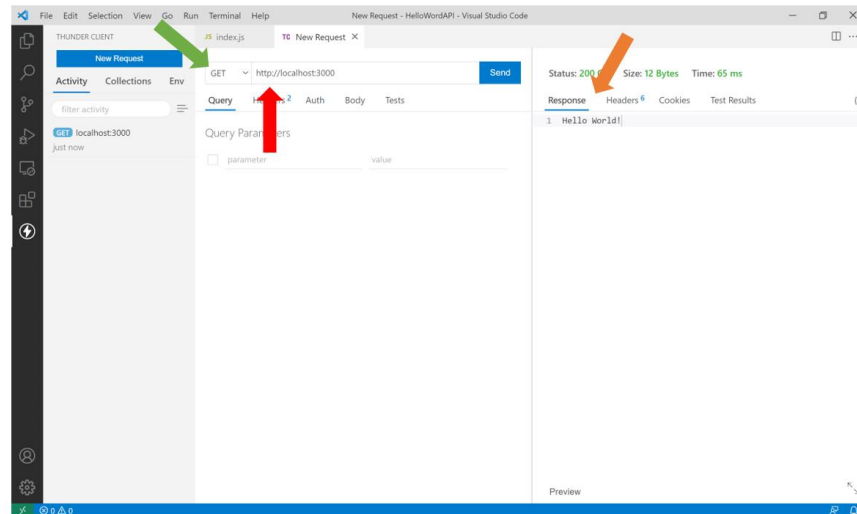
Pada sub bab di atas untuk mengakses REST API yang telah dibuat dengan menggunakan web browser. Tetapi web browser memiliki keterbatasan jika digunakan untuk menguji REST API pada tahap pengembangan. Untuk mengakses REST API pada saat tahap pengembangan biasanya digunakan adalah aplikasi dengan nama Postman yang dapat diunduh di <https://www.postman.com/>.

Namun pada hands on labs ini akan digunakan Thunder Client yang merupakan extension pada Visual Studio Code. Pada Visual Studio Code klik tombol Extension (lingkaran merah pada Gambar 13) kemudian pada kotak pencarian (kotak merah) ketikkan thunder. Kemudian pilih Thunder Client pada daftar pencarian kemudian klik tombol install.



Gambar 13. Thunder client.

Setelah extension tersebut diinstall maka dapat dilihat tombol Thunder Client yang berada di bawah tombol Extension. Setelah tombol diklik maka dapat dilihat antarmukanya seperti pada Gambar 14.



Gambar 14. New Request.

Klik tombol New Request jika ingin membuat request baru, kemudian ketikkan alamat <http://localhost:3000> (ini adalah alamat REST API yang telah dibuat pada sub bab sebelumnya) pada kotak yang ditunjuk oleh panah merah pada Gambar 14. Kemudian secara default HTTP verb atau method dipilih adalah GET seperti yang ditunjuk oleh panah hijau pada Gambar 14. Kemudian klik tombol Send maka hasilnya dapat dilihat pada tab Response yang ditunjuk pada panah orange di gambar di atas.

Penjelasan mengenai HTTP verb atau method akan dijelaskan pada sub bab berikutnya.

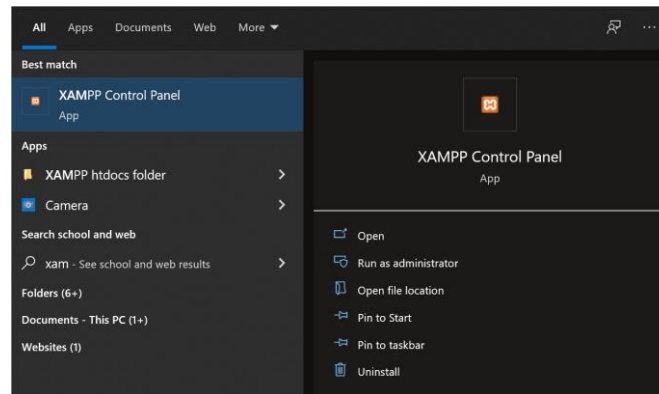
4. XAMPP

XAMPP adalah adalah paket instalasi yang berisi MariaDB, PHP, Perl dan Apache.

Instalasi

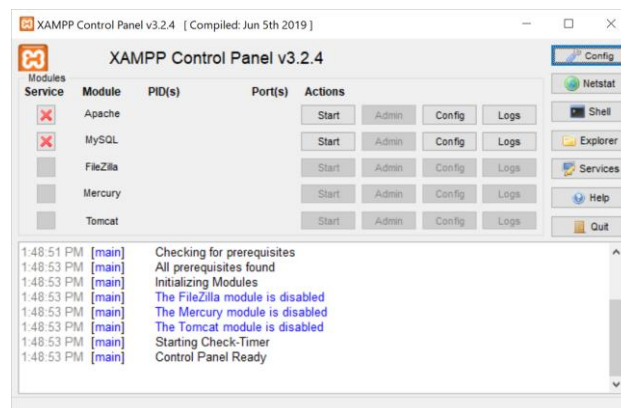
Untuk mendownload paket ini dapat mengunjungi alamat berikut <https://www.apachefriends.org/download.html>. Download paket sesuai dengan sistem operasi yang digunakan. XAMPP digunakan pada hands on labs ini untuk memanfaatkan database server MariaDB. Selain itu juga memanfaatkan aplikasi web PHPMyAdmin untuk mengelola database.

Setelah instalasi selesai maka jalankan XAMPP Control Panel dengan cara memilih pada menu, kemudian pilih Run as administrator.



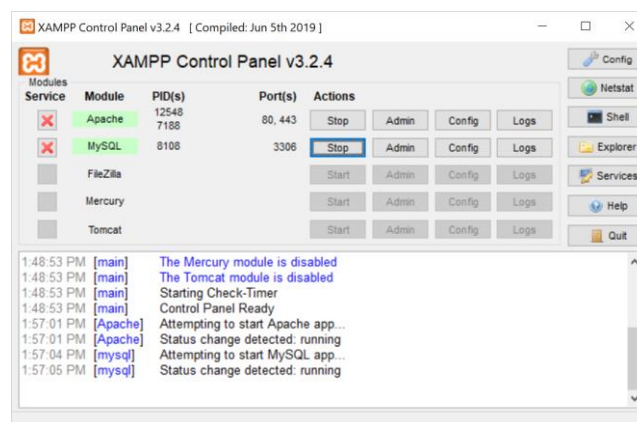
Gambar 15. Menu XAMPP Control Panel.

Maka dapat dilihat window XAMPP Control Panel seperti pada Gambar 16.



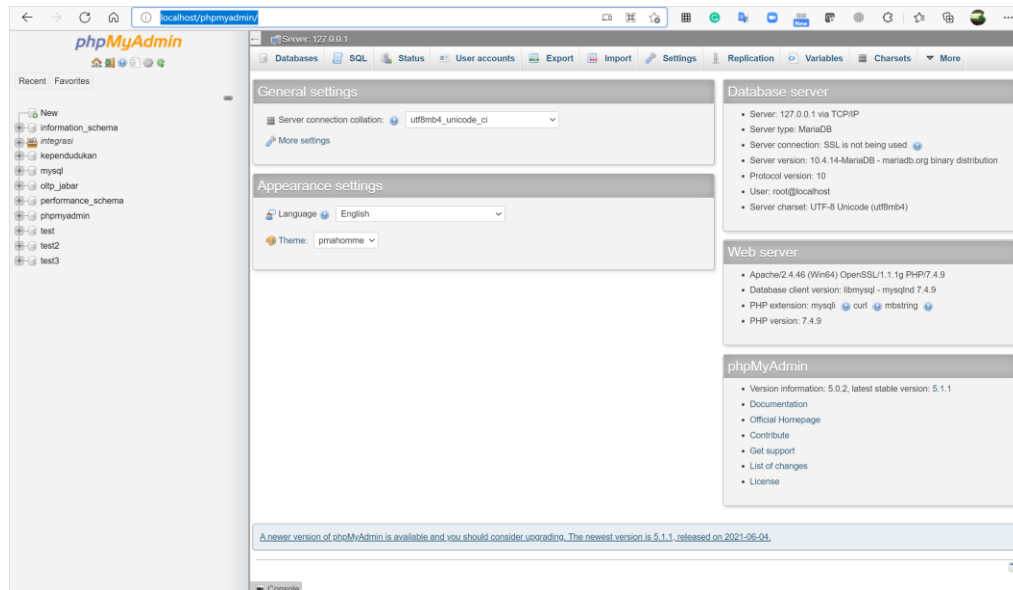
Gambar 16. XAMPP Control Panel – Status service mati.

Pada gambar di atas layanan (service) web server Apache dan database server MySQL (MariaDB) belum dihidupkan. Klik tombol Start pada baris Apache dan MySQL sehingga jika service jalan dapat dilihat tampilan seperti pada Gambar 17.



Gambar 17. XAMPP Control Panel – Status service hidup.

Setelah kedua service hidup maka buka web browser kemudian ketikkan alamat berikut pada kolom alamat berikut <http://localhost/phpmyadmin/> dan hasilnya dapat dilihat seperti pada Gambar 18.



Gambar 18. phpMyAdmin.

Setup Data

Pada hands on labs diperlukan sebuah database dan sebuah tabel. Buat database dengan nama adalah `social_media` dengan menggunakan phpMyAdmin. Kemudian buat tabel dengan nama `users` dengan atribut sebagai berikut:

- `username`.
- `password`.
- `fullname`.
- `picture`.

Untuk mempermudah membuat tabel dapat digunakan kode SQL berikut ini.

users.sql	
1	CREATE TABLE `users` (
2	`username` varchar(50) NOT NULL,
3	`password` varchar(256) NOT NULL,
4	`fullname` varchar(256) NOT NULL,
5	`picture` varchar(256) NOT NULL
6) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

5. Dasar-Dasar Express

Pada sub bab ini dijelaskan tentang kode `index.js` yang telah diberikan pada sub bab Express atau kembali dapat dilihat pada kode di bawah ini.

Kode 4.index.js

```
index.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.get('/', (req, res) => {
6    res.send('Hello World!')
7  });
8
9  app.listen(port, () => {
10    console.log(`cli-nodejs-api listening at http://localhost:${port}`)
11  });
```

Pada baris ke-1 dilakukan import modul express. Kemudian pada baris ke-2 dibuat instance atau obyek dari class express, nama obyeknya adalah app.

Pada baris ke-3 dibuat variable dengan nama `port` untuk menyimpan nomor port yang akan digunakan untuk mengakses service REST API ini. Nilai port ini digunakan untuk menghidupkan service dengan perintah yang ditulis pada kode yang dimulai pada baris ke-9. Nilai yang diberikan pada variable `port` adalah 3000 sehingga dapat dilihat cara untuk mengakses service ini dilakukan dengan menulis alamat sebagai berikut <http://localhost:3000>. Jika nilai port yang digunakan adalah 4000 maka alamat yang digunakan untuk mengakses service adalah sebagai berikut <http://localhost:4000>.

Sedangkan pada baris ke-5 sampai dengan baris ke-7 adalah cara untuk membuat API endpoint dengan HTTP verb atau method adalah GET. Fungsi yang digunakan untuk membuat API endpoint GET adalah `app.get()`. Method ini memiliki dua parameter input, yaitu:

- URL pattern untuk mengakses API endpoint. Pada kode di baris ke-5, nilai dari parameter input ini adalah `'/'`. Jika nilainya adalah `'/api'` maka alamat yang digunakan untuk mengakses endpoint ini menjadi <http://localhost:3000/api>.
- (`req`, `res`) merupakan obyek yang merepresentasikan cara komunikasi aplikasi web yaitu request (`req`) dan response (`res`). Selanjutnya dapat dilihat pada baris ke-6 digunakan method `send` dari obyek `res` untuk memberikan respon jika endpoint ini diakses.

6. HTTP Method

Pada sub bab sebelumnya telah diberikan contoh pembuatan endpoint yang diakses dengan menggunakan HTTP method GET. Pada sub bab ini akan diperkenalkan HTTP method yang lain dan cara pembuatannya dengan menggunakan framework Express. HTTP method tersebut adalah:

- GET, umumnya method ini digunakan untuk mendapatkan suatu nilai baik berupa nilai tunggal atau collection.
- POST, method ini digunakan untuk mengirimkan nilai, biasanya method ini digunakan untuk menyimpan data.
- PUT, method ini digunakan untuk mengupdate nilai, sehingga biasanya method ini digunakan untuk mengupdate data.

- DELETE, sesuai dengan namanya digunakan menghapus, sehingga biasanya method ini digunakan untuk menghapus data.

Pada kode berikut ini contoh kode yang berisi endpoint dengan menggunakan empat HTTP method tersebut.

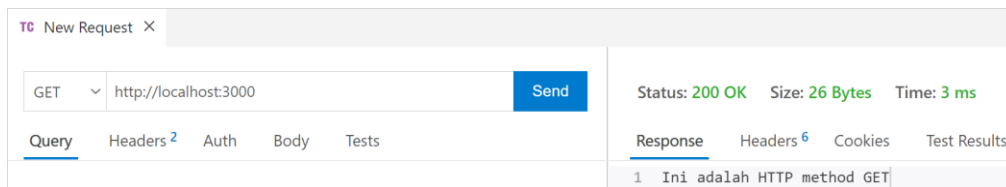
Kode 5. *http_method.js*

```
http_method.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.get('/', (req, res) => {
6    res.send('Ini adalah HTTP method GET')
7  });
8
9  app.post('/', (req, res) => {
10   res.send('Ini adalah HTTP method POST')
11 });
12
13 app.put('/', (req, res) => {
14   res.send('Ini adalah HTTP method PUT')
15 });
16
17 app.delete('/', (req, res) => {
18   res.send('Ini adalah HTTP method DELETE')
19 });
20
21 app.listen(port, () => {
22   console.log(`cli-nodejs-api listening at http://localhost:${port}`)
23 });
```

Matikan service yang sebelumnya jika telah ada service yang jalan pada port 3000. Kemudian jalankan file ini dengan menggunakan perintah.

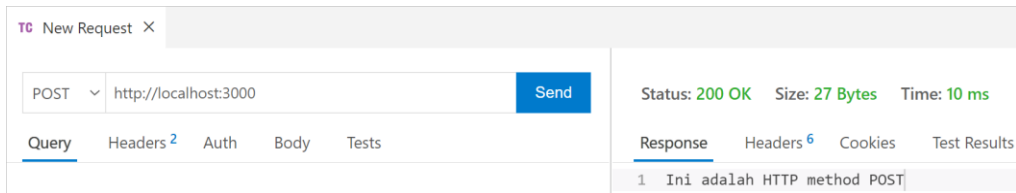
```
node http_method.js
```

Selanjutnya akan digunakan Thunder Client pada Visual Studio Code untuk mengakses layanan ini. Langkah pertama adalah memilih HTTP method GET seperti yang terlihat pada Gambar 19.



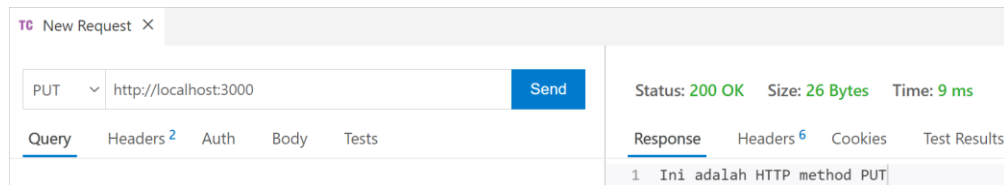
Gambar 19. HTTP Get.

Kemudian ganti method GET menjadi POST seperti yang terlihat pada Gambar 20. Dan dilanjutkan dengan menekan tombol Send.



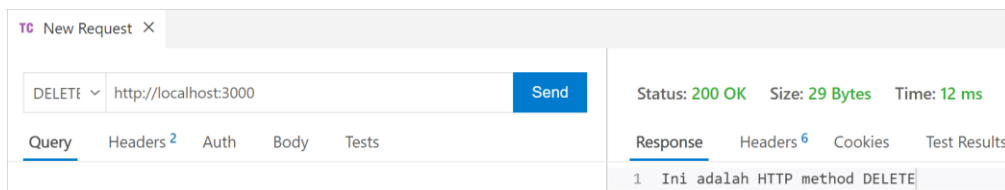
Gambar 20. HTTP Post.

Percobaan ketika adalah menggunakan method PUT seperti yang terlihat pada Gambar 21.



Gambar 21. HTTP Put.

Dan yang terakhir adalah mengakses endpoint method DELETE dengan hasil yang dapat dilihat seperti pada Gambar 22.



Gambar 22. HTTP Delete.

Dari sub bab ini maka telah diketahui empat method HTTP utama dan cara pembuatan endpointnya dengan menggunakan framework Express.

7. Routing

Routing adalah penentuan cara bagaimana aplikasi merespon permintaan client ke endpoint tertentu yang berupa URL. Routing memungkinkan kita untuk menentukan method HTTP apa yang digunakan untuk mengakses suatu endpoint. Selain itu juga dapat ditentukan path untuk mengakses suatu endpoint.

Dari penjelasan pada sub bab di atas, dapat diketahui cara pembuatan endpoint berdasarkan method HTTP. Sehingga jika disimpulkan dapat dilihat pada daftar fungsi di bawah ini:

- `app.get()` untuk membuat endpoint dengan cara akses via HTTP method GET.
- `app.post()` untuk membuat endpoint dengan cara akses via HTTP method POST.
- `app.put()` untuk membuat endpoint dengan cara akses via HTTP method PUT.
- `app.delete()` untuk membuat endpoint dengan cara akses via HTTP method DELETE.

Dimana `app` adalah obyek yang dibuat dari class `express`. Seperti diketahui bahwa keempat method di atas memiliki dua parameter input. Dimana parameter input pertama menentukan path untuk mengakses sebuah endpoint. Pada Kode 5 dapat dilihat path yang digunakan untuk mengakses keempat endpoint

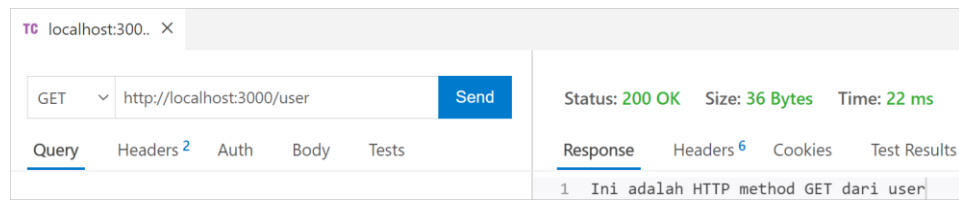
adalah '/'. Sehingga dapat dilihat pada Gambar 19, Gambar 20, Gambar 21, dan Gambar 22 untuk mengakses semua endpoint tersebut cukup dengan menggunakan alamat <http://localhost:3000>.

Sebagai contoh pada file `http_method.js` ditambahkan beberapa method seperti yang dapat dilihat pada Kode 6 mulai baris ke-21 sampai baris ke-35.

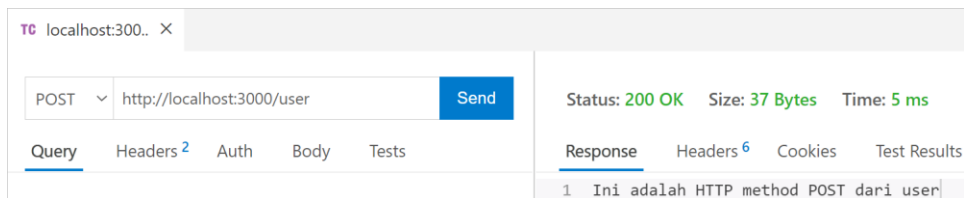
Kode 6. `http_method.js`

```
http_method.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.get('/', (req, res) => {
6    res.send('Ini adalah HTTP method GET')
7  });
8
9  app.post('/', (req, res) => {
10   res.send('Ini adalah HTTP method POST')
11 });
12
13 app.put('/', (req, res) => {
14   res.send('Ini adalah HTTP method PUT')
15 });
16
17 app.delete('/', (req, res) => {
18   res.send('Ini adalah HTTP method DELETE')
19 });
20
21 app.get('/user', (req, res) => {
22   res.send('Ini adalah HTTP method GET dari user')
23 });
24
25 app.post('/user', (req, res) => {
26   res.send('Ini adalah HTTP method POST dari user')
27 });
28
29 app.put('/user', (req, res) => {
30   res.send('Ini adalah HTTP method PUT dari user')
31 });
32
33 app.delete('/user', (req, res) => {
34   res.send('Ini adalah HTTP method DELETE dari user')
35 });
36
37 app.listen(port, () => {
38   console.log(`cli-nodejs-api listening at http://localhost:${port}`)
39 });
```

Pada baris tersebut dapat dilihat digunakan path adalah '/user'. Sehingga untuk mengakses keempat endpoint tersebut digunakan cara seperti pada Gambar 23 atau pada Gambar 24.



Gambar 23. HTTP Get dengan Path adalah /user



Gambar 24. HTTP Post dengan Path adalah /user

8. Pengiriman Nilai

Pada sub bab di atas dapat dilihat bagaimana cara mengakses endpoint. Seperti fungsi pada umumnya fungsi mengeluarkan atau mengembalikan nilai seperti contoh-contoh di atas. Fungsi juga pada umumnya menerima nilai input untuk diolah. Sebagai contoh misal ada fungsi sebagai berikut.

```
function tambah(bilangan1, bilangan2) {  
  hasil = bilangan1 + bilangan2;  
}
```

Dapat dilihat pada fungsi terdapat nilai input yaitu `bilangan1` dan `bilangan2`. Kemudian dapat dilihat juga bagaimana kedua nilai input tersebut digunakan dalam operasi di dalam fungsi.

Secara konsep endpoint juga seperti fungsi yang dapat menerima nilai sebagai input. Ada beberapa cara pengiriman nilai ke endpoint sebagai nilai input.

Params

Berikut ini adalah contoh cara pengiriman value sebagai parameter dan cara mengambil nilai parameter tersebut.

Kode 7. parameter.js

```
parameter.js  
1  const express = require('express');  
2  const app = express();  
3  const port = 3000;  
4  
5  app.get('/:bilangan1/:bil1/bilangan2/:bil2', (req, res) => {  
6    res.send(req.params)  
7  });  
8  
9  app.get('/:bil1/:bil2', (req, res) => {  
10   res.send(req.params)
```

```
11 });  
12  
13 app.get('/tambah/bilangan1/:bil1/bilangan2/:bil2', (req, res) => {  
14     hasil = parseInt(req.params.bil1) + parseInt(req.params.bil2);  
15     res.json({"data": req.params, "result": hasil})  
16 });  
17  
18 app.get('/kurang/:bil1/:bil2', (req, res) => {  
19     hasil = parseInt(req.params.bil1) - parseInt(req.params.bil2);  
20     res.json({"data": req.params, "result": hasil})  
21 });  
22  
23 app.listen(port, () => {  
24     console.log(`cli-nodejs-api listening at http://localhost:${port}`)  
25 });
```

Dari Kode 7 dapat dilihat contoh penggunaan parameter. Pada baris ke-5 dapat dilihat bagaimana cara untuk mengakses endpoint sekaligus cara untuk menyisipkan nama parameter dan nilainya. Pada baris tersebut dapat dilihat path untuk mengakses endpoint adalah / sehingga alamat yang dituliskan adalah sebagai berikut.

```
http://localhost:3000/
```

Selanjutnya nama parameter pertama adalah `bil1` yang mana nanti nilainya disisipkan pada bagian `:bil1`, sehingga jika ingin menyisipkan nilai 4 maka untuk mengakses endpoint menjadi seperti berikut.

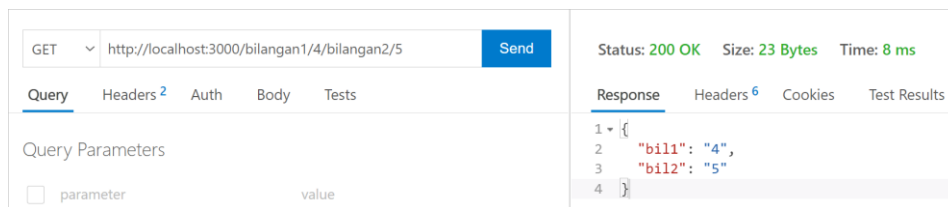
```
http://localhost:3000/bilangan1/4
```

Selanjutnya nama parameter pertama adalah `bil2` yang mana nanti nilainya disisipkan pada bagian `:bil2`, sehingga jika ingin menyisipkan nilai 5 maka untuk mengakses endpoint menjadi seperti berikut

```
http://localhost:3000/bilangan1/4/bilangan2/5
```

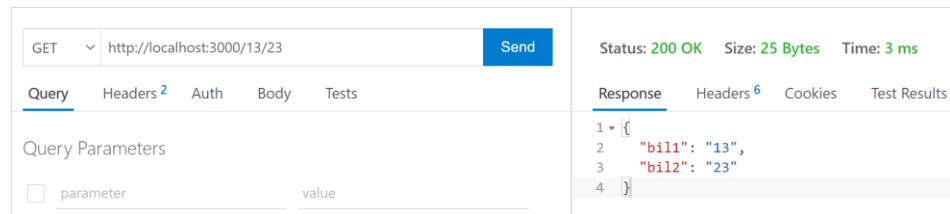
Sedangkan kata `bilangan1` dan `bilangan2` hanya berfungsi sebagai keterangan saja yang menunjukkan agar terlihat nilai apa setelah keterangan tersebut. Untuk mengakses nilai parameter digunakan mengakses property `param` dari obyek `req`, sehingga dapat ditulis sebagai berikut `req.param` seperti yang terlihat pada baris ke-6.

Selanjutnya gunakan alamat tersebut untuk mengakses endpoint dengan menggunakan REST API Client seperti yang terlihat pada Gambar 25.



Gambar 25. Endpoint param pertama.

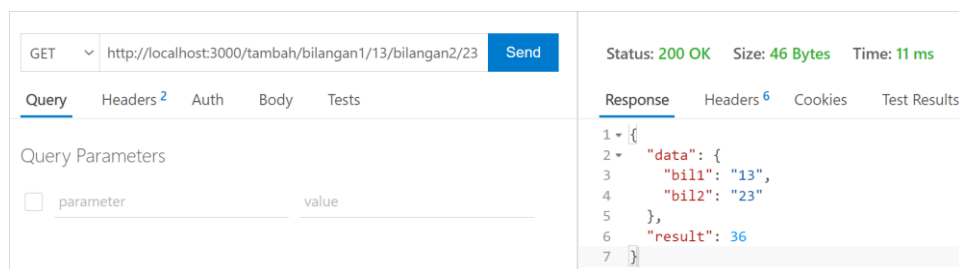
Pada contoh endpoint di baris ke-9 dapat dilihat contoh yang sama seperti sebelumnya namun hanya menggunakan nama parameter tanpa menggunakan tambahan keterangan parameter. Sehingga cara untuk mengakses endpoint ini dapat dilihat pada Gambar 26.



Gambar 26. Endpoint param kedua.

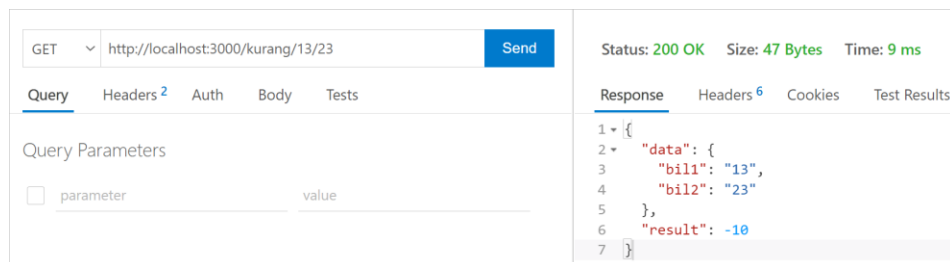
Selanjutnya contoh endpoint ketiga dimulai dari baris ke-13 dan endpoint keempat dimulai dari baris ke-18. Pada endpoint ketiga dan keempat dapat dilihat nilai parameter :bill1 dan :bil2 diproses kembali di dalam fungsi tersebut.

Pada endpoint ketiga path dari endpoint adalah /tambah sedangkan path endpoint keempat adalah /kurang. Sehingga untuk mengakses endpoint ketiga dengan contoh sebagai yang dapat dilihat pada Gambar 27.



Gambar 27. Endpoint param ketiga.

Dan pada Gambar 28 adalah contoh cara untuk mengakses endpoint keempat.



Gambar 28. Endpoint param keempat.

Pada endpoint ketiga dan keempat dapat dilihat cara untuk mengakses nilai parameter bill1 dengan menggunakan cara sebagai berikut req.param.bill1 dan untuk mengakses nilai parameter bil2 dengan cara menggunakan cara berikut req.param.bil2.

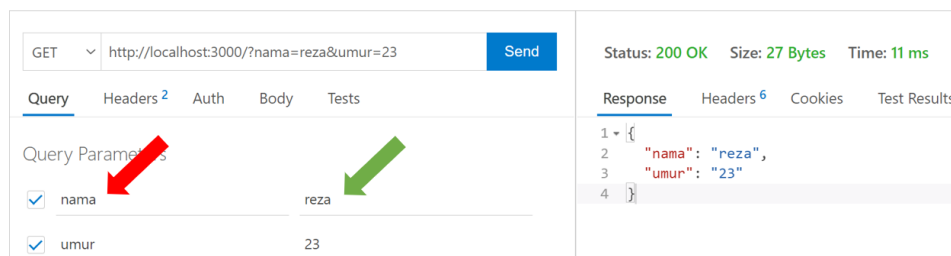
Query

Selain menggunakan parameter, pengiriman data juga bisa dengan menggunakan query string. Query string ini sudah umum digunakan sebagai salah satu cara pengiriman nilai pada aplikasi web. Dengan query string maka tidak perlu ditambahkan nama variable pada URL endpoint. Untuk melihat bagaimana hal ini bekerja maka dapat dilihat contoh kode pada Kode 8.

Kode 8. query.js

```
query.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.get('/', (req, res) => {
6    res.send(req.query)
7  });
8
9  app.get('/tambah', (req, res) => {
10    hasil = parseInt(req.query.bil1) + parseInt(req.query.bil2);
11    res.json({"data": req.query, "result": hasil})
12  });
13
14  app.listen(port, () => {
15    console.log(`cli-nodejs-api listening at http://localhost:${port}`)
16  });
```

Pada kode di atas dapat dilihat untuk mengakses nilai pada query string dengan cara menggunakan `req.query`. Sebagai contoh untuk mengakses endpoint pertama dengan REST API client dapat dilihat seperti pada Gambar 29.



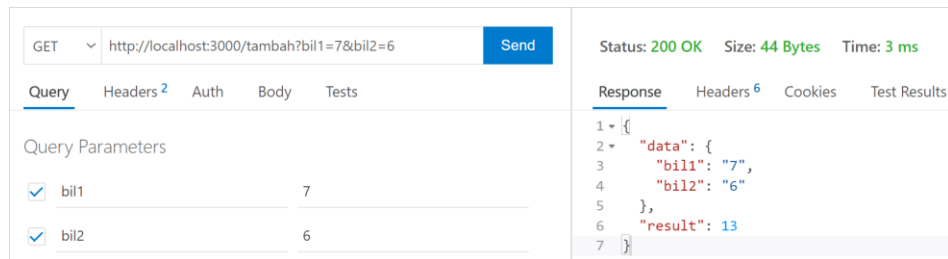
Gambar 29. Endpoint query pertama.

Jika dengan menggunakan REST API Client maka nama parameter dapat diketikkan pada input yang ditunjuk oleh panah merah. Sedangkan nilainya disamping kanan pada nama parameter seperti yang ditunjuk oleh panah hijau pada Gambar 29.

Hasil dari pengisian nama parameter dan nilainya akan berpengaruh terhadap pola URL dari endpoint tersebut. Sebagai contoh yang ditunjukkan pada Gambar 29 hasilnya dapat dilihat URL seperti di bawah ini.

```
http://localhost:3000/?nama=reza&umur=23
```

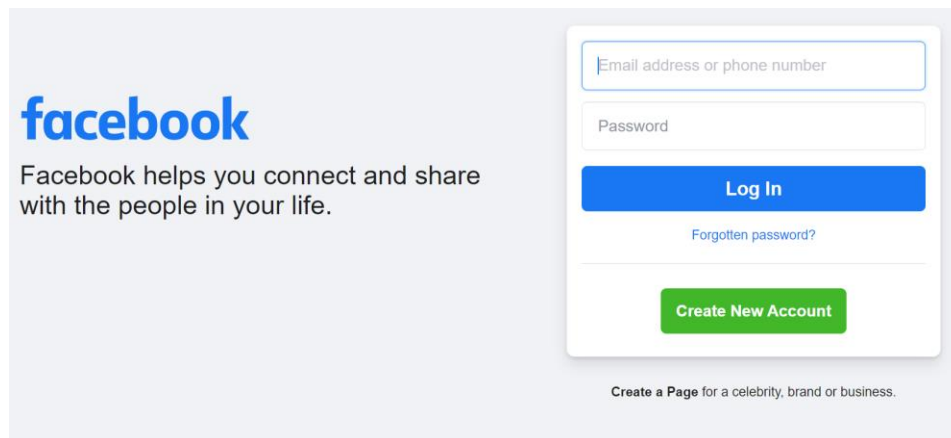
Endpoint kedua dapat dilihat dimulai dari baris ke-9. Pada contoh ini dapat dilihat pada isi fungsi ingin diakses parameter `bil1` dan `bil2` dari query string. Sehingga agar endpoint ini bisa dijalankan maka cara akses dan pemberian nilainya seperti pada Gambar 30.



Gambar 30. Endpoint query kedua.

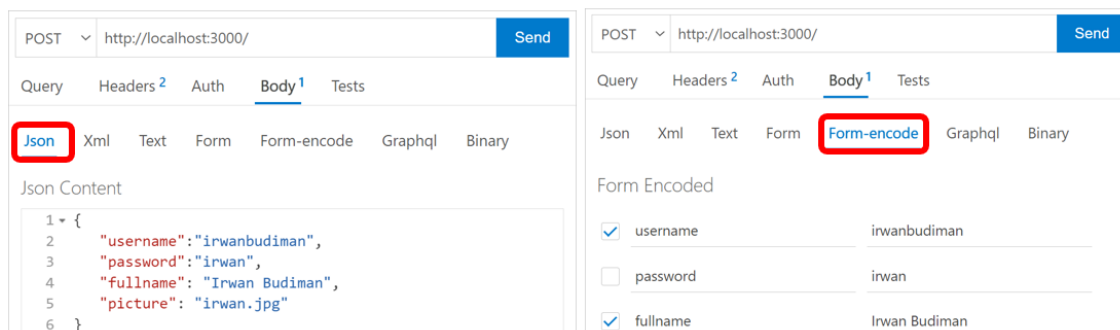
Body

Selain dalam bentuk parameter dan query, data juga dapat disisipkan di body untuk dikirimkan ke server. Contoh pesan yang dikirimkan dari body adalah form pada aplikasi web seperti form login Facebook yang terlihat pada Gambar 31. Contoh yang lain adalah form pada aplikasi mobile.



Gambar 31. Form login Facebook.

Pada sub bab ini akan diberikan contoh penanganan data dengan format JSON dan Form-encode. Seperti yang terlihat pada tab Body di Gambar 32.



Gambar 32. Body.

Sebagai informasi kedua format ini mengirimkan data plain text. Sehingga tidak dimungkinkan untuk mengirim binary seperti file atau gambar.

Kode 9 adalah kode untuk menangani data dari body. Perbedaan Kode 9 memiliki perbedaan dibandingkan kode sebelumnya. Pada kode ini terdapat tambahan baris seperti yang terlihat pada baris ke-5 dan ke-6. Tujuan tambahan baris untuk menangani data yang dikirim dari HTTP method POST dan PUT.

Kode 9. body.js.

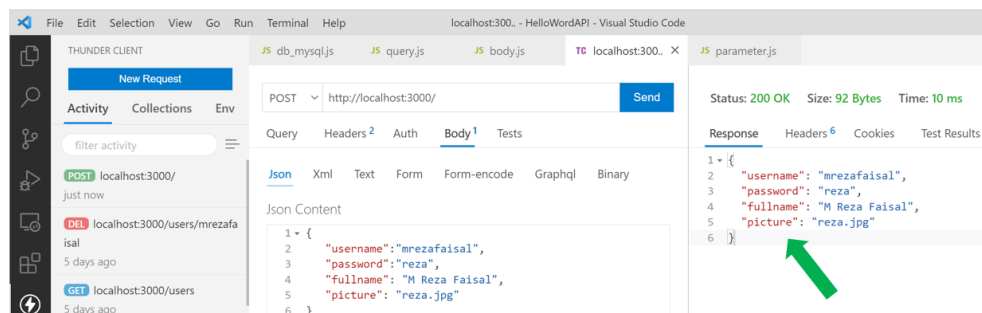
```
body.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.use(express.urlencoded({ extended: true }));
6  app.use(express.json());
7
8  app.post('/', (req, res) => {
9    res.json(req.body);
10 });
11
12 app.listen(port, () => {
13   console.log(`cli-nodejs-api listening at http://localhost:${port}`)
14 });
```

Method `express.json` yang digunakan pada obyek `app` yang digunakan pada pada baris ke-6 agar dapat mengenali request berbentuk obyek JSON. sedangkan method `express.urlencoded()` yang digunakan pada baris ke-5 berfungsi mengenali obyek request sebagai string atau array.

Untuk uji coba kode di atas lakukan langkah-langkah berikut. jalankan file di atas dengan perintah.

```
node body.js
```

Kemudian pada Visual Studio Code pilih menu Thunder Client kemudian klik tombol New Request seperti pada Gambar 33. Kemudian pilih metode POST dan ketikkan endpoint pada kolom request URL.



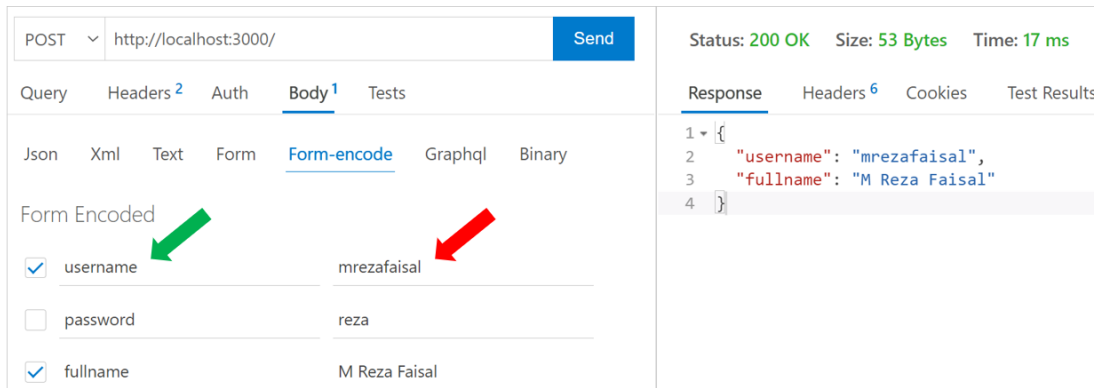
Gambar 33. Contoh data dari body format JSON.

Selanjutnya klik tab Body, dan klik tab Json. Masukkan format data JSON di bawah ini pada kolom Json Content.

```
{
  "username": "mrezafaisal",
  "password": "reza",
  "fullname": "M Reza Faisal",
  "picture": "reza.jpg"
}
```

Kemudian lakukan request dengan menekan tombol Send dan hasilnya dapat dilihat pada area yang ditunjuk oleh panah hijau pada Gambar 33.

Berikutnya dicontohkan cara penanganan data yang dimasukkan dari form input di Form-encode seperti yang dapat dilihat pada Gambar 34. Form biasanya terdiri atas nama field yang ditunjuk oleh panah hijau dan value yang ditunjuk oleh panah merah. Isikan nilai-nilai pada kolom input field dan value. Pada form ini terdapat checkbox untuk menentukan field mana yang digunakan atau tidak. Jika checkbox dicentang maka artinya field tersebut digunakan.



Gambar 34. Contoh data dari body dari form input Form-encode.

Setelah itu tekan tombol Send dan hasilnya dapat dilihat pada bagian Response seperti pada Gambar 34.

Untuk mengelola atau mendapatkan data dari body digunakan property `req.body` seperti yang dapat dilihat pada baris ke-9.

9. File Static

Contoh file static adalah gambar, file CSS dan lain-lain. Pada aplikasi web umumnya terdapat file-file tersebut. File-file tersebut biasanya dikumpulkan pada folder. Sebagai contoh gambar-gambar disimpan pada folder `images`.

Untuk mencoba penanganan file static buat folder `images` di dalam project kemudian tambahkan sebuah gambar di dalamnya sebagai contoh nama filenya adalah `img01.png`. Kemudian tambahkan file `static_file.js` pada folder dengan kode seperti pada Kode 10.

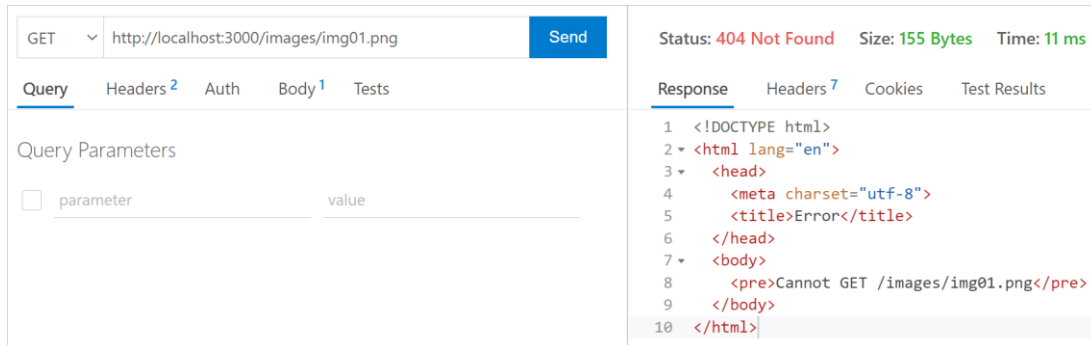
Kode 10. `static_file.js`.

```
static_file.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5
6  app.listen(port, () => {
7    console.log(`cli-nodejs-api listening at http://localhost:${port}`)
8  });
```

Kemudian jalankan kode di atas dengan mengetikkan perintah berikut.

```
node static_file.js
```

Kemudian pada kolom input request url ketikkan alamat endpoint seperti pada Gambar 35, tetapi dapat dilihat terjadi kesalahan dengan pesan seperti pada bagian response ketika tombol Send ditekan.



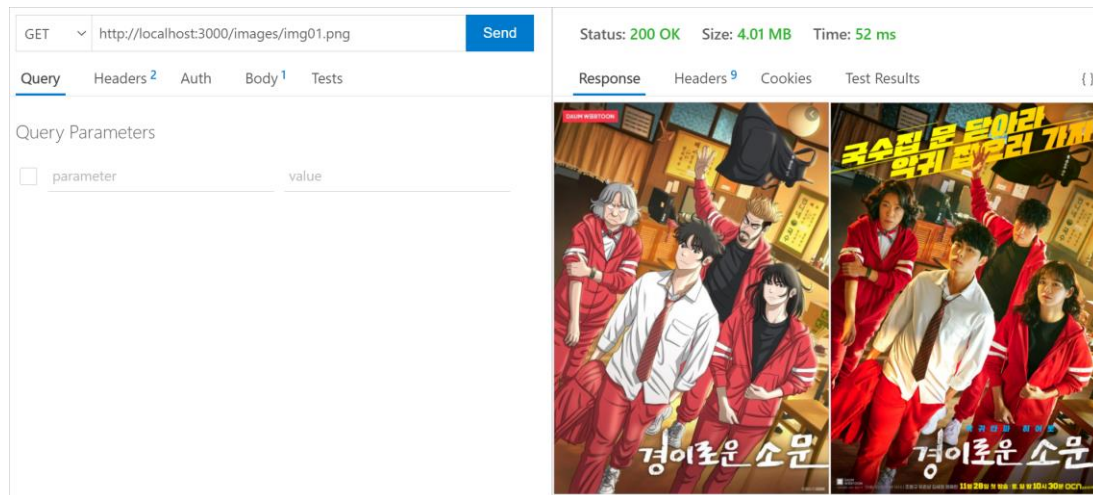
Gambar 35. Akses file static – gagal.

Kegagalan ini terjadi karena Kode 10 tidak menangani file static. Agar layanan atau service ini dapat memberikan response file static ketika ada request maka kode di atas perlu ditambahkan fungsi untuk mendaftarkan folder yang menyimpan file static seperti pada Kode 11.

Kode 11. `static_file.js` dengan penambahan penanganan file static.

```
static_file.js
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.use('/images', express.static('images'));
6
7  app.listen(port, () => {
8    console.log(`cli-nodejs-api listening at http://localhost:${port}`)
9  });
```

Untuk mendaftarkan folder yang berisi file static dapat dilihat caranya seperti pada baris ke-5 dengan menggunakan fungsi `express.static()`. Sekarang jalankan kembali file tersebut dan ulangi lakukan mengakses endpoint seperti pada Gambar 36. Dapat dilihat gambar `img01.png` yang diakses ditampilkan pada area response.



Gambar 36. Akses file static – berhasil.

10. Operasi Database

Operasi database pada umumnya terdiri atas dua (2) tahap yaitu:

- Koneksi database.
- Mengeksekusi SQL.

Sedangkan perintah SQL umumnya digunakan untuk melakukan empat (4) operasi dasar yaitu:

- Create adalah operasi untuk menambah record pada tabel. Perintah SQL untuk operasi ini menggunakan kata kunci INSERT.
- Retrieve adalah operasi untuk mengambil data pada tabel. Perintah SQL untuk operasi ini menggunakan kata kunci SELECT.
- Update adalah operasi untuk mengupdate sebuah atau lebih record yang dipilih pada suatu tabel. Perintah SQL untuk operasi ini menggunakan kata kunci UPDATE.
- Delete adalah operasi untuk menghapus sebuah atau lebih record yang dipilih pada suatu tabel. perintah SQL untuk operasi ini menggunakan kata kunci DELETE.

Instalasi

Untuk melakukan operasi database pada Node.js pada umumnya dan pada framework Express pada khususnya digunakan package atau modul `mysql`. Modul ini harus diinstall pada project. Tahapan instalasi modul ini pada project adalah sebagai berikut. Langkah pertama adalah buka command prompt kemudian pindah ke folder project yang akan ditambahkan modul ini. Setelah berada pada folder project, ketikkan perintah berikut ini atau dapat melakukan instalasi package-package pendukung seperti yang telah dicontohkan pada Kode 3.

```
npm install mysql --save
```

Koneksi

Untuk mencoba operasi database buat file dengan nama operasi_db.js dengan kode dasar sebagai berikut.

Kode 12. operasi_db.js – koneksi.

```
operasi_db.js
1  const express = require('express');
2  const app = express();
3  const mysql = require('mysql');
4  const port = 3000;
5
6  app.use(express.urlencoded({ extended: true }));
7  app.use(express.json());
8
9  // koneksi database
10 const db = mysql.createConnection({
11     host: 'localhost',
12     user: 'root',
13     password: '',
14     database: 'social_media'
15 });
16
17 app.get('/connection', (req, res) => {
18     db.connect(function(err) {
19         if(err) {
20             res.json({ "status": 200,
21                 "message": "Koneksi gagal dilakukan.",
22                 "data": err });
23         } else {
24             res.json({ "status": 200,
25                 "message": "Koneksi berhasil dilakukan." });
26         }
27     });
28 });
29
30 // tabel users -- start
31 // create
32
33 // retrieve
34
35 // update
36
37 // delete
38
39 // tabel user - end
40
41 app.listen(port, () => {
42     console.log(`Example app listening at http://localhost:${port}`)
43 });
```

Untuk melakukan operasi ke database MySQL perlu ditambahkan modul mysql pada kode seperti terlihat pada baris ke-3. Kemudian untuk melakukan koneksi ke database digunakan method atau fungsi createConnection() yang berisi informasi alamat server, user dan password untuk mengakses database

server dan nama database yang diakses seperti yang dapat dilihat pada kode dari baris ke-10 sampai dengan baris ke-15.

Sedangkan dari baris ke-17 sampai dengan ke-28 dapat dilihat contoh untuk melakukan koneksi dari di endpoint `/connection`. Pada endpoint ini digunakan obyek db dengan method `connect()` untuk melakukan koneksi ke database. Kemudian untuk memberikan respon yang berisi status koneksi digunakan method `res.json()` seperti yang terlihat pada baris ke-20 dan ke-24.

Selanjutnya lakukan uji coba endpoint dengan menggunakan Thunder Client.

Create

Sesuai dengan konsep HTTP method yang telah dibahas sebelumnya, maka untuk menambahkan data digunakan endpoint dengan HTTP method POST dengan menggunakan method `app.post()` seperti yang terlihat pada baris ke-3. Pada Kode 13 adalah sebagian kode yang ditambahkan pada file `operasi_db.js`.

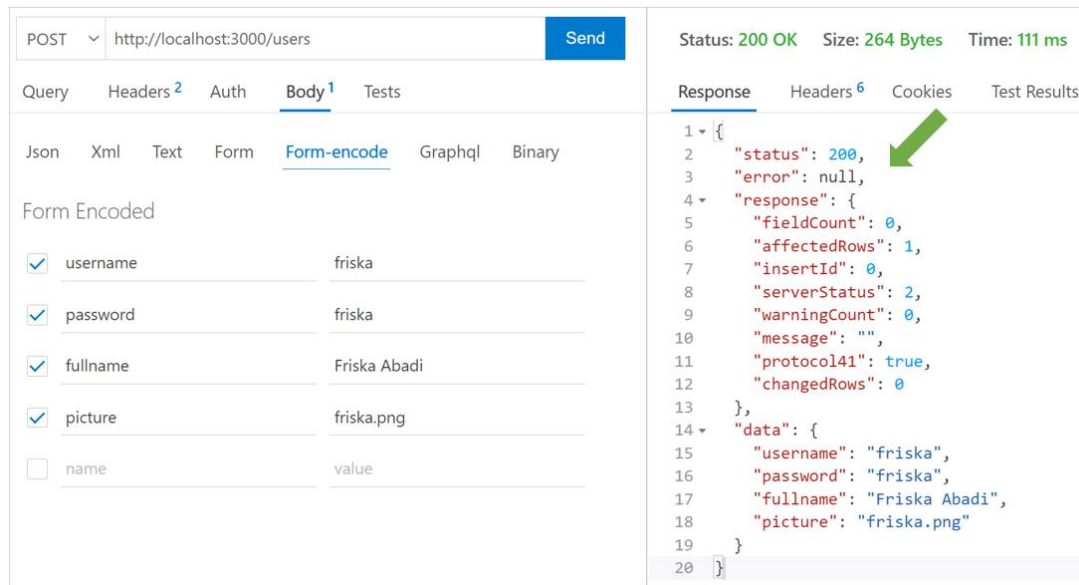
Kode 13. `operasi_db.js` – create.

```
operasi_db.js
1  ...
2  // create
3  app.post('/users', (req, res) => {
4      let sql = "INSERT INTO users SET username='"+req.body.username
5              +"', password=password('"+req.body.password
6              +"',), fullname='"+req.body.fullname
7              +"', picture='"+req.body.picture+"'";
8      db.query(sql, (err, results) => {
9          if(err) throw err;
10         res.json({ "status": 200,
11                  "error": null,
12                  "response": results,
13                  "data": req.body });
14     });
15 });
16 ...
```

Pada kode di atas dapat dilihat terdapat kode SQL dengan kata kunci INSERT. Kemudian pada kode SQL ini disisipkan nilai-nilai dari data yang dikirimkan dari body seperti yang dapat dilihat pada baris ke-4. Dan pada baris ke-8 dapat dilihat kode SQL dieksekusi dengan menggunakan method `db.query()`.

Untuk uji coba gunakan Thunder Client, kemudian pilih HTTP method POST dan ketikkan alamat endpointnya yaitu <http://localhost:3000/users>. Kemudian pilih Tab Body dan dilanjutkan pilih tab Form-encode. Kemudian isi nama field dan nilai-nilainya seperti yang terlihat pada Gambar 37.

Setelah semua nama field dan value atau nilainya diisi, klik tombol Send. Dan hasilnya dapat dilihat pada kolom respons yang ditunjuk oleh panah hijau di Gambar 37.

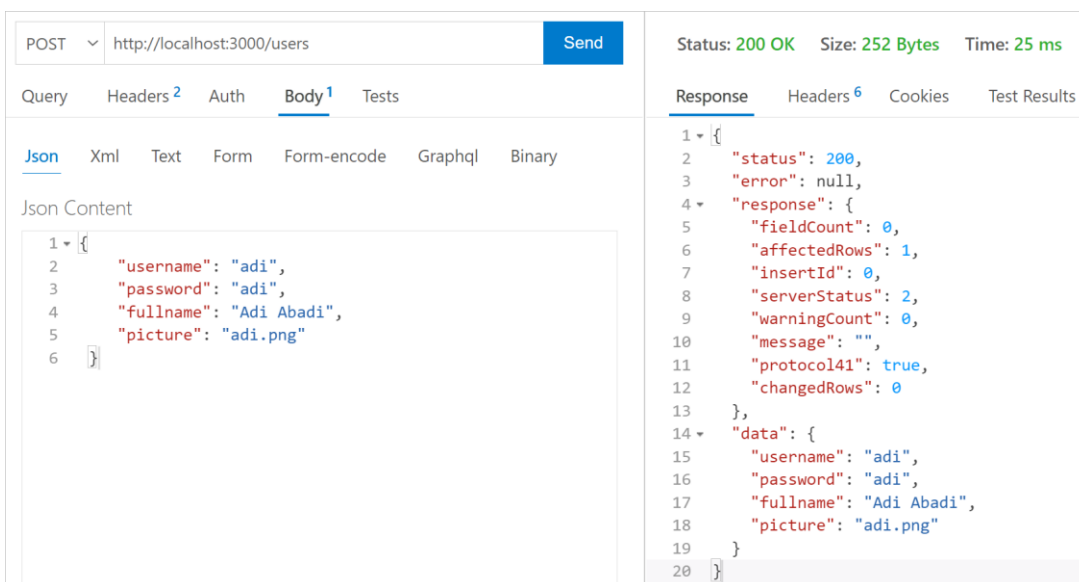


Gambar 37. input data users dari form-encode.

Jika menggunakan data format JSON sebagai input maka pilih tab Body dan kemudian pilih tab Json kemudian masukkan data berikut ini.

```
{
  "username": "adi",
  "password": "adi",
  "fullname": "Adi Abadi",
  "picture": "adi.png"
}
```

Kemudian klik tombol Send sehingga dapat dilihat response seperti pada Gambar 38.



Gambar 38. Input data user dengan data format JSON.

Retrieve

Sub bab ini memberikan contoh untuk mengambil data dari tabel users. Pada Kode 14 dapat dilihat potongan kode untuk melakukan pengambilan data users.

Kode 14. operasi_db.js – retrieve.

	operasi_db.js
1	...
2	// retrieve
3	app.get('/users',(req, res) => {
4	let sql = "SELECT * FROM users";
5	db.query(sql, (err, results) => {
6	if(err) throw err;
7	res.json({"status": 200,
8	"message": "Data berhasil diambil.",
9	"data": results});
10	});
11	});
12	...

Perbedaan Kode 14 dibandingkan dengan Kode 13 adalah dibagian kode SQL yang dimasukkan ke variable sql seperti yang terlihat pada baris ke-4. Untuk melakukan pengambilan data digunakan kode SQL dengan kata kunci adalah SELECT.

Lakukan pengujian dengan Thunder Client. Pilih HTTP method GET. Kemudian alamat endpoint adalah /users. Kemudian klik tombol Send maka hasilnya dapat dilihat pada bagian Response seperti yang terlihat pada Gambar 39.

GET

http://localhost:3000/users

Send

Query

Headers²

Auth

Body¹

Tests

Query Parameters

☐

parameter

value

Status: 200 OK

Size: 694 Bytes

Time: 66 ms

Response

Headers⁶

Cookies

Test Results

1

{

2

"status": 200,

3

"message": "Data berhasil diambil.",

4

"data": [

5

{

6

"username": "adi",

7

"password": "

8

*62F4A939227A8DD2176E3B797686948FFACB0506"

9

"fullname": "Adi Abadi",

10

"picture": "adi.png"

11

},

12

{

13

"username": "dodontn",

14

"password": "

15

*F8A59D0538CEF415368B8A9BDFB50A9EA6EBDBEC"

16

"fullname": "Dodon T Nugrahadi",

17

"picture": "dodon.png"

18

},

19

{

20

"username": "friska",

21

"password": "

22

*C99C786FD1969B44CDC318E30E1603393AA6A052"

23

"fullname": "Friska Abadi",

24

"picture": "friska.png"

25

},

Gambar 39. Pengambilan data dari tabel users.

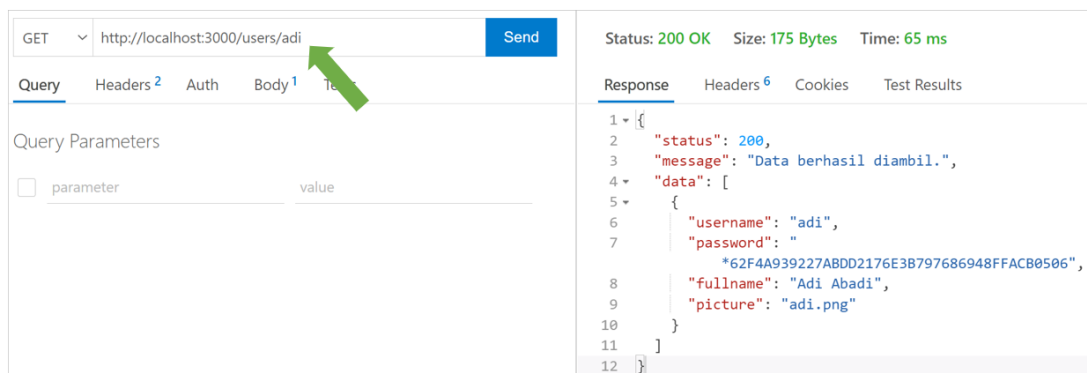
Pada contoh di atas dapat dilihat data yang ditampilkan adalah seluruh record yang ada pada tabel users. Pada sebuah aplikasi umumnya juga dibutuhkan pengambilan record berdasarkan atribut dengan nilai

tertentu. Sebagai contoh adalah mengambil data user berdasarkan username tertentu. Untuk melakukan hal tersebut dapat dilakukan potongan kode sebagai berikut.

Kode 15. operasi_db.js – retrieve berdasarkan username.

```
operasi_db.js
1  ...
2  app.get('/users/:id',(req, res) => {
3    let sql = "SELECT * FROM users WHERE username='"+req.params.id+"'";
4    db.query(sql, (err, results) => {
5      if(err) throw err;
6      res.json({"status": 200,
7               "message": "Data berhasil diambil.",
8               "data": results});
9    });
10 });
11 ...
```

Pada endpoint ini terdapat tambahan parameter untuk mengirimkan data. Selanjutnya adalah mencoba endpoint dengan mengaksesnya dengan menggunakan Thunder Client. Pada Gambar 40 dapat dilihat cara untuk memberikan nilai parameter pada endpoint. Nilai yang dimasukkan sebagai parameter adalah adi seperti yang ditunjuk oleh panah hijau pada Gambar 40.



Gambar 40. Pengambilan data user berdasarkan username.

Tekan tombol Send dan hasilnya dapat dilihat pada bagian Response.

Update

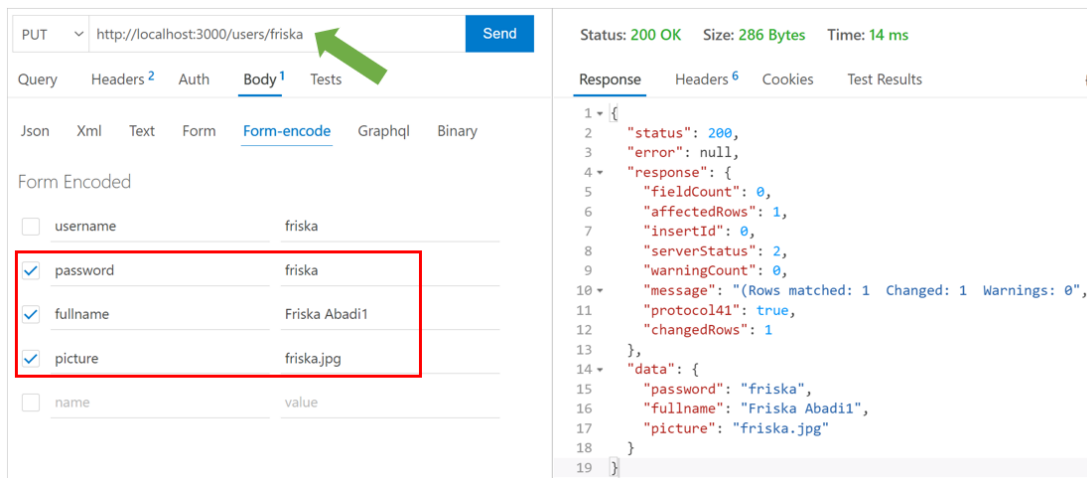
Untuk update digunakan HTTP method PUT dengan potongan kode sebagai berikut.

Kode 16. operasi_db.js – update.

```
operasi_db.js
1  ...
2  // update
3  app.put('/users/:id',(req, res) => {
4    let sql = "UPDATE users SET password=password('"+req.body.password
5              +"',), fullname='"+req.body.fullname
6              +"', picture='"+req.body.picture+"'"
7              + " WHERE username='"+req.params.id+"'";
```

```
8 db.query(sql, (err, results) => {  
9   if(err) throw err;  
10  res.json({"status": 200,  
11           "error": null,  
12           "response": results,  
13           "data": req.body});  
14 });  
15 });  
16 ...
```

Kode di atas dapat dilihat pengiriman data dilakukan dari parameter dan body. Untuk melihat bagaimana cara pengiriman data ke endpoint tersebut seperti yang terlihat pada Gambar 41.



Gambar 41. Update data.

Untuk menentukan username mana yang akan diupdate maka nilai username diberikan pada parameter di URL endpoint seperti yang ditunjuk oleh panah hijau. Kemudian nilai-nilai yang ingin diupdate dapat dilihat pada kotak warna merah di Gambar 41.

Delete

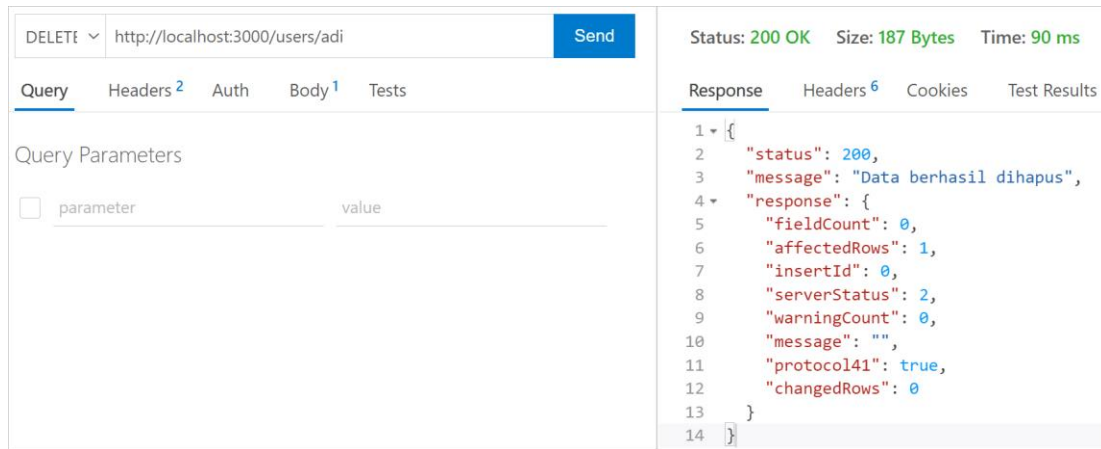
Operasi database yang terakhir adalah menghapus record. Berikut adalah contoh kode untuk menghapus sebuah record. Berikut adalah potongan kode untuk menghapus sebuah record.

Kode 17. operasi_db.js – delete.

```
operasi_db.js  
1 ...  
2 // delete  
3 app.delete('/users/:id', (req, res) => {  
4   let sql = "DELETE FROM users WHERE username='"+req.params.id+"'";  
5   db.query(sql, (err, results) => {  
6     if(err) throw err;  
7     res.json({"status": 200,  
8             "message": "Data berhasil dihapus",  
9             "response": results});  
10  });
```

11	});
12	...

Selanjutnya lakukan pengujian endpoint tersebut dengan menggunakan Thunder Client dengan hasil seperti pada.



Gambar 42. Menghapus data.

Tugas

Lakukan tugas-tugas berikut:

1. Ikuti seluruh instruksi di atas.
2. Buat tabel `books` dengan atribut adalah `ISBN` sebagai `primary key`, `title`, `author`, `page`, `picture`. Kemudian buat REST API untuk proses CRUD ke tabel `books`.

Referensi

1. <https://expressjs.com/en/starter/installing.html>

