

NODE.JS MONGODB

DIGITAL BOOK

Abstract

Buku ini berisi tentang tutorial pemrogramman web dengan Node.js dan MongoDB

Ipung Purwono
Ipungofficial@gmail.com

KATA PENGANTAR

Buku ini diperuntukan bagi pemula yang ingin migrasi dari bahasa PHP atau mencoba mengenal Node.js sebagai bahasa server. Pada awalnya buku ini ingin kami komersialkan dan masuk ke penerbit, namun dalam perjalannya yang panjang akhirnya kami memutuskan untuk dibagikan secara gratis.

Pada bahasan buku sebetulnya merupakan beberapa gabungan dari tutorial yang kami dapatkan di media internet, namun kemudian oleh kami terjemahkan dalam bahasa indonesia agar lebih mudah dipahami.

Kita tahu bahwa Node.js dan MongoDB semakin popular akhir-akhir ini dan memaksa naluri semangat belajar kami untuk mempelajari keduanya. Banyak tutorial yang tersebar di Internet dan kami pikir akan sangat menarik jika dikemas dalam bahasa Indonesia. Node.js dan MongoDB menjadi trend baru buat para pecinta dunia pemrograman khususnya website.

Kami berharap buku ini tidak hanya berhenti dianda, namun bisa dibagikan ke teman-teman yang sedang belajar pemrograman website atau teman-teman dari sekolah ataupun kampus. Buku ini bersifat gratis dan boleh disebarluaskan secara terbuka, namun kami berharap jangan dikomersialkan, seandainya ingin dikomersialkan pun wajib ada

inovasi dan penjelasan lebih mendalam lagi dengan studi kasus yang lebih kompleks dalam bentuk project website.

Ucapan terimakasih kami sampaikan kepada Allah SWT yang telah memberikan kami kesempatan dalam menulis buku Node.js dan MongoDB ini. Tak lupa kami ucapkan terimakasih untuk semua pihak yang telah membantu menyelesaikan buku ini.

Akhirnya kami berharap buku ini bisa menjadi referensi berguna bagi mereka yang mau belajar dan membaca. Dengan ini pula, kami menyatakan bahwa semua kesalahan dalam buku ini adalah milik kami.

Menulis dengan cinta,

Ipung Purwono

DAFTAR ISI

BAB I	1
A. Berkenalan dengan Node.js	1
B. Installasi Node.js	2
C. Cek Hasil Instalasi.....	2
D. Code Editor Visual Studio Code.....	3
E. HelloWorld pada Node.js.....	4
F. Membuat dan Mengenal Modul pada Node.js	8
G. Node.Js sebagai Web Server	12
H. Node.Js dalam File System Module	13
Read Files	14
Create Files & Update Files	15
Delete Files & Rename Files.....	17
Upload Files.....	19
I. Modul URL dalam Node.js	22
J. Node Package Manager (NPM)	24
K. Email	25
BAB II	28
A. Berkenalan dengan MongoDB	28
B. Installasi MongoDb	28
C. MongoDB Compass (GUI)	41
D. Membuat Database dan Collection MongoDB.....	42
E. MongoDB Insert.....	47
F. MongoDB Find	49
G. MongoDB Query	53
H. MongoDB Sort	57
I. MongoDB Delete.....	57
J. MongoDB Drop Collection	59
K. MongoDB Update	59
L. MongoDB Limit	61

M. MongoDB Join.....	62
BAB III	65
A. Instalasi Express.....	65
B. View Engine.....	69
C. Routing Express	69
D. Static Folder.....	70
E. Membuat Project CRUD Warga.....	71
1. Koneksi Express dan MongoDB.....	71
2. Membuat Models	72
3. Membuat Controllers	74
4. Membuat Routes	77
5. Panggil routes pada app.js	79
6. Membuat View Input Data.....	80
7. Melakukan Edit & Hapus Data	89
BAB IV TENTANG PENULIS.....	106

BAB I

Node.Js

A. Berkenalan dengan Node.js

Perkembangan dunia bahasa pemrograman melaju sangat cepat, saat ini kita mengetahui banyak bahasa pemrograman yang mendukung untuk pemrograman website. Dimulai yang paling populer di Indonesia yaitu PHP yang identik digandengkan dengan database MySQL. Selain itu kita mengenal bahasa pemrograman lain seperti ASP.NET, Ruby, Python, JsP dan sebagainya. Bahasa-bahasa tersebut menjadi andalan banyak programmer ketika menciptakan sebuah website yang professional.

Pada sekitar tahun 2009, Ryan Dahl telah menciptakan Node.js yang disponsori oleh perusahaan tempat ia bekerja yaitu Joyent. Node Js adalah sebuah platform perangkat lunak pada sisi server dan aplikasi jaringan dimana dapat berjalan secara cross platform di sistem operasi Windows, Linux dan Mac OsX. Node.Js ditulis dengan bahasa pemrograman javascript yang memiliki pustaka HTTP sendiri yang menjadikannya dapat berjalan disisi server tanpa bantuan dari program web server seperti Apache atau Lighttpd. Hal ini yang membedakannya dengan PHP yang sudah familiar digunakan oleh orang Indonesia yaitu selalu menginstal paket Xampp yang didalamnya terdapat Apache Server dan MySQL.

Kelebihan-kelebihan Node.js antara lain:

- Dibangun dengan bahasa javascript sehingga bagi yang sudah terbiasa membangun frontend web dengan javascript, akan mudah menguasai dari sisi backend.
- Adanya pertukaran data antar code javascript pada frontend dan backend yaitu server-side rendering.
- Bersifat open source sehingga kita bebas untuk mengembangkannya.
- Mampu membuat aplikasi yang bersifat realtime (waktu nyata)

- Mendukung penyimpanan sementara (cache)

B. Installasi Node.js

Agar Node.js dapat berjalan pada komputer anda, kita wajib untuk menginstalnya. Untuk mendapatkan Node.js anda dapat mendownloadnya pada situs resminya yaitu <https://nodejs.org/en/download/>. Disana tersedia beberapa alternatif pilihan downloader Node.js yang disesuaikan dengan sistem operasi anda.

Untuk sistem operasi windows, jangan lupa untuk memilih jenis bit komputer anda yaitu 32-bit atau 64-bit. Setelah itu anda akan mendapatkan file dalam bentuk .msi, untuk menginstalnya anda dapat mengklik dua kali file .msi yang sudah anda download sebelumnya, dan tunggu hingga proses installasi selesai. Langkah instalasinya sebetulnya sama ketika anda menginstal program windows lainnya. Sebetulnya pengguna windows sekarang sudah dimanjakan dengan adanya Visual Studio yang di dalamnya sudah terdapat Express Js yaitu salah satu framework Node.js terpopuler. Anda bisa juga memanfaatkan Visual Studio sebagai alternatif dalam membangun web dengan Node.js.

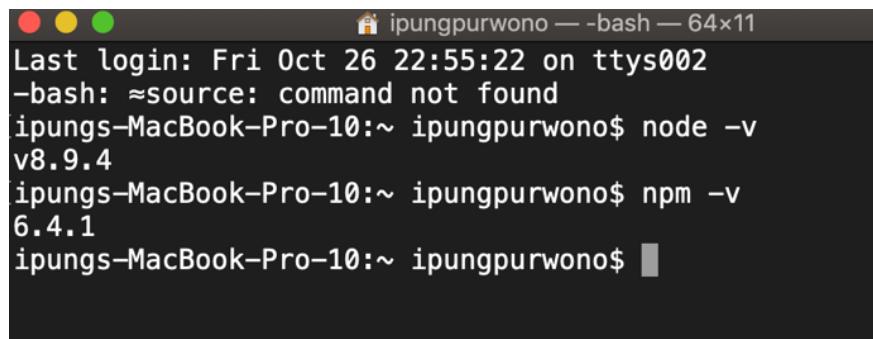
Untuk pengguna Mac anda bisa mendownload installer dalam bentuk .pkg. Karena Mac defaultnya 64-bit maka anda otomatis mendapatkan versi 64-bit. Proses instalasinya cukup sederhana anda tinggal klik dua kali .pkg file installer Node.js dan ikuti proses nya hingga selesai seperti anda menginstal software Mac lainnya.

Untuk pengguna Linux dengan basis Debian atau Ubuntu akan lebih mudah menggunakan package manager anda dapat melihat cara menginstalnya pada link berikut ini <https://nodejs.org/en/download/package-manager/>.

C. Cek Hasil Instalasi

Agar kita dapat melihat hasil instalasi Node.js sudah benar atau belum kita dapat melihatnya dengan mengetikan perintah “**node -v**” pada console / command

prompt / terminal. Serta ketikan perintah “**npm -v**” pada console / command prompt / terminal.

A screenshot of a macOS terminal window titled "ipungpurwono — bash — 64x11". The window shows the following text:

```
Last login: Fri Oct 26 22:55:22 on ttys002
-bash: ≈source: command not found
ipungs-MacBook-Pro-10:~ ipungpurwono$ node -v
v8.9.4
ipungs-MacBook-Pro-10:~ ipungpurwono$ npm -v
6.4.1
ipungs-MacBook-Pro-10:~ ipungpurwono$ █
```

The terminal has three colored tabs at the top: red, yellow, and green.

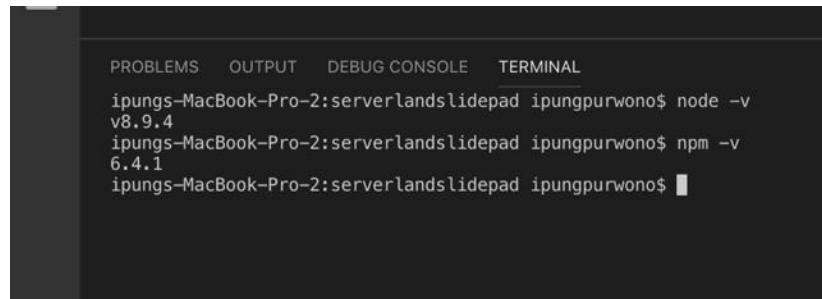
Gambar 1. Cek Hasil Instalasi Node.js via Terminal

Jika anda sudah berhasil menginstalnya maka akan terlihat versi dari Node.js dan NPM (Node Package Manager) tersebut, saat penulis menulis buku ini, versi yang digunakan adalah v8.9.4. dan NPM versi 6.4.1 Anda mungkin menemukan hasil berbeda karena Node.js terus dikembangkan versinya. NPM tersebut adalah sebuah manajemen library yang dapat dipasang atau dibuang pada aplikasi Node.js kita, kalau dalam CodeIgniter itu ya libraries, atau dalam Xamarin itu Nuget Package.

Nah sekarang kita sudah memiliki Node.js pada komputer kita ya, selanjutnya kita akan belajar tentang dasar-dasar dari Node.js tujuannya agar anda tidak kebingungan saat membuat project.

D. Code Editor Visual Studio Code

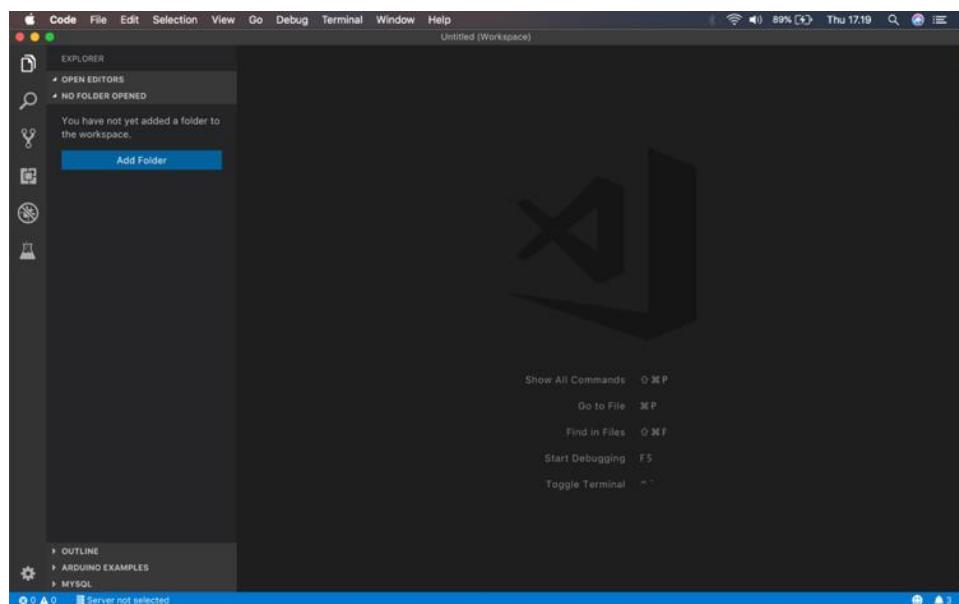
Untuk memudahkan kita dalam memprogram aplikasi dengan Node.js saran kami adalah menggunakan Visual Studio Code. Software tersebut adalah editor populer yang dibuat oleh Microsoft. Anda dapat mendownloadnya di situs resminya yaitu <https://code.visualstudio.com>. Anda dapat memilih jenis sistem operasi yang anda gunakan. Alasan kami menggunakan Visual Studio Code adalah add ons nya yang sangat support dengan javascript ditambah bisa mengoperasikan terminal pada visual studio code secara langsung. Terminal ini akan digunakan untuk installasi library yang dibutuhkan dan menjalankan server Node.js.



Gambar 2. Visual Studio Code Terminal

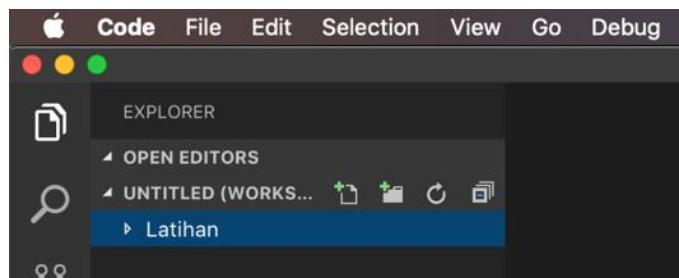
E. HelloWorld pada Node.js

Seperti pada umumnya ketika kita mencoba bahasa pemrograman baru, kita biasanya melakukan uji coba membuat “Hello World!” ya? Nah, agar kita bisa mulai mencoba Node.js kita buat sebuah file untuk menghasilkan data “Hello World!” di Browser. Nah untuk membuatnya anda wajib membuka Code Editor andalan yaitu Visual Studio Code.



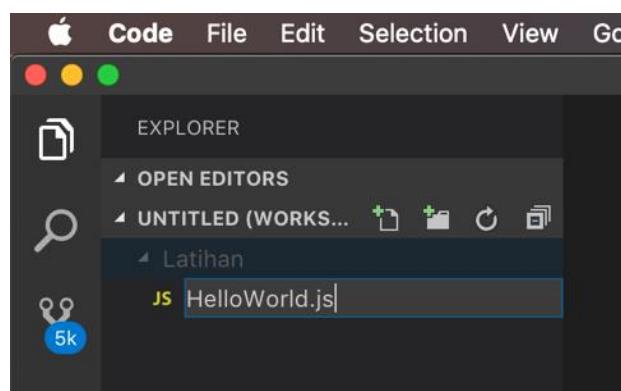
Gambar 3. Antarmuka Visual Studio Code

Agar project dapat tertata dengan rapi alangkah baiknya kita buat sebuah folder untuk menyimpan code Node.js yang akan dibuat. Pilihlah menu Add Folder, lalu arahkan ke tempat folder dimana kita akan menyimpan file code Node.js. Pada kesempatan kali ini penulis menyimpannya pada folder “Latihan” di Documents, jika anda pengguna windows bisa menyimpannya di hardisk D dengan nama folder “Latihan” atau sejenisnya terserah anda. Asumsi kami anda menggunakan nama folder “Latihan” untuk menaruh file code Node.js. Jika anda belum memiliki folder “Latihan” anda bisa membuatnya terlebih dahulu lalu tekan OK.



Gambar 4. Menambah Folder Latihan pada Visual Studio Code

Sekarang folder “Latihan” sudah masuk pada Visual Studio Code, kita akan mencoba sebuah file baru untuk menampilkan “Hello World!” pada browser. Klik kanan pada Latihan, kemudian pilih New File, berikan nama HelloWorld.js.



Gambar 5. Membuat Code HelloWorld.js

Pada file code **HelloWorld.js** ketikan code sebagai berikut:

```

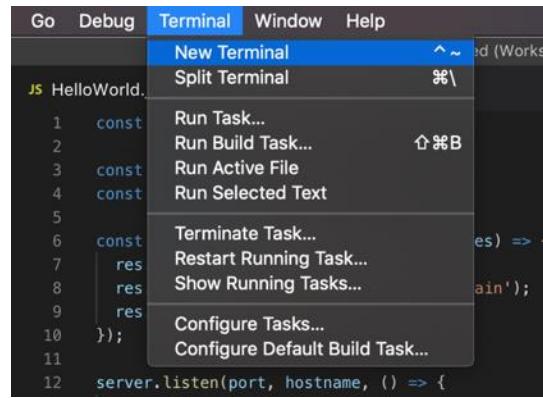
1. const http = require('http');
2.
3. const hostname = '127.0.0.1';
4. const port = 3000;
5.
6. const server = http.createServer((req, res) => {
7.   res.statusCode = 200;
8.   res.setHeader('Content-Type', 'text/plain');
9.   res.end('Hello World\n');
10. });
11.
12. server.listen(port, hostname, () => {
13.   console.log(`Server running at http://${hostname}:${port}/`);
14. });

```

Penjelasan code program:

- Pada baris 1 kita membuat sebuah variable dengan tipe const, const adalah tipe variable yang bersifat consistent alias tetap jadi tidak boleh diubah. Disana kita buat variable http dengan memanggil library http.
- Pada baris 3 kita juga membuat variable hostname kalau kalian yang sudah sering pake php ya localhost yang ber ip address 127.0.0.1.
- Pada baris 4 membuat variable port dimana aplikasi Node.js berjalan, kita buat port 3000
- Pada baris 6 kita buat variable dengan nama server yang di dalamnya library http akan membuat sebuah server dan mengeset header dengan tipe text/plain dan menampilkan text dengan tulisan “Hello World”.
- Pada baris 12 server yang sudah kita ciptakan akan melakukan listen yaitu akan dijalankan pada port dan hostname yang sudah didefinisikan sebelumnya.
- Baris 13 akan menghasilkan console berupa dimana server akan berjalan.

Sekarang kita gunakan Terminal pada Visual Studio Code, Pilih Terminal -> New Terminal



Gambar 6. Mengaktifkan Terminal Baru

Kita memilih New Terminal dengan tujuan agar kita masuk pada terminal baru dengan lokasi folder menyesuaikan project kita, karena kita menggunakan folder latihan, maka terminal langsung diarahkan ke folder terminal.

A screenshot of the VS Code terminal window. The code editor above shows the 'HelloWorld.js' file with the following content:

```

10  });
11
12 server.listen(port, hostname, () => {
13   console.log(`Server running at http://${hostname}:${port}/`);
14 });

```

The terminal window below shows the command being run and its output:

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
bash: ≈source: command not found
ipungs-MacBook-Pro-10:Latihan ipungpurwono$ █

```

Gambar 7. Terminal pada folder Latihan

Dapat dilihat, saat ini posisi terminal pada folder latihan, jadi kita bisa mengetikan beberapa perintah untuk menjalankan code Node.js. Sekarang kita akan jalankan file HelloWorld.js tersebut dengan mengetikan perintah “**node HelloWorld**” pada terminal. Maka anda bisa melihat pada console terminal “**Server running at http://127.0.0.1:3000/**”.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

bash: ~source: command not found
ipungs-MacBook-Pro-10:Latihan ipungpurwono$ node HelloWorld
Server running at http://127.0.0.1:3000/

```

Gambar 8. Menjalankan Node.js sebagai Server

Dapat kita lihat pada console terminal, bahwa server sedang berjalan pada port 3000. Untuk menyakinkan kita apakah benar sudah berjalan kita bisa membuka browser lalu ketikan url <http://127.0.0.1:3000>

**Gambar 9.** Server berjalan pada port 3000

Dan, hola selamat anda sekarang sudah menjadi programmer Node.js. Untuk menghentikan server / stop server dapat menggunakan Ctrl + C. Selanjutnya kita akan mengeksplor keunggulan lain dari Node.js yang sangat bermanfaat untuk membuat aplikasi website sebagai alternatif PHP atau bahasa server lainnya.

F. Membuat dan Mengenal Modul pada Node.js

Node.js sendiri sebetulnya sudah memiliki modul bawaan yang dapat kita lihat di tabel berikut:

Tabel 1. Modul Built-in Node.js

Module	Description
assert	Provides a set of assertion tests
buffer	To handle binary data
child_process	To run a child process
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions

dgram	Provides implementation of UDP datagram sockets
dns	To do DNS lookups and name resolution functions
domain	Deprecated. To handle unhandled errors
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.
net	To create servers and clients
os	Provides information about the operation system
path	To handle file paths
punycode	Deprecated. A character encoding scheme
querystring	To handle URL query strings
readline	To handle readable streams one line at the time
stream	To handle streaming data
string_decoder	To decode buffer objects into strings
timers	To execute a function after a given number of milliseconds
tls	To implement TLS and SSL protocols
tty	Provides classes used by a text terminal
url	To parse URL strings
util	To access utility functions
v8	To access information about V8 (the JavaScript engine)
vm	To compile JavaScript code in a virtual machine
zlib	To compress or decompress files

Modul built-in di atas sengaja tidak kami jelaskan lebih detail dikarenakan ada keterbatasan waktu dalam penulisan buku ini, namun kami akan fokus pada beberapa hal yang sekiranya wajib terlebih dahulu dikuasai untuk memahami pembuatan web menggunakan Node.js.

Modul adalah fungsi yang dapat kita buat sendiri menggunakan javascript dan dapat kita tempatkan pada aplikasi Node.js kita. Jika pada tabel diatas berisi

modul bawaan dari Node.js, kita bisa menciptakan modul sendiri atau bisa kita sebut dengan nama Export Module.

Untuk membuat modul sebetulnya sangat mudah, kita hanya perlu menuliskan code dalam balutan fungsi yang bisa dimanfaatkan pada kode lain, jika kalian pernah memakai framework CodeIgniter misalnya, kita memanggil sebuah model kan? Pada model tersebut didalamnya sudah terdapat beberapa fungsi untuk memanipulasi database MySQL. Nah secara umum kinerjanya sama seperti itu. Sekarang kita akan membuat modul sederhana pada Node.js.

Kita akan membuat 2 buah file javascript dimana salah satunya adalah modul berjenis export module.

- App.js
- Data.js

Nah terlebih dahulu kita buat file Data.js dengan code berikut

Data.js

```
1. module.exports = {
2.   fungsi1: 'Ipung Purwono ',
3.   fungsi2: function(a,b) {
4.     return a * b;
5.   },
6.   fungsi3: 'Belajar Node.js'
7. }
```

Penjelasan code:

- Baris 1 kita mengexports modul dari file Data.js
- Baris 2 kita membuat fungsi1 berisi data string “Ipung Purwono”
- Baris 3 membuat fungsi baru mengalikan 2 buah bilangan
- Baris 6 membuat fungsi menampilkan data string “Belajar Node.js”

App.js

```

1. var metodeku = require('./Data.js');
2. var method1 = metodeku.fungsi1;
3. var method2 = metodeku.fungsi2(4,3);
4. var method3 = metodeku.fungsi3;
5.
6. console.log(method1 + method3);
7. console.log("hasil penjumlahan =" + method2);

```

Penjelasan code:

- Baris 1 kita membuat variable baru dengan nama metodeku dimana memanggil modul dengan nama Data.js yang sudah kita buat sebelumnya.
- Baris 2 definisikan kembali variable method1 dimana isinya adalah turunan dari variable metodeku yang memanggil fungsi1.
- Baris 3 definisikan kembali variable method1 dimana isinya adalah turunan dari variable metodeku yang memanggil fungsi2.
- Baris 4 definisikan kembali variable method1 dimana isinya adalah turunan dari variable metodeku yang memanggil fungsi1.
- Baris 6 tampilkan pada layar console untuk melihat hasil dari isi variable method1 dan method2.
- Baris 7 tampilkan pada layar console untuk melihat hasil dari fungsi method2.

Sekarang, anda jalankan file Node.js yaitu App.js dengan perintah “**node App.js**” pada terminal.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
ipungs-MacBook-Pro-10:Latihan ipungpurwono$ node App.js
Ipung Purwono Belajar Node.js
hasil penjumlahan =12
ipungs-MacBook-Pro-10:Latihan ipungpurwono$ █

```

Gambar 10. Hasil export module pada Node.js

Dapat kita lihat di layar console, muncul “Ipung Purwono Belajar Node.js” dan dibawahnya muncul “hasil penjumlahan = 12”. Untuk menghentikan server / stop server dapat menggunakan Ctrl + C. Bagaimana menarik bukan Node.js ini.

G. Node.Js sebagai Web Server

Pada saat kita membuat file **HelloWorld.js** sebetulnya sudah memanfaatkan Node.js sebagai web server karena didalamnya kita sudah memanggil modul http. Modul http yang sudah termasuk modul bawaan Node.js ini memiliki beberapa fungsi antara lain adalah menambahkan HTTP Header dan membaca Query String. Nah agar lebih mendalam lagi memahami bagaimana Node.js sebagai web server bekerja akan kita gunakan beberapa file.

HelloWorld.js

```

1. const http = require('http');
2.
3. const hostname = '127.0.0.1';
4. const port = 3000;
5.
6. const server = http.createServer((req, res) => {
7.   res.statusCode = 200;
8.   res.setHeader('Content-Type', 'text/html');
9.   res.end(`<h1 style=color:red;>Hello World\n</h1>`);
10. });
11.
12. server.listen(port, hostname, () => {
13.   console.log(`Server running at
      http://${hostname}:${port}/`);
14. });

```

Code diatas adalah modifikasi dari file HelloWorld.js yang kami rubah pada baris ke 8 dan ke 9. Pada baris ke 8 header kita set sebagai text/html dan pada baris ke 9 kita tampilkan kode HTML dengan imbuhan css untuk merubah warna Heading H1 menjadi warna merah. Sekarang anda jalankan dengan mengetikan “**node HelloWorld**” Jika berhasil maka anda akan melihat pada brwoser anda dengan mengetikan url 127.0.0.1:3000 dengan tampilan text “Hello World!” berwarna merah.

Untuk membaca sebuah Query String pada url sebuah web, kita dapat menggunakan modul http juga. Sebagai contoh domain website kita adalah <http://ipungpurwono.com/halaman>. Nah string “halaman” adalah query string yang dimaksud. Cara membuatnya adalah sebagai berikut.

HelloWorld.js

```

1. const http = require('http');
2.
3. const hostname = '127.0.0.1';
4. const port = 3000;
5.
6. const server = http.createServer((req, res) => {
7.   res.statusCode = 200;
8.   res.setHeader('Content-Type', 'text/html');
9.   res.write(req.url);
10.  res.end();
11. });
12.
13. server.listen(port, hostname, () => {
14.   console.log(`Server running at
    http://${hostname}:${port}/`);
15. });

```

Kita kembali memodifikasi file **HelloWorld.js** ya, dengan modifikasi code pada baris ke 9 dan 10. Sekarang anda jalankan dengan mengetikan “**node HelloWorld**” pada terminal anda lalu pada browser anda tambahkan string di belakang url sebagai contoh <http://127.0.0.1:3000/Purwokerto> maka pada halaman browser anda akan muncul text “/Purwokerto”. Selamat mencoba.

H. Node.Js dalam File System Module

Penanganan file juga dapat dilakukan oleh Node.js dengan memanfaatkan modul file system. Modul ini dapat bekerja dengan sistem file yang ada pada komputer kita. File system (fs) dapat kita panggil modulnya dengan fungsi **require('fs')**.

Beberapa fitur yang dapat dikerjakan oleh modul file system(fs) adalah:

- Read files (Membaca file dalam komputer kita)
- Create files (Menciptakan file)
- Update files (Mengubah file)

- Delete files (Menghapus file)
- Rename files (Mengubah nama file)

Read Files

Read files merupakan fungsi dapat digunakan untuk membuka sebuah file dalam komputer kita. Misalkan kita memiliki sebuah halaman dalam bentuk file .html. Sebagai contoh kita akan membuat sebuah halaman dengan nama halaman.html dengan code sebagai berikut.

Halaman.html

```

1. <html>
2. <body>
3. <h1>Halaman Kita</h1>
4. <p>Ini adalah sebuah file yang dapat dibuka dengan Node.Js lho, menarik bukan?.</p>
5. </body>
6. </html>
```

Kemudian kita membuat sebuah file Node.js dengan nama BacaFile.js

BacaFiles.js

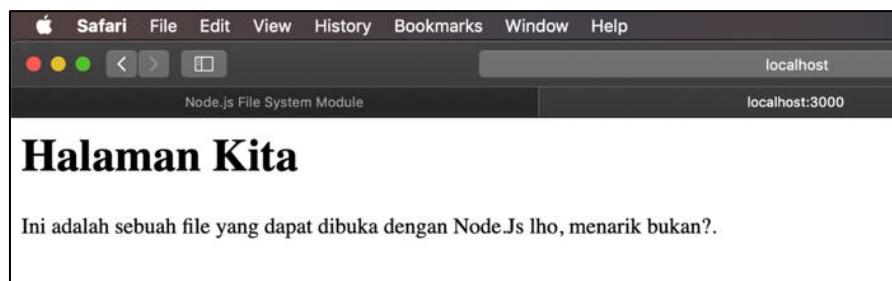
```

1. const http = require('http');
2. var fs = require('fs');
3. const hostname = '127.0.0.1';
4. const port = 3000;
5. const server = http.createServer((req, res) => {
6.   fs.readFile('Halaman.html', function(err, data) {
7.     res.statusCode = 200;
8.     res.setHeader('Content-Type', 'text/html');
9.     res.write(data);
10.    res.end();
11.  });
12. });
13.
14. server.listen(port, hostname, () => {
15.   console.log(`Server running at
16.   http://${hostname}:${port}/`);
17. });
```

Penjelasan code:

- Baris 2 kita memanggil modul file system (fs)
- Baris 6 kita gunakan modul fs untuk membaca file Halaman.html yang sudah kita buat sebelumnya.
- Baris 14 kita jalankan server node js

Jika anda sudah membuat dua buah file tersebut selanjutnya anda bisa menjalankan server dengan cara mengetikan perintah “**node BacaFiles**” pada terminal visual studio code anda. Untuk melihat hasilnya anda bisa mengakses halaman <http://127.0.0.1:3000>



Gambar 11. Membaca Files dengan Node.js

Create Files & Update Files

Modul file system (fs) juga dapat dimanfaatkan untuk membuat sebuah file menggunakan Node.js. Untuk lebih jelasnya kita dapat melihat dari beberapa contoh berikut ini.

- a. Membuat file dengan nama **filepertamasaya.txt** yang akan disimpan pada folder **files**. Nah folder files ini sebelumnya harus dibuat pada folder Latihan. Untuk membuat files kita bisa menggunakan fungsi **appendFile()**. Nah untuk lebih jelas format codenya sebagai berikut :

BuatFile.js

```
1. var fs = require('fs');
2.
3. fs.appendFile('./files/filepertamasaya.txt', 'Hello ini
   adalah file pertama saya!', function (err) {
```

```

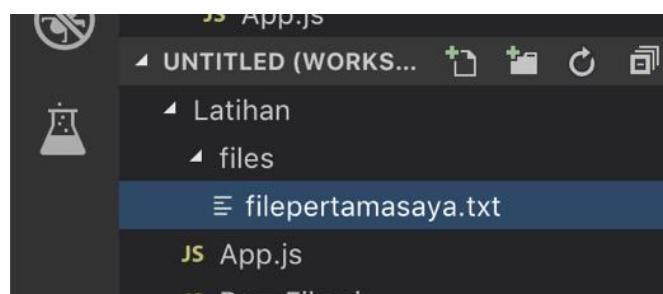
4. if (err) throw err;
5. console.log('File Tersimpan!');
6. });

```

Penjelasan code:

- Baris 1 akan dipanggil modul file system (fs).
- Baris 3 gunakan fungsi appendFile dimana di dalamnya kita menentukan lokasi folder dan nama filenya. Lokasi folder kami taruh di folder files dan nama filesnya adalah filepertamasaya.txt. Selanjutnya kita isi dengan konten text “Hello ini adalah file pertama saya!”.
- Baris 5 berikan layar console jika tidak error dengan log File Tersimpan!

Sekarang jalankan dengan perintah “node BuatFile.js” pada terminal visual studio code anda. Dan anda akan menjumpai sebuah file baru pada folder files.



Gambar 12. Membuat File dengan Node.js

- b. Membuat file dengan nama **filekeduasaya.txt** namun tanpa konten di dalamnya. Kita bisa menggunakan fungsi **open()**.

BuatFile.js

```

1. var fs = require('fs');
2.
3. fs.open('./files/filekeduasaya.txt', 'w', function (err,
   file) {
4.   if (err) throw err;

```

```

5.   console.log('Tersimpan!');
6. });

```

Penjelasan code:

- Baris 3 kita memodifikasi file BuatFile.js yang sudah sebelumnya kita buat dengan memanfaatkan fungsi **open()** dimana di dalamnya kita memanggil flag “w” yang artinya write. Jika anda menjalankannya, maka anda akan mendapatkan satu file baru dengan nama **filekeduasaya.txt** namun isinya kosong.
- c. Mereplace file dari **filepertamasaya.txt** dengan menggunakan fungsi **writeFile()**. Fungsi tersebut mampu mereplace file yang sudah ada sebelumnya untuk diganti isinya sesuai perintah. Fungsi ini dapat anda manfaatkan untuk mengubah file ya atau biasa kita sebut dengan nama **Update**.

BuatFile.js

```

1. var fs = require('fs');
2.
3. fs.writeFile('./files/filepertamasaya.txt', 'Hello isi
   content!', function (err) {
4.   if (err) throw err;
5.   console.log('Tersimpan!');
6. });

```

Penjelasan code:

- Baris ke 3 memanfaatkan **writeFile()** dengan tujuan mengubah isi dari file **filepertamasaya.txt** dengan isi “Hello isi content!”. Jika anda menjalankannya tentunya dengan perintah “node BuatFile” maka secara otomatis isi dari file **filepertama.txt** akan berubah.

Delete Files & Rename Files

Modul file system (fs) selain untuk membuat dan mengubah file, juga bisa dimanfaatkan untuk melakukan delete (hapus) file atau merename (mengubah

nama) file. Fungsi delete dapat memanfaatkan **unlink()** nah jika kalian sudah pernah coding CRUD (create, read, update, dan delete) pada PHP yang menggunakan images. Ketika proses update dan delete maka images lama akan dihapus dengan **unlink()** lalu diganti dengan images yang baru.

HapusFiles.js

```
1. var fs = require('fs');
2.
3. fs.unlink('./files/filepertamasaya.txt', function (err) {
4.   if (err) throw err;
5.   console.log('Data Terhapus!');
6. });
```

Penjelasan Code

- Baris ke 3 kita memanfaatkan fungsi **unlink()** untuk menghapus file **filepertamasaya.txt** yang terdapat pada folder **files**. Jika anda menjalankannya dengan mengetikan perintah “**node HapusFile**” maka secara otomatis file **filepertamasaya.txt** akan terhapus.

RenameFiles.js

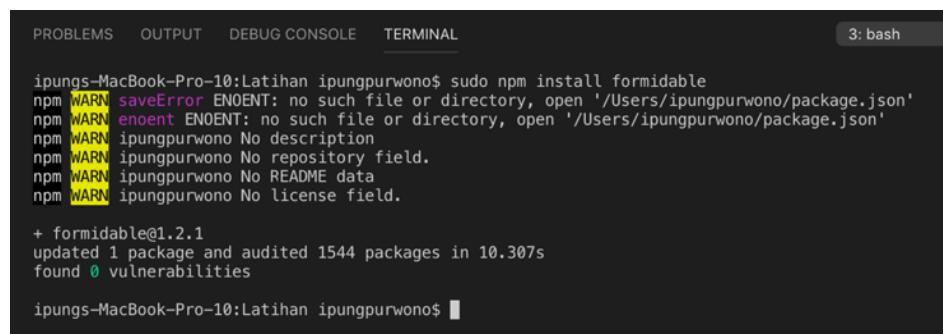
```
1. var fs = require('fs');
2.
3. fs.rename('./files/filekeduasaya.txt', './files/filekeduasaya
ubah.txt', function (err) {
4.   if (err) throw err;
5.   console.log('Data Berhasil di Rename!');
6. });
```

Penjelasan Code

- Baris ke 3 kita menggunakan fungsi **rename()** dimana akan merename file **filekeduasaya.txt** menjadi **filekeduasayaubah.txt**. Masing-masih file kami taruh pada folder **files**. Jika anda menjalankan perintah “**node RenameFiles**” pada terminal maka **filepertamasaya.txt** secara otomatis akan berubah namanya menjadi **filekeduasayaubah.txt**.

Upload Files

Node.js mendukung kemudahan untuk melakukan upload files. Agar Node.js dapat menjalankan fungsi untuk mengupload files kita harus memanfaatkan library bernama *formidable*. Modul *formidable* harus kita instal terlebih dahulu melalui NPM (*node package manager*). Penjelasan tentang NPM akan dibahas setelah bahasan upload files. Untuk menginstal *formidable* dapat mengetikan perintah “**npm install formidable**” pada terminal visual studio code anda.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
3: bash

ipungs-MacBook-Pro-10:Latihan ipungpurwono$ sudo npm install formidable
npm WARN saveError ENOENT: no such file or directory, open '/Users/ipungpurwono/package.json'
npm WARN enoent ENOENT: no such file or directory, open '/Users/ipungpurwono/package.json'
npm WARN ipungpurwono No description
npm WARN ipungpurwono No repository field.
npm WARN ipungpurwono No README data
npm WARN ipungpurwono No license field.

+ formidable@1.2.1
updated 1 package and audited 1544 packages in 10.307s
found 0 vulnerabilities

ipungs-MacBook-Pro-10:Latihan ipungpurwono$
```

Gambar 13. Instal *formidable* via npm pada terminal

Setelah kita berhasil menginstal *formidable* kita akan membuat sebuah file Node.js baru dengan nama **Upload.js** dimana dalam file tersebut kita akan membuat sebuah form upload files.

Upload.js

```
1. const http = require('http');
2. var formidable = require('formidable');
3. var fs = require('fs');
4.
5. const hostname = '127.0.0.1';
6. const port = 3000;
7.
8. const server = http.createServer((req, res) => {
9.   if (req.url == '/fileupload') {
10.     var form = new formidable.IncomingForm();
11.     form.parse(req, function (err, fields, files) {
12.       var oldpath = files.filetoupload.path;
13.       var newpath = './files/' + files.filetoupload.name;
14.       fs.rename(oldpath, newpath, function (err) {
15.         if (err) throw err;
16.         res.write('File uploaded and moved!');
17.       });
18.     });
19.   }
20. });
21.
22. server.listen(port, hostname, () => {
23.   console.log(`Server running at http://${hostname}:${port}/`);
24. });


```

```

17.           res.end();
18.       });
19.   });
20. } else {
21.res.writeHead(200, {'Content-Type': 'text/html'});
22.res.write('<form action="fileupload" method="post"
    enctype="multipart/form-data">');
23.res.write('<input type="file" name="filetoupload"><br>');
24.    res.write('<input type="submit">');
25.    res.write('</form>');
26.    return res.end();
27. }
28.});
29.
30.server.listen(port, hostname, () => {
31.console.log(`Server running at http://${hostname}:${port}/`);
32.});

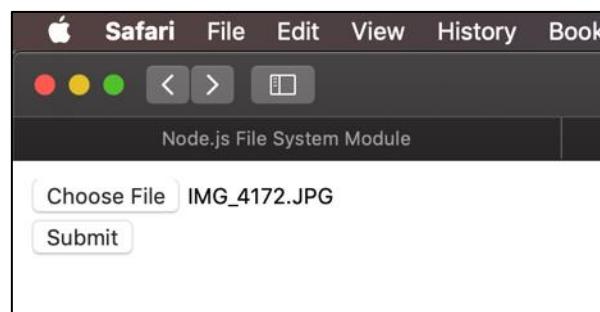
```

Penjelasan code :

- Baris 2 kita panggil modul formidable yang kita tampung dalam bentuk variable.
- Baris ke 9 adalah sebuah route atau url file yang kita gunakan sebagai action sebuah form, hal ini akan kita jalankan jika ternyata sebuah form sudah dibuat.
- Baris 10 kita panggil sebuah form untuk upload files
- Baris 11 form kita parse dalam request sebuah fungsi untuk mendefinisikan sebuah files.
- Baris 12 definisikan lokasi lama file disimpan pada kasus ini kami mengambil gambar pada salah satu lokasi di komputer kita misalkan Desktop.
- Baris 13 definisikan lokasi baru file akan disimpan.
- Baris 14 gunakan fungsi file sistem(fs) renam untuk mengubah lokasi file menjadi lokasi baru yang sudah di definisikan pada lokasi folder files yang sudah kita buat sebelumnya.
- Baris 16 jika berhasil berpindah lokasi maka munculkan log bahwa file sudah berhasil diupload dan berpindah pada lokasi baru.
- Baris 20 jika rupanya belum ada action url yang dipanggil alias pada saat server berjalan belum menjalankan submit button, maka ciptakan sebuah form.

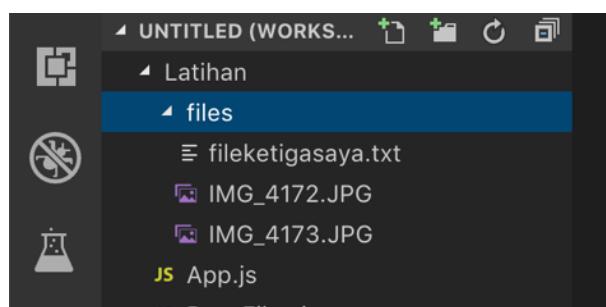
- Baris 22 buat sebuah form dengan tipe multipart/form-data yaitu form yang akan menangani upload files. Pada form ini terdapat action yang akan menjalankan kode pada baris ke 9 yaitu fileUpload.
- Baris 23 buat sebuah inputan dengan tipe files.
- Baris 24 buat sebuah submit button.

Sekarang anda jalankan perintah “node Upload” pada terminal anda, kemudian server Node.js akan berjalan. Jika anda buka browser anda lalu ketikan url <http://127.0.0.1:3000> maka anda akan menemukan sebuah form upload pada browser. Pilihlah sebuah file dan tekan tombol submit, maka anda berhasil mengupload sebuah file.



Gambar 14. Form upload files

Anda pilih sebuah files, misalkan kami memilih sebuah image, ketika kita pilih Submit maka file akan terupload pada folder files.



Gambar 15. File berhasil terupload ke folder files

I. Modul URL dalam Node.js

Jika kita lihat berbagai web setiap navigasi pasti memiliki sebuah url atau alamat. Misalkan ketika kita berbelanja pada toko online, kita menemukan url misalkan <https://namatoko.com/shop>. Hal tersebut adalah sebuah url yang dapat kita buat sendiri atau bisa kita katakan adalah routing. Jika anda pernah ngoding dengan CodeIgniter atau Laravel mungkin, kalian akan sering bertemu untuk membuat sebuah link routes. Nah konsepnya kurang lebih seperti itu. Modul URL ini sudah built-in atau termasuk dalam bawaan Node.js kita hanya perlu memanggilnya saja dengan perintah **require('url')**. Sebagai contoh kita akan membuat sebuah file dengan nama ContohUrl.js sebagai berikut.

ContohUrl.js

```

1. var url = require('url');
2. var alamat = 'http://localhost:8080/cari.htm?nama_kota=Purwok
   erto&id_user=3';
3. var q = url.parse(alamat, true);
4.
5. console.log(q.host);
6. console.log(q.pathname);
7. console.log(q.search);
8.
9. var qdata = q.query;
10. console.log(qdata.nama_kota);

```

Penjelasan Code

- Baris 1 kita panggil modul url
- Baris 2 buat sebuah variable alamat website
- Baris 3 parsingkan variable alamat dengan modul url
- Baris 5 tampilkan data pada console untuk menampilkan nama host
- Baris 6 tampilkan data pada console untuk menampilkan nama pathname
- Baris 7 tampilkan query pencarian
- Baris 9 ambil data query dari url yang sudah diparsing
- Baris 10 tampilkan data pada console untuk menampilkan nama_kota

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

ipungs-MacBook-Pro-10:Latihan ipungpurwono$ node ContohUrl
localhost:8080
/cari.htm
?nama_kota=Purwokerto&id_user=3
Purwokerto
ipungs-MacBook-Pro-10:Latihan ipungpurwono$ █

```

Gambar 16. Penanganan url pada Node.js

Modul URL bisa kita gunakan untuk membuka sebuah file lain atau bisa disebut sebagai File Server, jika kita pernah mendesain website misalkan kita memiliki home.html lalu ada hyperlink menuju about.html dan sebagainya. Nah disini kita juga bisa membuatnya. Sebagai contoh kami punya sebuah file dengan nama About.html dan juga FileServer.js

About.html

```

1. <!DOCTYPE html>
2. <html>
3. <body>
4. <h1>About Us</h1>
5. <p>Ini adalah halaman about us anda!</p>
6. </body>
7. </html>

```

FileServer.js

```

1. const http = require('http');
2. var fs = require('fs');
3. var url = require('url');
4.
5. const hostname = '127.0.0.1';
6. const port = 3000;
7.
8. const server = http.createServer((req, res) => {
9.   var q = url.parse(req.url, true);
10.  var filename = "." + q.pathname;
11.  fs.readFile(filename, function(err, data) {
12.    if (err) {
13.      res.writeHead(404, {'Content-Type': 'text/html'});
14.      return res.end("404 Not Found");
15.    }
16.    res.writeHead(200, {'Content-Type': 'text/html'});

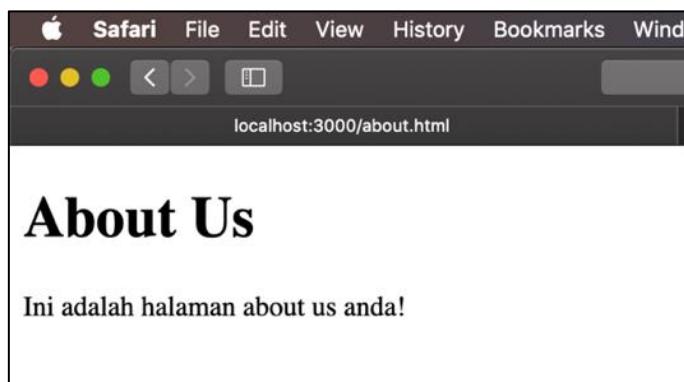
```

```

17.     res.write(data);
18.     return res.end();
19.   });
20. });
21.
22. server.listen(port, hostname, () => {
23.   console.log(`Server running at http://${hostname}:${port}/`);
24. });

```

Selanjutnya jalankan FileServer.js via terminal pada Visual Studio Code anda dengan mengetik “**node FileServer**”. Server akan berjalan, kemudian coba buka pada browser anda dengan mengetikan alamat **http://127.0.0.1/about.html** maka file about.html anda akan terbuka.



Gambar 17. File Server dengan Node.js

J. Node Package Manager (NPM)

Node Package Manager atau yang biasa kita sebut NPM adalah sebuah alat yang dimiliki oleh Node.js untuk melakukan installasi berbagai library atau modul dari Node.js. Modul atau library tersebut dapat berjalan pada aplikasi web Node.js anda untuk melakukan instalasi setiap package modul tersebut bisa dilakukan via terminal. Misalkan ketika kami ingin menginstal template engine yaitu ejs misalnya, kita cukup menginstal dengan mengetik perintah pada terminal dengan “**npm install ejs**” dan kita bisa menginstal berbagai macam kebutuhan library. Anda bisa melihat berbagai macam package yang bisa anda gunakan pada web Node.js di situs <https://www.npmjs.com>.

Sebagai contoh kita akan menggunakan npm chalk yaitu sebuah package yang akan menjadikan console log kita menjadi berwarna. Sekarang kita instal pada

terminal “**npm install chalk**”. Tunggu hingga proses instalasi selesai. Selanjutnya kita buat sebuah ubah file HelloWorld sebagai berikut:

HelloWorld.js

```

1. const http = require('http');
2. var chalk = require('chalk');
3.
4. const hostname = '127.0.0.1';
5. const port = 3000;
6.
7. const server = http.createServer((req, res) => {
8.   res.statusCode = 200;
9.   res.setHeader('Content-Type', 'text/html');
10.  res.write('Hello World');
11.  res.end();
12. });
13.
14. server.listen(port, hostname, () => {
15.   console.log(`Server running at http://${hostname}:${port}/`);
16.   console.log(chalk.blue('Hello') + ' World' + chalk.red('!'));
17. });

```

Penjelasan Code :

- Pada baris ke 2 kita panggil library chalk
- Pada baris ke 16 kita aplikasikan chalk pada console.log sehingga berwarna biru.
- Jalankan dengan mengetik perintah pada terminal “node HelloWorld”

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
ipungs-MacBook-Pro-10:Latihan ipungpurwono$ node HelloWorld
Server running at http://127.0.0.1:3000/
Hello World!

```

Gambar 18. Mengaplikasikan NPM chalk pada Console

K. Email

Node.js juga dapat kita gunakan sebagai alat untuk mengirimkan email. Package yang digunakan adalah Nodemailer. Nodemailer terlebih dahulu harus

didownload via NPM. Dengan mengetikan perintah “**node install nodemailer**” pada terminal anda, maka package sudah bisa dipanggil dan digunakan. Untuk lebih jelasnya bisa kita lihat pada contoh file berikut :

KirimEmail.js

```

1. var nodemailer = require('nodemailer');
2.
3. var transporter = nodemailer.createTransport({
4.   service: 'gmail',
5.   auth: {
6.     user: 'youremail@gmail.com',
7.     pass: 'yourpassword'
8.   }
9. });
10.
11.var mailOptions = {
12.   from: 'youremail@gmail.com',
13.   to: 'myfriend@yahoo.com',
14.   subject: 'Sending Email using Node.js',
15.   text: 'That was easy!'
16. };
17.
18.transporter.sendMail(mailOptions, function(error, info){
19.   if (error) {
20.     console.log(error);
21.   } else {
22.     console.log('Email sent: ' + info.response);
23.   }
24. });

```

Penjelasan Kode :

- Baris 1 kita panggil library Nodemailer
- Baris 3-9 nodemailer akan membuat sebuah transporter dimana akan mendefinisikan email dan password server
- Baris 11-16 buat sebuah array yang berisi kemana email akan dikirim beserta isinya.
- Baris 18-23 transporter server email yang sudah didefinisikan akan mengirim isi konten yang sudah dibuat pada array mailOptions.

Untuk mengirim lebih dari satu penerima , kita bisa mengubah pada array mailOptions seperti berikut

```
1. var mailOptions = {  
2.   from: 'youremail@gmail.com',  
3.   to: 'temanmu@gmail.com, temanmulagi@gmail.com,  
      temanmulain@gmail.com',  
4.   subject: 'Kirim email dengan Node.js!',  
5.   text: 'Ini mudah!'  
6. }
```

Penjelasan Kode :

- Baris 3 kita mendefinisikan beberapa email penerima

Untuk mengirim email dalam bentuk HTML bisa menggunakan setup seperti berikut ini :

```
1. var mailOptions = {  
2.   from: 'youremail@gmail.com',  
3.   to: 'myfriend@yahoo.com',  
4.   subject: 'Kirim email dengan Node.js',  
5.   html: '<h1>Selamat Datang</h1><p>Ini sangat mudah!</p>'  
6. }
```

Penjelasan Kode :

- Pada baris ke 5 kita gunakan html code agar email bisa tersusun lebih rapi.

BAB II

MongoDB Database

A. Berkenalan dengan MongoDB

MongoDB didirikan pada 2007 oleh Dwight Merriman, Eliot Horowitz dan Kevin Ryan - tim di perusahaan DoubleClick. MongoDB adalah platform database modern, tujuan umum tekemuka yang dirancang untuk memaksimalkan kinerja data dan perangkat lunak dalam proses pengembangannya. MongoDB merupakan sebuah model database bergaya NoSQL (Not Only SQL) dimana tidak menggunakan relasi antar tabel dan tidak menyimpannya dalam format tabel baku. MongoDB menggunakan struktur data JSON (Javascript Object Notation) untuk menyimpan datanya. Berikut ini adalah penjelasan singkat mengenai MongoDB:

- MongoDB menyimpan data secara fleksibel, dokumen mirip JSON, yang berarti bidang dapat bervariasi dari dokumen ke dokumen dan struktur data dapat diubah dari waktu ke waktu
- Model dokumen memetakan ke objek dalam kode aplikasi Anda, membuat data mudah digunakan
- Kueri ad-hoc, pengindeksan, dan agregasi waktu nyata memberikan cara yang kuat untuk mengakses dan menganalisis data Anda
- MongoDB adalah basis data terdistribusi pada intinya, sehingga ketersediaan tinggi, skala horizontal, dan distribusi geografis dibangun dan mudah digunakan
- MongoDB gratis dan bersumber terbuka (open source). Versi yang dirilis sebelum 16 Oktober 2018 diterbitkan di bawah AGPL. Semua versi yang dirilis setelah 16 Oktober 2018, termasuk perbaikan patch untuk versi sebelumnya, diterbitkan di bawah Server Side Public Lisensi (SSPL) v1.

B. Installasi MongoDb

MongoDB berjalan secara cross platform pada berbagai sistem operasi seperti Linux, Windows dan Mac. Jadi kita bisa menginstalnya pada Linux, Mac ataupun

Windows. Ada dua versi MongoDB yaitu MongoDB Enterprise dan Community, sebagai alat belajar maka kita akan menggunakan versi Communitynya. Untuk installasinya sebetulnya sudah dijelaskan secara jelas pada web MongoDB yaitu:

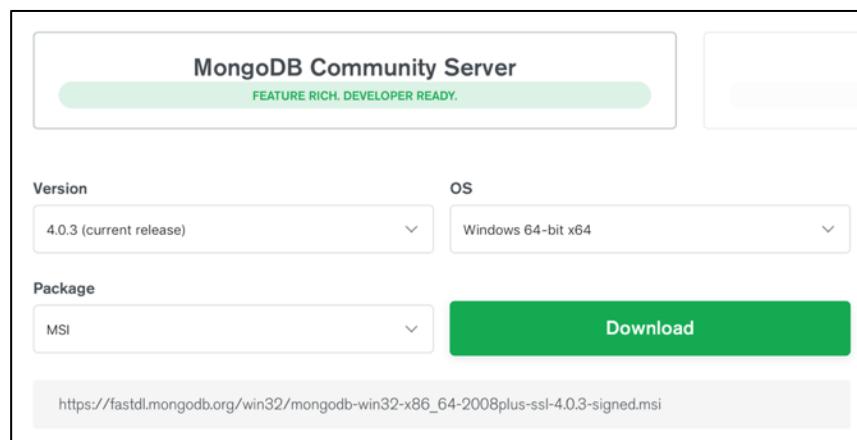
- <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/> untuk windows
- <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/> untuk mac
- <https://docs.mongodb.com/manual/administration/install-on-linux/> untuk installasi pada linux.

Namun dalam prakteknya ada beberapa kendala ketika melakukan installasi mongodb yang membuat kita sedikit pusing. Oleh karena itu akan kami jelaskan proses installasi MongoDB agar lebih mudah diaplikasikan.

Instalasi MongoDB Community pada Windows 10

Untuk melakukan instalasi MongoDB pada windows anda dapat mengikuti cara-cara sebagai berikut:

- Download MongoDB Community Edition Server pada <https://www.mongodb.com/download-center/community?jmp=docs> kemudian anda pilih yang versi windows dan packagenya pilih MSI.



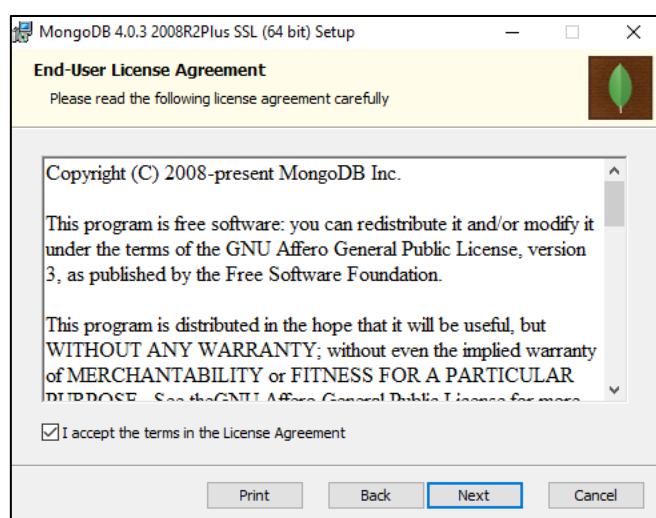
Gambar 19. Memilih MongoDB for Windows

- Tekan button download dan tunggu hingga proses download selesai, biasanya file download akan tersimpan pada folder Downloads di komputer anda.
- Setelah file berhasil terdownload yaitu file .msi, selanjutnya ada bisa klik dua kali untuk melakukan proses instalasi MongoDB pada Windows 10.



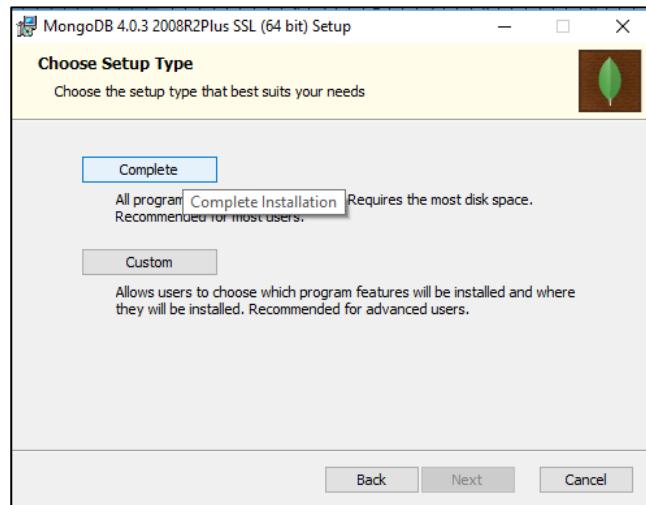
Gambar 20. Instalasi MongoDB

- Pilih Next dan jangan lupa centang agreements nya



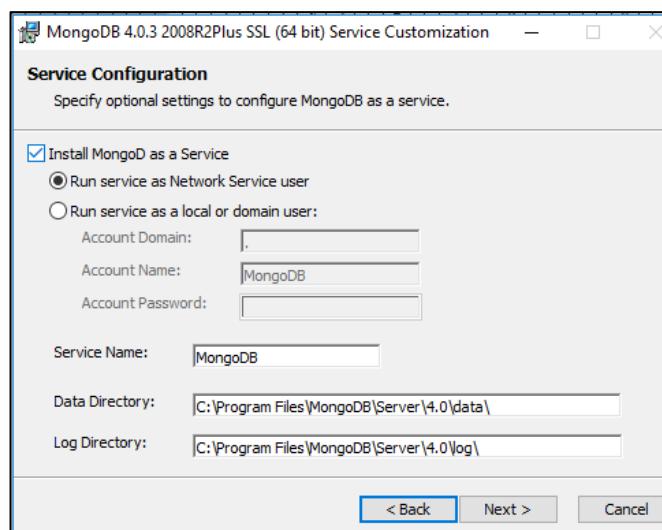
Gambar 21. Agreement MongoDB

- Pada choose setup type pastikan memilih complete setup



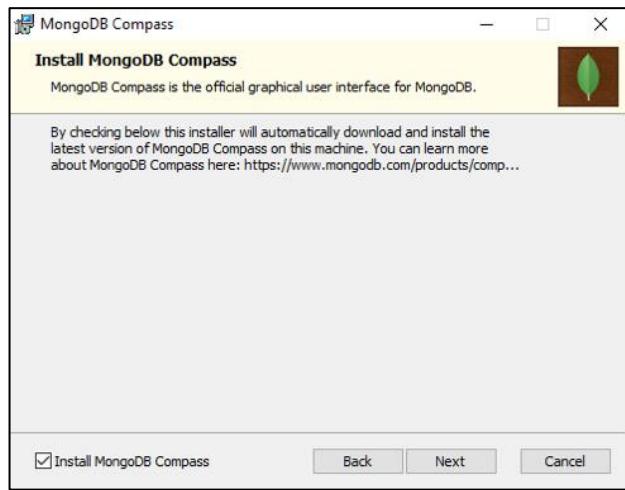
Gambar 22. Memilih Complete Installation

- Pilih complete setup



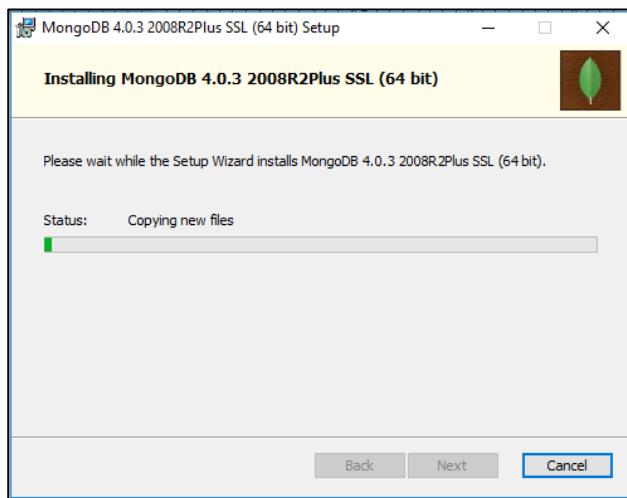
Gambar 23. Service Configuration

- Centang Install MongoDB Compass



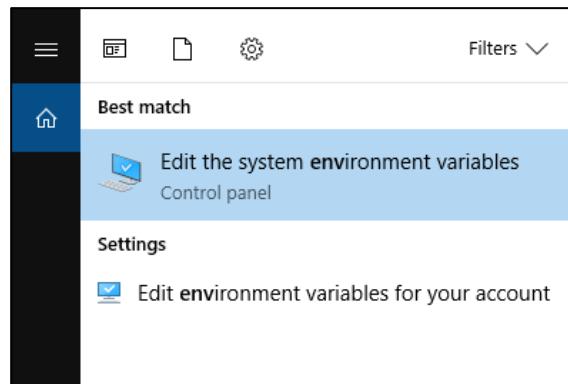
Gambar 24. Checklit MongoDB Compass

- Klik next dan tunggu hingga proses installasi selesai



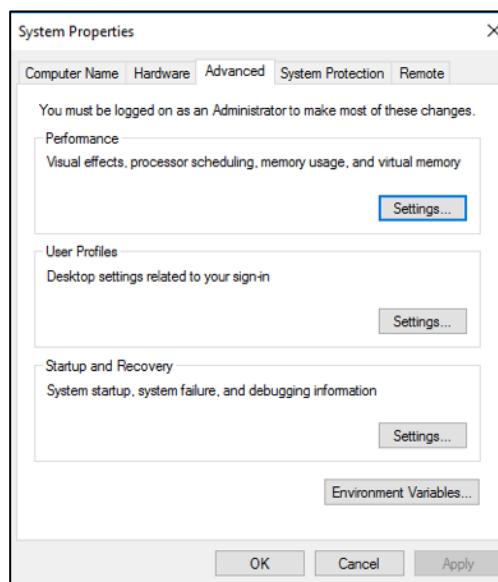
Gambar 25. Proses Installasi MongoDB

- Setelah proses install selesai perlu ada setup Environtment Variable dengan cara klik menu search pda windows 10 dan ketikan Environtment Variables



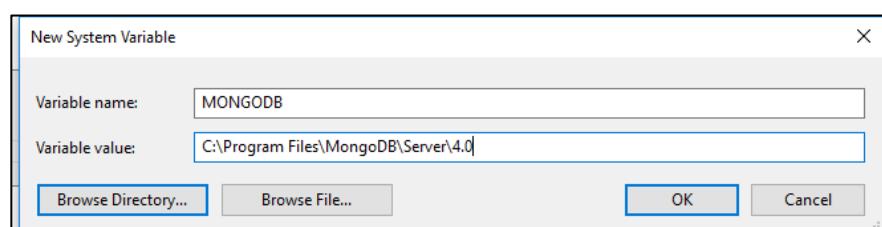
Gambar 26. Edit System Environtment Windows

- Pilih Tab advanced



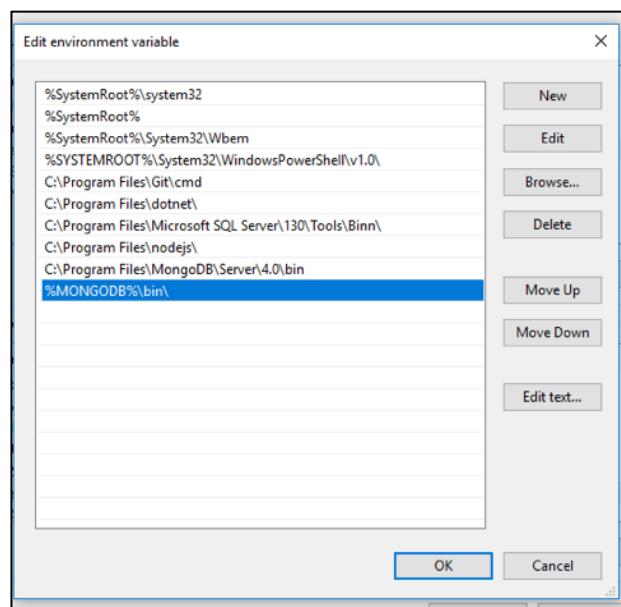
Gambar 27. Memilih Environtment Variables

- Pilih menu Environtment Variables lalu Klik New dan isikan variable baru



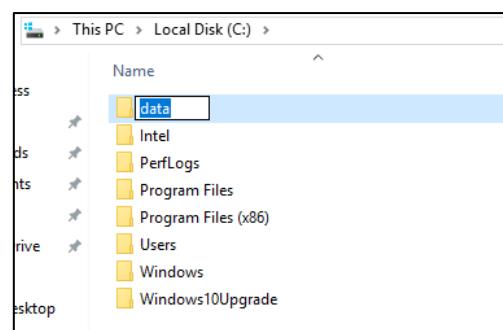
Gambar 28. Membuat New System Variable

- Kemudian cari Variable PATH, pilih edit dan buat Variable baru dengan memilih New. Masukan **%MONGODB%\bin**



Gambar 29. Menambahkan PATH Environtment

- Tekan Ok untuk menyimpan Environtments Variables baru tersebut, Kemudian buat sebuah folder data/db pada drive C.



Gambar 30. Membuat Folder Data

- Di dalam folder data pastikan Anda sudah membuat folder db, tujuannya adalah folder tersebut sebagai tempat penyimpanan database MongoDB.
- Selanjutnya buka Command Prompt dan ketikan perintah “**mongod**”

```
C:\Users\ipung>mongod
2018-11-07T17:51:10.787+0700 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2018-11-07T17:51:11.133+0700 I CONTROL [initandlisten] MongoDB starting : pid=2816 port=27017 dbpath=C:\data\db\ 64-bit
host=DESKTOP-3AVDAB1
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten] db version v4.0.3
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten] allocator: tcmalloc
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten] modules: none
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten] build environment:
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten]     distmod: 2008plus-ssl
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten]     distarch: x86_64
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten]     target_arch: x86_64
2018-11-07T17:51:11.134+0700 I CONTROL [initandlisten] options: {}
2018-11-07T17:51:11.134+0700 I STORAGE [initandlisten] WiredTiger open config: create,cache_size=3520M,session_max=2000
0,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal
,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2018-11-07T17:51:11.381+0700 I STORAGE [initandlisten] WiredTiger message [1541587871:381079][2816:140723793450720], tx
n-recover: See global recovery timestamp: 0
2018-11-07T17:51:11.490+0700 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2018-11-07T17:51:11.730+0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-11-07T17:51:11.731+0700 I CONTROL [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2018-11-07T17:51:11.736+0700 I CONTROL [initandlisten]
2018-11-07T17:51:11.737+0700 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-11-07T17:51:11.739+0700 I CONTROL [initandlisten] **          Remote systems will be unable to connect to this ser
ver.
```

Gambar 31. Server MongoDB Berjalan

Selamat Anda sudah berhasil menginstal MongoDB pada komputer anda

Instalasi MongoDB Community pada Mac

Untuk melakukan instalasi MongoDB pada MacOSX anda dapat mengikuti cara-cara sebagai berikut:

- Download MongoDB Community Edition Server pada <https://www.mongodb.com/download-center/community?jmp=docs> kemudian anda pilih yang versi mac dan packagenya pilih TGZ.

**Gambar 32.** Download MongoDB for Mac

- Tekan button download dan tunggu hingga proses download selesai, biasanya file download akan tersimpan pada folder Downloads di Mac anda.

- Setelah download file .tgz sekarang kita akan menginstalnya via terminal. Buka terminal mac anda.
- Setelah terbuka terminal nya ketikan perintah “**cd/Downloads**” lalu enter dimana akan merubah ke direktori Downloads. Setelah itu ketik perintah “**ls**” untuk melihat semua direktori di folder Downloads.

```
ipungs-MacBook-Pro-56:~ ipungpurwono$ cd Downloads
ipungs-MacBook-Pro-56:Downloads ipungpurwono$ ls
backup mongodb-osx-ssl-x86_64-4.0.3.tar
ipungs-MacBook-Pro-56:Downloads ipungpurwono$
```

Gambar 33. Melihat isi direktori di Downloads Folder

- Dari perintah “**ls**” kita melihat file **mongodb-osx-ssl-x86_64-4.0.3.tar** yaitu installer MongoDB for Mac, untuk itu kita harus mengekstraknya, cara mengekstraknya ketikan perintah “**tar xzf mongodb-osx-ssl-x86_64-4.0.3.tar**” sehingga akan terekstact menjadi folder. Lalu ketikan perintah “**ls**” untuk melihat foldernya.

```
ipungs-MacBook-Pro-56:~ ipungpurwono$ cd Downloads
ipungs-MacBook-Pro-56:Downloads ipungpurwono$ ls
backup mongodb-osx-ssl-x86_64-4.0.3.tar
ipungs-MacBook-Pro-56:Downloads ipungpurwono$ tar xzf mongodb-osx-ssl-x86_64-4.0.3.tar
ipungs-MacBook-Pro-56:Downloads ipungpurwono$ ls
backup mongodb-osx-x86_64-4.0.3
mongodb-osx-ssl-x86_64-4.0.3.tar
ipungs-MacBook-Pro-56:Downloads ipungpurwono$
```

Gambar 34. Mengekstract file tar ke dalam folder

- Setelah berhasil diekstrak kita akan memindahkannya folder hasil ekstract tersebut ke direktory “**/usr/local/mongodb**” nah tujuannya adalah agar kita mudah ketika menjalankan server MongoDB via terminal nantinya. Untuk memindahkannya kita harus mengetikan perintah “**sudo mv mongodb-osx-x86_64-4.0.3 /usr/local/mongodb**”

karena menggunakan sudo berarti kita bertindak sebagai super administrator, maka wajib menginputkan password.

```
ipungs-MacBook-Pro-56:Downloads ipungpurwono$ ls
backup                               mongodb-osx-x86_64-4.0.3
mongodb-osx-ssl-x86_64-4.0.3.tar
ipungs-MacBook-Pro-56:Downloads ipungpurwono$ sudo mv mongodb-osx-x86_64-4.0.3 /usr/local/mongodb
ipungs-MacBook-Pro-56:Downloads ipungpurwono$
```

Gambar 35. Memindahkan folder ke /usr/local/mongodb

- Sekarang anda sudah berhasil memindahkan file MongoDB ke local komputer Mac, untuk melihatnya kita change direktori ke mongodb folder. Cara nya ketikan perintah “**cd /usr/local/mongodb**” lalu ketikan perintah “**ls**”

```
ipungs-MacBook-Pro-56:Downloads ipungpurwono$ cd /usr/local/mongodb
ipungs-MacBook-Pro-56:mongodb ipungpurwono$ ls
GNU-AGPL-3.0                         README
LICENSE-Community.txt                  THIRD-PARTY-NOTICES
MPL-2                                    bin
ipungs-MacBook-Pro-56:mongodb ipungpurwono$
```

Gambar 36. File telah berpindah ke folder mongodb

- Kita buat sebuah folder data untuk menampung database MongoDB yang tersimpan ketikan perintah berikut “**sudo mkdir -p /data/db**”. Cobalah untuk mengetikan perintah “**ls**” disana anda tidak akan melihat folder data karena dihidden, namun anda bisa mengakses dengan cara change direktori “**cd /data//db**”

```
[ipungs-MacBook-Pro-56:mongodb ipungpurwono$ sudo mkdir -p /data/db
[ipungs-MacBook-Pro-56:mongodb ipungpurwono$ cd /data//db
[ipungs-MacBook-Pro-56:db ipungpurwono$ pwd
/data/db
```

Gambar 37. Membuat folder data/db untuk menyimpan database

- Agar folder db dapat diakses kita harus memberikan permission dengan cara mengetikan perintah “**sudo chown namakomputermu /data/db**”. Namakomputermu dapat anda cek dengan mengetikan perintah “**whoami**”.

```
[ipungs-MacBook-Pro-56:mongodb ipungpurwono$ cd /data/db
[ipungs-MacBook-Pro-56:db ipungpurwono$ pwd
/data/db
[ipungs-MacBook-Pro-56:db ipungpurwono$ sudo chown ipungpurwono /data/db
```

Gambar 38. Mengubah permission pada folder data/db

- Setelah berhasil mengubah permission pada db, sekarang anda ketikan perintah “**cd**” sehingga akan mengarahkan pada home komputer anda. Kemudian ketikan perintah “**ls -al**” perintah tersebut akan kita gunakan untuk mengecek apakah sudah memiliki file **.bash_profile** ? Jika belum anda bisa membuatnya dengan mengetik perintah “**touch .bash_profile**”, sedangkan jika sudah anda tinggal open file **.bash_profile** saja.

```
ipungs-MacBook-Pro-56:db ipungpurwono$ cd
ipungs-MacBook-Pro-56:~ ipungpurwono$ ls -al
total 75384
drwxr-xr-x+ 101 ipungpurwono staff      3232 Nov  7 14:19 .
drwxr-xr-x  6 root      admin       192 Sep 27 20:41 ..
-rw-----  1 ipungpurwono staff       3 Nov  1 2014 .CFUserTextEncoding
-rw-r--r--@ 1 ipungpurwono staff     51204 Nov  7 14:53 .DS_Store
drwxr-xr-x  4 ipungpurwono staff      128 Feb 23 2018 .Deco
drwxr-xr-x  6 ipungpurwono staff      192 Oct  3 14:17 .IdentityService
drwxr-xr-x  2 ipungpurwono staff      64 Jun  2 2017 .InstallAnywhere
-rw-r--r--  1 ipungpurwono staff      11 Nov  4 2014 .JavaW
drwx----- 4 ipungpurwono staff      128 Oct 18 13:27 .ServiceHub
drwx----- 8 ipungpurwono staff      256 Nov  7 14:42 .Trash
-rw-----  1 ipungpurwono staff      0 Dec 14 2017 .Xauthority
drwxr-x--- 4 ipungpurwono staff      128 May  9 2016 .adobe
drwxr-xr-x 30 ipungpurwono staff      960 Oct 22 22:00 .android
drwxr-xr-x  3 ipungpurwono staff      96 Dec 30 2017 .aspnet
-rw-r--r--  1 ipungpurwono staff  35013134 Nov  1 2017 .babel.json
-rw-----  1 ipungpurwono staff      8758 Nov  1 17:16 .bash_history
-rw-r--r--@ 1 ipungpurwono staff      1018 Sep 27 2017 .bash_profile
-rw-r--r--@ 1 ipungpurwono staff      521 Aug  1 2017 .bash_profile.pysave
drwx----- 41 ipungpurwono staff     1312 Nov  7 14:39 .bash_sessions
drwxr-xr-x  8 ipungpurwono staff      256 Aug 20 14:27 .cache
drwxr-xr-x  4 ipungpurwono staff      128 Aug 15 2017 .composer
```

Gambar 39. Ls -al untuk mengecek file .bash_profile

Dari gambar di atas sudah terlihat bahwa , kami sudah memiliki file **.bash_profile**, oleh karena itu tinggal open dan mengeditnya. Sekali lagi jika anda belum memilikinya anda harus membuatnya dengan ketik perintah “**touch .bash_profile**” lalu baru “**open .bash_profile**”.

- Setelah dipastikan file **.bash_profile** sudah tersedia kita lakukan open dengan perintah “**open .bash_profile**”, maka akan terbuka code editor.

```

.bash_profile — Edited
PATH="/Library/Frameworks/Python.framework/Versions/3.5/bin:${PATH}"
export PATH

# Setting PATH for Python 2.7
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/2.7/bin:${PATH}"
export PATH

# Setting PATH for Python 3.6
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"
export PATH

# Setting PATH for Python 3.7
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.7/bin:${PATH}"
export PATH

```

Gambar 40. Membuka .bash_profile

- Tambahkan beberapa baris code berikut di bawah pada file .bash_profile.

```

export MONGO_PATH=/usr/local/mongodb
export PATH=$PATH:$MONGO_PATH/bin

```

Setelah ditambah, kemudian pilih file - save atau (ctrl + s) pada keyboard anda.

- Restart terminal anda, lalu buka dua tabs terminal. Pada tab pertama ketikan perintah “**mongod**” untuk menjalankan server. Dan pada tab kedua ketikan perintah “**mongo**” untuk menjalankan database mongodb.

```

ipungpurwono — mongod — 80x24
Last login: Wed Nov  7 15:53:26 on ttys011
-bash: ~source: command not found
[ipungs-MacBook-Pro-56:~ ipungpurwono$ mongod
2018-11-07T15:57:44.408+0700 I CONTROL [main] Automatically disabling TLS 1.0,
to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2018-11-07T15:57:44.476+0700 I CONTROL [initandlisten] MongoDB starting : pid=2437
port=27017 dbpath=/data/db 64-bit host=ipungs-MacBook-Pro-56.local
2018-11-07T15:57:44.476+0700 I CONTROL [initandlisten] db version v4.0.3
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] git version: 7ea530946fa
7880364d88c8d8b6026bbc9ffa48c
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] allocator: system
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] modules: none
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] build environment:
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten]     distarch: x86_64
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten]     target_arch: x86_64
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] options: {}
2018-11-07T15:57:44.479+0700 I STORAGE [initandlisten] Detected data files in /
data/db created by the 'wiredTiger' storage engine, so setting the active storage
engine to 'wiredTiger'.
2018-11-07T15:57:44.479+0700 I STORAGE [initandlisten] wiredtiger_open config:
create,cache_size=3584M,session_max=20000,eviction=(threads_min=4,threads_max=4)
,config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal
,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0
),verbose=(recovery_progress),

```

Gambar 41. Perintah mongod untuk jalankan server

```

ipungpurwono — mongod — 80x24
Last login: Wed Nov  7 15:53:26 on ttys011
-bash: ~source: command not found
[ipungpurwono-MacBook-Pro-56:~ ipungpurwono$ mongo]
2018-11-07T15:57:44.408+0700 I CONTROL [main] Automatically disabling TLS 1.0,
to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2018-11-07T15:57:44.476+0700 I CONTROL [initandlisten] MongoDB starting : pid=7
2437 port=27017 dbpath=/data/db 64-bit host=ipungpurwono-MacBook-Pro-56.local
2018-11-07T15:57:44.476+0700 I CONTROL [initandlisten] db version v4.0.3
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] git version: 7ea530946fa
7880364d88c8d8b6026bbc9ffa48c
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] allocator: system
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] modules: none
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] build environment:
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten]   distarch: x86_64
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten]   target_arch: x86_64
2018-11-07T15:57:44.477+0700 I CONTROL [initandlisten] options: {}
2018-11-07T15:57:44.479+0700 I STORAGE [initandlisten] Detected data files in /
data/db created by the 'wiredTiger' storage engine, so setting the active storage
engine to 'wiredTiger'.
2018-11-07T15:57:44.479+0700 I STORAGE [initandlisten] wiredtiger_open config:
create,cache_size=3584M,session_max=20000,eviction=(threads_min=4,threads_max=4)
,config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal
,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0
),verbose=(recovery_progress),

```

Gambar 42. Perintah mongo untuk jalankan database mongo client

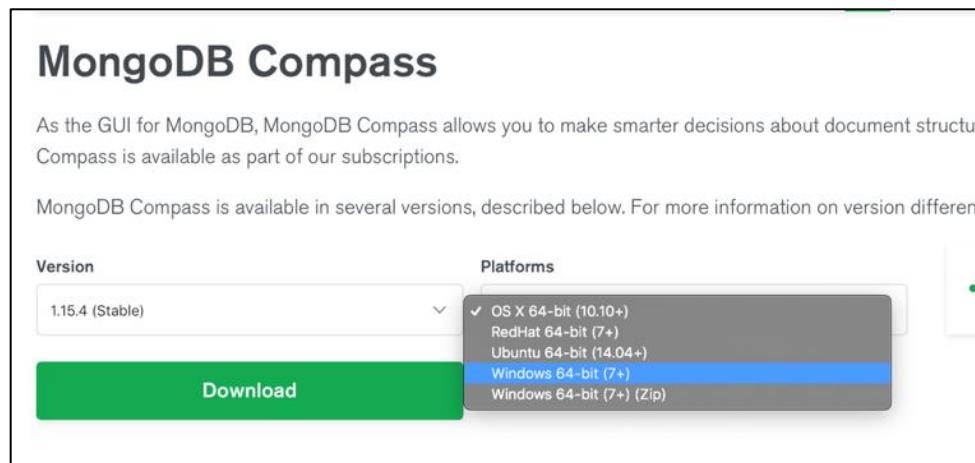
Instalasi MongoDB Community pada Linux

Untuk melakukan instalasi MongoDB pada Linux anda dapat mengikuti tutorial pada web resmi <https://docs.mongodb.com/manual/administration/install-on-linux/>. Seperti kita tahu bahwa linux memiliki banyak versi dari Ubuntu, Redhat, Debian dan sebagainya. Untuk installasi pada linux sendiri sebetulnya hampir mirip dengan installasi pada Mac karena sama-sama Operating System bertipe Unix. Installasi lebih banyak menggunakan terminal.

- **Ubuntu:** <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>
- **Red Hat:** <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/>
- **Debian:** <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-debian/>
- **SUSE:** <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-suse/>

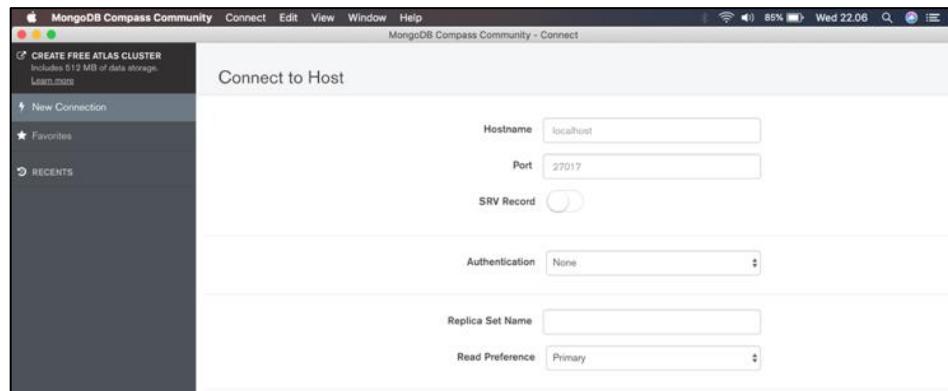
C. MongoDB Compass (GUI)

Untuk mempermudah kita dalam menggunakan MongoDB yang berbasis data dalam bentuk JSON, MongoDB sebetulnya sudah menyediakan GUI versi Communitynya juga yaitu MongoDB Compass. Dengan MongoDB Compass kita akan lebih mudah mengelola database MongoDB seperti kita menggunakan PHP MyAdmin di Xampp. MongoDB Compass dapat didownload secara langsung pada link <https://www.mongodb.com/download-center/compass>. Anda bisa memilih Compass sesuai dengan Operating System anda dan pastikan Version yang Community.



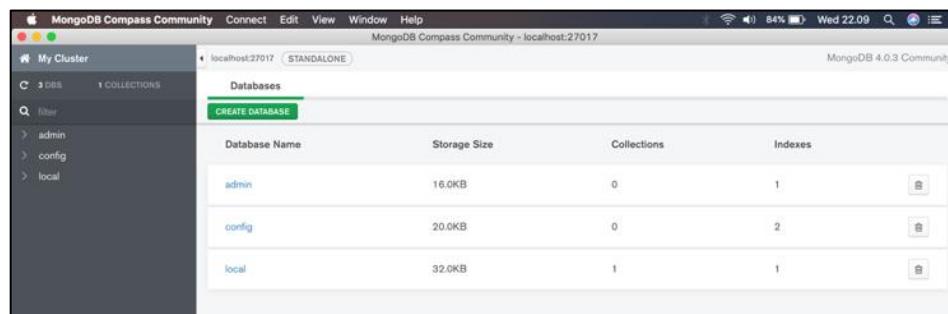
Gambar 43. Mendownload MongoDB Compass

Setelah anda mendownloadnya, lakukan installasi seperti pada umumnya maksudnya instal MongoDB Compass tidak sulit, dimana tidak perlu konfigurasi seperti anda menginstal MongoDB server.



Gambar 44. Tampilan Awal MongoDb Compass

Agar kita terkoneksi ke MongoDb server klik connect, isi field form secara default. Jika anda sudah tekoneksi maka anda akan menuju ke halaman dashboard MongoDB Compass.



Gambar 45. Halaman Dashboard MongoDB Compass

Dengan MongoDb Compass kita bisa membuat Database, Collection ataupun Operasi manipulasi data Documents melalui GUI ini dengan mudah.

D. Membuat Database dan Collection MongoDB

Sebelum kita membuat database MongoDB kita akan melihat berbagai perbedaan antara MongoDB yang merupakan NoSQL dengan MySQL dalam tabel berikut:

Tabel 2. Perbedaan MongoDB vs MySQL

MySQL	MongoDB
Database	Database

Table	Collections
Row	Documents
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key Auto Generate

Yang paling menonjol menurut kami adalah Table Join vs Embedded Documents, Sebagai contoh sebuah Kecamatan yaitu Gumelar memiliki Tiga Desa yaitu Tлага, Paningkaban dan Gancang. Jika dalam MySQL maka struktur menjadi seperti berikut:

Id (Primary Key)	nama_desa	Id_kecamatan (foreign key)
1	Tлага	1
2	Gancang	1
3	Paningkaban	1

Angka 1 merupakan relasi dari table kecamatan

Id (Primary Key)	nama_kecamatan
1	Gumelar
2	Ajibarang
3	Karang Lewas

Nah, dalam MySQL tabel Desa dimana id_kecamatan akan berelasi dengan table kecamatan. Sedangkan dalam **MongoDb** databasenya akan terformat sebagai berikut:

```

1. {
2.   "_id": ObjectId("52ffc33cd85242f43600001"),
3.   "nama_kecamatan": "Gumelar",
4.   "desa": [
5.     {
6.       "id_desa": "52ffc33cd85242f436000011",
7.       "nama_desa": "Tлага",

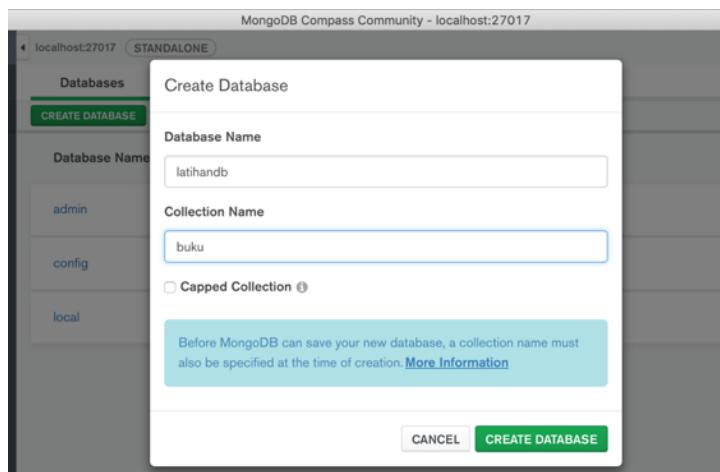
```

```

8.      },
9.      {
10.         "id_desa": "52ffc33cd85242f436000012",
11.         "nama_desa": "Gancang",
12.     },
13.     {
14.         "id_desa": "52ffc33cd85242f436000013",
15.         "nama_desa": "Kedung Urang",
16.     }
17.   ]
18. }
```

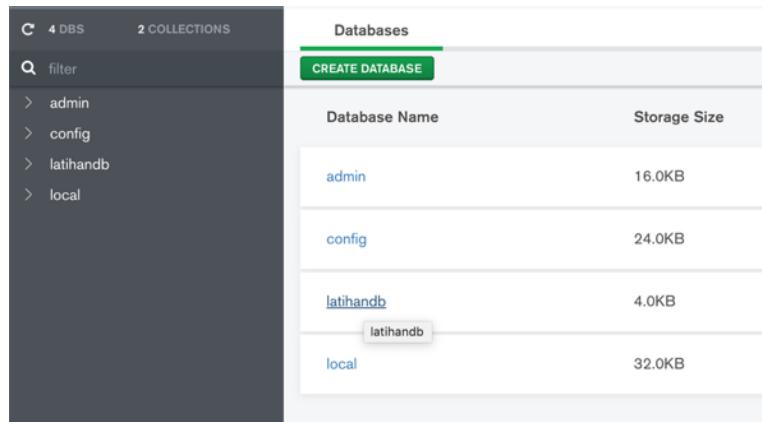
Jadi, dalam MongoDB bentuknya adalah JSON dan relasinya bersifat Embedded.

Nah untuk membuat Database lebih cepat kita menggunakan MongoDb Compass, caranya pada halaman MongoDB Compass yang sudah terkoneksi, klik menu Create Database, lalu masukan nama database “latihandb” dan collection “buku”. Nama database dan collection sebagai latihan kita ya.

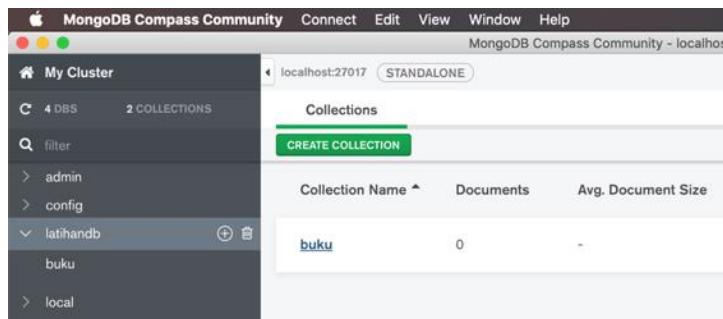


Gambar 46. Create Database MongoDB

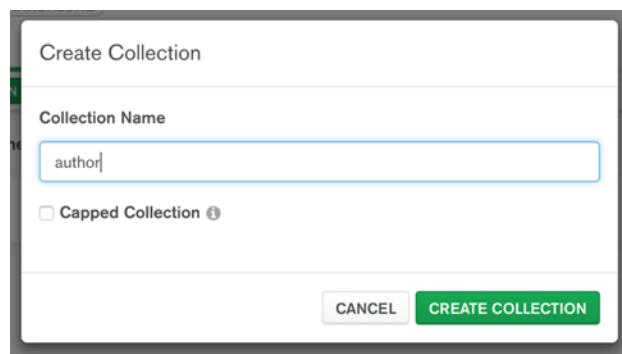
Jika sudah pilih tombol Create Database, maka anda sudah memiliki sebuah database dengan nama latihandb.

**Gambar 47.** Database Berhasil dibuat

Untuk melihat collection dalam database **latihandb** bisa di klik nama databasenya.

**Gambar 48.** Collection dalam Database latihandb

Anda juga bisa membuat collection lain dengan memilih klik Create Collection, lalu masukan nama Collection, misalkan saya buat nama collection baru "**author**"

**Gambar 49.** Membuat collection baru

Untuk membuat Documents, anda cukup klik nama collection, misalkan kita klik collection author dan masukan beberapa nama author. Caranya adalah klik tombol **Insert Document** lalu masukan beberapa nama author pada Document sebagai berikut.



Gambar 50. Membuat document baru

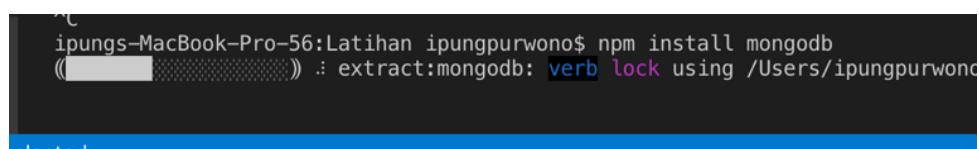
Pada saat anda memasukan document , anda akan melihat `_id` akan tergenerate secara otomatis dan pada `ObjectId` anda bisa melihat tipe data yang disupport oleh MongoDB baik dari String, Datae, Number, Array dan sebagainya. Jika sudah cukup isi documentnya klik insert, maka document baru telah berhasil ditambahkan.

	_id	nama	email	tanggal_lahir
1	ObjectId("5be3084e9929bb32d6bc5aae")	Ipung Purwono	ipungofficial@gmail.com	1989-05-16 00:00:00.000
2	ObjectId("5be309b99929bb32d6bc5aaaf")	Meli Sofiyana	meliso@gmail.com	2016-09-03 00:00:00.000
3	ObjectId("5be309e39929bb32d6bc5ab0")	Keenar A P	keenarAzPr12@yahoo.com	2015-07-02 00:00:00.000

Gambar 51. Menambahkan Document pada Collection

E. MongoDB Insert

Selain menggunakan MongoDB Compass kita juga bisa menggunakan Node.js sebagai alat untuk memasukan data di MongoDb. Pertama kita harus menginstall MongoDB drivernya agar bisa terkoneksi dengan Node.js. Berita bagusnya si MongoDB ini seolah diciptakan berjodoh dengan Node.js, jadi membangun aplikasi dengan Node.Js dan MongoDb akan menjadi semakin baik dan cepat. Untuk bisa memasukan data via Node.js terlebih dahulu harus melakukan installasi via NPM melalui terminal. Kami asumsikan terminal masih pada posisi folder Latihan. Ketikan perintah “npm install mongodb” dan tunggu hingga proses installasi selesai.



Gambar 52. Menambahkan package mongodb

Setelah terinstal, berarti anda sudah bisa memanggil modul tersebut untuk digunakan sebagai alat manipulasi data pada database MongoDB dengan Node.Js

Insert Satu Document

Sebagai contoh kita akan memasukan sebuah data pada database “latihandb” dan collection “author”. Database tersebut sudah kita buat sebelumnya melalui MongoDB Compass.

Insert.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true }, function(err,
   db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var myobj = { nama: "Puji Priono",
8.     email: "puji@gmail.com",
9.     tanggal_lahir:new Date("2016-05-18T16:00:00Z") };

```

```

10. dbo.collection("author").insertOne(myobj, function(err, res) {
11.   if (err) throw err;
12.   console.log("1 document inserted");
13.   db.close();
14. });
15. );

```

Penjelasan Code:

- Baris 1 kita panggil modul mongodb dan jalankan services mongodb client.
- Baris 2 definisikan url server mongodb berjalan
- Baris 4 Koneksikan file Node.js dengan MongoDB server
- Baris 6 Definisikan database yang dituju yaitu “latihandb”
- Baris 7 – 9 query yang akan dimasukan ke documents
- Baris 10 masukan query tersebut ke dalam collections “author”
- Baris 12 tampilkan log jika berhasil diinput
- Baris 13 database di close

Jalankan perintah “node Insert” pada terminal anda, sekarang kita cek pada MongoDB Compass apakah data tersebut masuk atau tidak.



Gambar 53. Menambahkan data di collection

Ternyata anda sudah berhasil menambahkan document baru pada collection.

Insert Multi Document

Node.Js mendukung penuh untuk memasukan document banyak sekaligus, hal ini tentunya akan mempermudah kinerja sebuah sistem dalam penanganan manipulasi data.

Insertbanyak.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true },function(err,
   , db) {
5.     if (err) throw err;
6.     var dbs = db.db("latihandb");
7.     var arrData = [
8.         { nama: 'Tri Setyo', email: 'try@teri.com'},
9.         { nama: 'Fajar Pamiliu', email: 'gojeng@yahoo.com'},
10.        { nama: 'Imam Subekti', email: 'imamjava@gmail.com'}
11.    ];
12.    dbs.collection("author").insertMany(arrData, function(err,
   res) {
13. if (err) throw err;
14. console.log("beberapa dokument
 inserted:" + res.insertedCount);
15.     dbs.close();
16.    });
17. });

```

Penjelasan Code

- Baris 7 tampung data dalam sebuah array myObj
- Baris 12 masukan data array tersebut ke dalam collection “author” dengan fungsi **insertMany**

Jika anda cek di MongoDB Compass anda akan menemukan 3 data baru masuk pada collection anda.

F. MongoDB Find

Find sebuah kata dalam bahasa inggris yang artinya cari, dalam MongoDB find digunakan untuk menampilkan data field, seperti select kalau dalam MySQL. Misalkan select nama, email from tb_user. Nah pada MongoDB bisa menggunakan find. Find pada MongoDB dapat berfungsi pada kondisi seperti berikut ini:

Find One

Fungsinya adalah akan menyeleksi data dari sebuah collection yang terpilih dimana data yang dihasilkan berada pada order yg pertama. Misalkan kita dalam collection author punya lima data, maka dengan Find One, data yang akan tampil adalah data pertama.

FindOne.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url,{ useNewUrlParser: true }, function(err,
   , db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   dbo.collection("author").findOne({}, function(err, result) {
8.     if (err) throw err;
9.     console.log(result.nama);
10.    console.log(result.email);
11.    db.close();
12.  });
13. });

```

Penjelasan code:

- Baris ke 7 kita gunakan fungsi **findOne()** untuk menyeleksi data pertama (One) dari collection author.
- Baris 8-9 tampilkan dalam log Column nama dan email
- Jalankan dengan mengetikan perintah “**node FindOne**” dan anda akan menemukan hasil bahwa data pertama dengan seleksi nama dan email saja pada console log.

***note: setelah ini, setiap kode node js bisa anda jalankan dengan perintah “node namafilejsnya” kami tidak akan mengulang untuk menulis perintah tersebut, karena kami asumsikan bahwa anda sudah memahami cara menjalankan perintah node.js**

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

ipungs-MacBook-Pro-56:Latihan ipungpurwono$ node FindOne
Ipung Purwono
ipungofficial@gmail.com
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ █

```

Gambar 54. Menggunakan fungsi findOne()

Find All (Menyeleksi semua data)

Fungsinya adalah akan menyeleksi semua data datam suatu collection, fungsi ini mirip dengan **SELECT * FROM TABLE** pada MySQL.

FindAll.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true },function(err,
   , db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   dbo.collection("author").find({}).toArray(function(err, result) {
8.     if (err) throw err;
9.     console.log(result);
10.    db.close();
11.  });
12. });

```

Penjelasan code:

- Baris ke 7 kita gunakan fungsi **find({})** untuk menampilkan semua data document pada collection author.
- Baris 9 tampilkan data hasil seleksi find ke dalam console
- Jalankan file untuk melihat hasilnya.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

ipungs-MacBook-Pro-56:Latihan ipungpurwono$ node FindAll
[ { _id: 5be3122e05c5263cda23d7e8,
  nama: 'Ipung Purwono',
  email: 'ipungofficial@gmail.com',
  tanggal_lahir: 1989-05-16T00:00:00.000Z },
  { _id: 5be3084e9929bb32d6bc5aae,
  nama: 'Meli Sofiyana',
  email: 'meliso@gmail.com',
  tanggal_lahir: 2016-09-03T00:00:00.000Z },
  { _id: 5be309b99929bb32d6bc5aab,
  nama: 'Keenar A P',
  email: 'keenarAzPr12@yahoo.com',
  tanggal_lahir: 2015-07-02T00:00:00.000Z },
  { _id: 5be3122e05c5263cda23d7e8,
  nama: 'Puji Priono',
  email: 'puji@gmail.com',
  tanggal_lahir: 2016-05-18T16:00:00.000Z },
  { _id: 5be21620040a6c2ef0444f4f
]

```

Gambar 55. Menggunakan fungsi find()

Find Some (Menyeleksi beberapa column)

Fungsinya adalah akan menyeleksi beberapa data column suatu collection, fungsi ini mirip dengan **SELECT (nama field1, namafield2) FROM TABLE** pada MySQL. Biasanya digunakan untuk meringankan kinerja database dalam mengakses / mempilkan banyak data dengan meminimalisir penggunaan memori dan berdampak pada kecepatan load data.

FindSome.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true },function(err,
   , db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   dbo.collection("author").find({},
8.     { projection: { _id: 0, nama: 1, email: 1 } }).toArray(function(
   err, result) {
9.     if (err) throw err;
10.    console.log(result);
11.    db.close();
12.  });
13. });

```

Penjelasan Code

- Baris ke 7 kita gunakan fungsi **find({})** untuk menampilkan semua data document pada collection author.
- Baris 8 kita gunakan projection, terlihat ada angka 0 pada `_id`, dan angka 1 pada `nama` dan `email`, hal itu bertujuan jika 0, maka tidak perlu ditampilkan sedangkan jika 1 maka tampilan.
- Baris 10 tampilkan data dalam log.
- Jalankan filenya.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ node FindSome
[ { nama: 'Ipung Purwono', email: 'ipungofficial@gmail.com' },
  { nama: 'Meli Sofiyana', email: 'meliso@gmail.com' },
  { nama: 'Keenar A P', email: 'keenarAzPr12@yahoo.com' },
  { nama: 'Puji Priono', email: 'puji@gmail.com' },
  { nama: 'Tri Setyo', email: 'try@teri.com' },
  { nama: 'Fajar Pamilu', email: 'gojeng@yahoo.com' },
  { nama: 'Imam Subekti', email: 'imamjava@gmail.com' } ]
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ █
Share  Server not selected

```

Gambar 55. Menampilkan beberapa column

Pada gambar, ditampilkan data JSON yaitu `nama` dan `email` sedangkan `_id` tidak karena di set 0 sedangkan `nama` dan `email` = 1.

G. MongoDB Query

Pada MongoDB tersedia penggunaan query yang bertujuan untuk mempermudah melakukan filter data, Biasanya filter digunakan ketika kita hendak menampilkan data yang diinginkan seperti menu search dalam website misalnya. Fungsi query ini tidak terlepas dari penggunaan dari `find()`.

Query.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url,{ useNewUrlParser: true }, function(
  err, db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var query = { email: "imamjava@gmail.com" };

```

```

8.   dbo.collection("author").find(query).toArray(function(err,
  result) {
9.     if (err) throw err;
10.    console.log(result);
11.    db.close();
12.  });
13. });

```

Keterangan Code:

- Baris 7 kita tentukan query yang hendak difilter, misalkan kami ingin mencari seseorang dengan nama yang emailnya adalah imamjava@gmail.com.
- Baris 8 kita implementasikan dalam fungsi find(query)
- Baris 10 tampilkan dalam log.

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

ipungs-MacBook-Pro-56:Latihan ipungpurwono$ node Query
[ { _id: 5be31639049ef63ef04445e1,
  nama: 'Imam Subekti',
  email: 'imamjava@gmail.com' } ]
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ █

```

Below the terminal, there is a blue status bar with the text "Share" and "Server not selected".

Gambar 56. Menampilkan hasil filter query

Query Filter dengan Regular Expression

Kita bisa memanfaatkan regular expression untuk melakukan filter tertentu misalnya melakukan filter nama seseorang dengan nama depan “M”

QueryReg.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url,{ useNewUrlParser: true }, function(err
  , db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var query = { nama: /^M/ };
8.   dbo.collection("author").find(query).toArray(function(err, re
  sult) {

```

```

9.      if (err) throw err;
10.     console.log(result);
11.     db.close();
12.   });
13. });

```

Penjelasan Code:

- Baris 7 kita memanfaatkan Regular Expression untuk melihat hasil document dengan nama author dengan nama depan “M”.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
nama: 'Muhammad',
email: 'muhammad@gmail.com' } ]
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ node QueryReg
[ { _id: 5be309b99929bb32d6bc5aaf,
  nama: 'Meli Sofiyana',
  email: 'meliso@gmail.com',
  tanggal_lahir: 2016-09-03T00:00:00.000Z },
{ _id: 5be508819929bb32d6bc5ab1,
  nama: 'Muhammad',
  email: 'muhammad@gmail.com' } ]
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ █

```

Gambar 57. Menampilkan hasil filter query regex

Kita juga bisa memanfaatkan regular expression untuk melakukan filter tertentu dimana fungsinya seperti %like% pada MySQL. Sebagai contoh ingin menampilkan data hanya email yang dibelakangnya bertipe “@gmail.com” saja, Maka kita bisa memanfaatkan regular expression “/@gmail.com/”.

```
var query = { "email": /@gmail/ };
```

Query Filter menggunakan Operator AND dan OR

Dalam MongoDB penciptaan Query bisa melibatkan AND dan OR, hal ini akan bermanfaat ketika anda membandingkan beberapa kondisi pada sebuah query. Untuk AND anda cukup memberikan tanda „,“ (koma) saja sedangkan untuk OR menggunakan \$or.

Penggunaan AND

```
var query = { "email": /@gmail/, "nama": /Muhammad/ };
```

Penggunaan OR

Sebagai contoh kita ingin menampilkan nama orang dengan umur dibawah 35 tahun dan nama depannya adalah “j”.

OrUmur.js

```
1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true }, function(err,
   , db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihan");
7.   var query = { umur: { $lt: 35 }, $or: [{ "nama": /j/ }] };
8.   dbo.collection("author").find(query).toArray(function(err, re
sult) {
9.     if (err) throw err;
10.    console.log(result);
11.    db.close();
12.  });
13.});
```

Penjelasan Code:

- Baris 7 kita definisikan kondisi \$lt yaitu lower than 35, atau (\$or) nama depannya adalah “j”.
- Maka akan menghasilkan nama orang dimana berumur dibawah 35 tahun dan nama depannya “j”.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ node QueryReg
[ { _id: 5be528829929bb32d6bc5ab2, nama: 'joko', umur: 30 },
  { _id: 5be528b39929bb32d6bc5ab4, nama: 'jono', umur: 25 } ]
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ █
```

Gambar 58. Menggunakan OR dalam Query

H. MongoDB Sort

Sorting atau pengurutan data documents, misalkan kita ingin urutkan dari abjad, nomer dan sebagainya. Atau mungkin kalau di MySQL itu Ascending atau Descending.

SortAscending.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true }, function(err
   , db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var defsort = { _id: 1 };
8.   dbo.collection("author").find().sort(defsort).toArray(function
   (err, result) {
9.     if (err) throw err;
10.    console.log(result);
11.    db.close();
12.  });
13. });

```

Penjelasan Code:

- Baris 7 kita definisikan kondisi sorting ascending dimana `_id:1`
- Jika hendak dijadikan descending dapat mengubahnya menjadi `_id:-1`

I. MongoDB Delete

MongoDB memiliki perintah juga untuk menghapus isi document dari sebuah collections, tidak jauh berbeda dengan perintah delete pada MySQL. Misalkan kita hendak menghapus nama dengan `_id` tertentu misalnya, MongoDB sudah menyediakannya. Kita bisa menggunakan `DeleteOne` atau `DeleteMany` (menghapus banyak document).

DeleteOne.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";

```

```

3.
4. MongoClient.connect(url,{ useNewUrlParser: true }, function(e
rr, db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var myquery = { nama: 'joko' };
8.   dbo.collection("author").deleteOne(myquery, function(err, o
bj) {
9.     if (err) throw err;
10.    console.log("1 document deleted");
11.    db.close();
12.  });
13.});

```

Penjelasan Code:

- Pada baris 7 kita definisikan sebuah nama yang hendak dihapus
- Baris 8 panggil fungsi deleteOne dari query yang sudah ditentukan.
- Jika dijalankan, maka document dari collection author dengan nama “joko” akan terhapus.

DeleteMany.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url,{ useNewUrlParser: true }, function(e
rr, db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var myquery = { "email": '@gmail/ ' };
8.   dbo.collection("author").deleteMany(myquery, function(err,
obj) {
9.     if (err) throw err;
10.    console.log(obj.result.n + " document(s) deleted");
11.    db.close();
12.  });
13.});

```

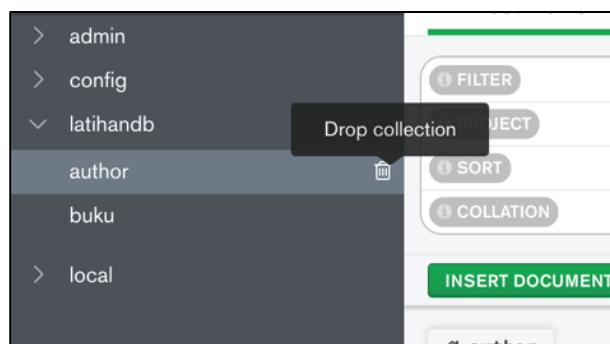
Penjelasan Code:

- Pada baris 7 kita definisikan sebuah query yang hendak dihapus yaitu semua document dengan email berekstensi @gmail
- Baris 8 panggil fungsi deleteMany dari query yang sudah ditentukan.

- Jika dijalankan, maka document dari collection author dengan email berkestensi @gmail akan terhapus.

J. MongoDB Drop Collection

Untuk menghapus collection dalam MongoDB kita bisa menggunakan MongoDB Compass, anda tinggal memilih nama collection yang hendak dihapus, kemudian pilih OK.



Gambar 59. Menghapus Collection

K. MongoDB Update

Seperti halnya pada MySQL ketika mengupdate data, kita juga bisa menerapkannya pada MongoDB, proses update data document bisa digunakan untuk satu query atau beberapa kondisi. Update pada MongoDB tersedia Update One dan Update Many.

UpdateOne.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://127.0.0.1:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true },function(e
rr, db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var myquery = { email: "try@teri.com" };
8.   var newvalues = { $set: {nama: "Tri
Setyo", email: "try@banyustudio.com" } };
9.   dbo.collection("author").updateOne(myquery, newvalues, func
tion(err, res) {
10.     if (err) throw err;
  
```

```

11.     console.log("1 document updated");
12.     db.close();
13.   });
14. });

```

Penjelasan Code:

- Baris 7 buat sebuah variabel query yang hendak diupdate
- Baris 8 buat nilai baru yang akan menggantikan query yang akan diubah
- Baris 9 tentukan nama collection yang dokumentnya hendak diubah.

UpdateMany.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://127.0.0.1:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true },function(e
rr, db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   var myquery = { nama: /^J/ };
8.   var newvalues = {$set: {umur: 26} };
9.   dbo.collection("author").updateMany(myquery, newvalues, fun
ction(err, res) {
10.    if (err) throw err;
11.    console.log(res.result.nModified + " document(s)
updated");
12.    db.close();
13.  });
14. });

```

Penjelasan Code:

- Baris 7 tentukan query yang akan diubah, sebagai contoh adalah semua nama yang berawalan huruf “J”
- Baris 8 ubah nama yang berawalan “J” umurnya menjadi 26 tahun
- Baris 9 tentukan nama collection yang hendak diubah datanya sekaligus menggunakan updateMany().

"andi"	"andi@yahoo.com"	2015-07-02 07:00:00.000	20
"Johan"	"johan@gmail.com"	2015-07-02 07:00:00.000	26
"Jaka"	"jaka@yahoo.com"	2015-07-02 07:00:00.000	26

Gambar 60. Mengubah banyak data sekaligus

L. MongoDB Limit

Limit adalah batas yang ditentukan pada setiap hasil seleksi sebuah documents collection. Sebagai contoh ketika kita hanya ingin menampilkan 5 data teratas saja dalam sebuah hasil pencarian data.

Limit.js

```

1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://localhost:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true },function(e
rr, db) {
5.   if (err) throw err;
6.   var dbo = db.db("latihandb");
7.   dbo.collection("author").find().limit(3).toArray(function(e
rr, result) {
8.     if (err) throw err;
9.     console.log(result);
10.    db.close();
11.  });
12.});
```

Penjelasan Code:

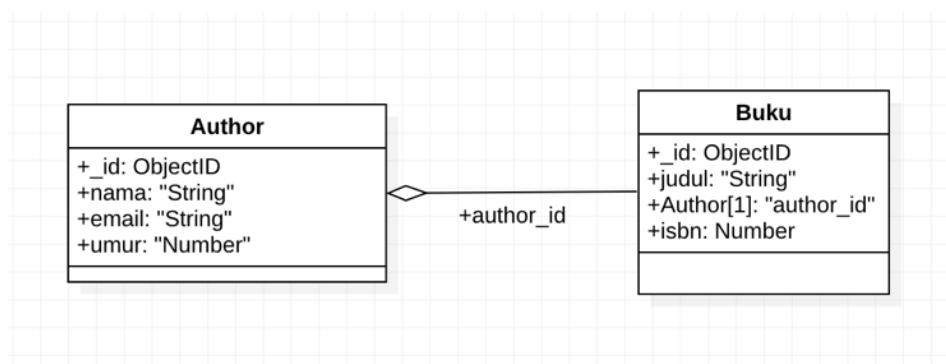
- Baris 7 kita menggunakan limit(3) yang artinya kita hanya akan menampilkan 3 data dari collection “author”

```
[ document(s) updated
ipungs-MacBook-Pro-56:Latihan ipungpurwono$ node Limit
[ { _id: 5be309e39929bb32d6bc5ab0,
  nama: 'Keenar A P',
  email: 'keenarAzPr12@yahoo.com',
  tanggal_lahir: 2015-07-02T00:00:00.000Z },
  { _id: 5be31639049ef63ef04445df,
  nama: 'Tri Setyo',
  email: 'try@banyustudio.com',
  tanggal_lahir: 2015-07-02T00:00:00.000Z },
  { _id: 5be31639049ef63ef04445e0,
  nama: 'Fajar Pamilu',
  email: 'gojeng@yahoo.com',
  tanggal_lahir: 2015-07-02T00:00:00.000Z } ]
```

Gambar 61. Menampilkan data dengan fungsi limit

M. MongoDB Join

Join atau bisa diartikan sebagai gabungan 2 atau lebih komponen membentuk suatu kesatuan lain. Kalau didalam database hal ini sangat penting sekali digunakan, sebagai contoh dalam MySQL anda memiliki tabel desa dan tabel kecamatan, setiap desa harus berelasi dengan id_kecamatan. Jika dilakukan join antara dua tabel tersebut maka anda mampu menghasilkan gabungan data baru. Hal serupa bisa dilakukan di MongoDB, jika di MySQL harus menggunakan foreign_key kemudian dilakukan relasi antar tabel ketika hendak melakukan joins, maka di dalam MongoDB tidak perlu, hanya cukup memanggil field yang dijadikan foreign field. Sebagai contoh kita memiliki dua buah collection yaitu buku dan author. Setiap buku memiliki author_id sendiri yang bisa kita jadikan alat penanda _id author pada collection author.



Gambar 62. Join Dua Collection

```

_id: ObjectId("5be5a0029929bb32d6bc5ab6")
judul: "Proyek Node Js MongoDB"
author_id: ObjectId("5be309e39929bb32d6bc5ab0")
isbn: "1234567890"

_id: ObjectId("5be5a0869929bb32d6bc5ab7")
judul: "Proyek Android Arduino"
author_id: ObjectId("5be309e39929bb32d6bc5ab0")
isbn: "12121334343434343"
  
```

Gambar 63. Collection Buku

<pre>_id: ObjectId("5be309e39929bb32d6bc5ab0") nama: "Keenar A P" email: "keenarAzPr12@yahoo.com" tanggal_lahir: 2015-07-02 00:00:00.000</pre>
<pre>_id: ObjectId("5be31639049ef63ef04445df") nama: "Tri Setyo" email: "try@banyustudio.com" tanggal_lahir: 2015-07-02 00:00:00.000</pre>
<pre>_id: ObjectId("5be31639049ef63ef04445e0") nama: "Fajar Pamilu" email: "gojeng@yahoo.com" tanggal_lahir: 2015-07-02 00:00:00.000</pre>

Gambar 64. Collection Author

Pada gambar di atas terdapat dua buah collection yang melakukan join, dimana collection Buku memiliki column Author yang berisi **author_id** dimana akan dijadikan foreignfield terhadap **localfield _id** miliki collection Author. Maka jika kita gunakan Node.js sebagai alat join sebagai berikut:

JoinMongo.js

```
1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://127.0.0.1:27017/";
3.
4. MongoClient.connect(url, { useNewUrlParser: true },function(err,
5.   , db) {
6.   if (err) throw err;
7.   var dbo = db.db("latihandb");
8.   dbo.collection('author').aggregate([
9.     { $lookup:
10.       {
11.         from: 'buku',
12.         localField: '_id',
13.         foreignField: 'author_id',
14.         as: 'details'
15.       }
16.     }
17.   ]).toArray(function(err, res) {
18.     if (err) throw err;
19.     console.log(JSON.stringify(res));
20.   }
21. );
```

```

19.      db.close();
20.  });
21. });

```

Penjelasan Code:

- Baris 7 kita tentukan terlebih dahulu collection yang menjadi forignField dari collection lain, yang pada kasus ini buku menggunakan author_id sebagai localField pada _id collection author.
- Baris 8 gunakan fungsi lookup
- Baris 10 nama collection yang hendak di join
- Baris 11 nama localField yang digunakan sebagai penghubung dengan forignField.
- Baris 12 nama foreignField yaitu _id milik author
- Baris 13 buat nama column aliasnya.
- Baris 18 cetak dalam log

```

[
  {
    "_id": "5be309e39929bb32d6bc5ab0",
    "nama": "Keenar A P",
    "email": "keenarAzPr12@yahoo.com",
    "tanggal_lahir": "2015-07-02T00:00:00.000Z",
    "details": [
      {
        "_id": "5be5a0029929bb32d6bc5ab6",
        "judul": "Proyek Node Js MongoDB",
        "author_id": "5be309e39929bb32d6bc5ab0",
        "isbn": "1234567890"
      },
      {
        "_id": "5be5a0869929bb32d6bc5ab7",
        "judul": "Proyek Android Arduino",
        "author_id": "5be309e39929bb32d6bc5ab0",
        "isbn": "12121334343434343"
      }
    ]
  },
]

```

Gambar 65. JSON hasil Join \$lookup

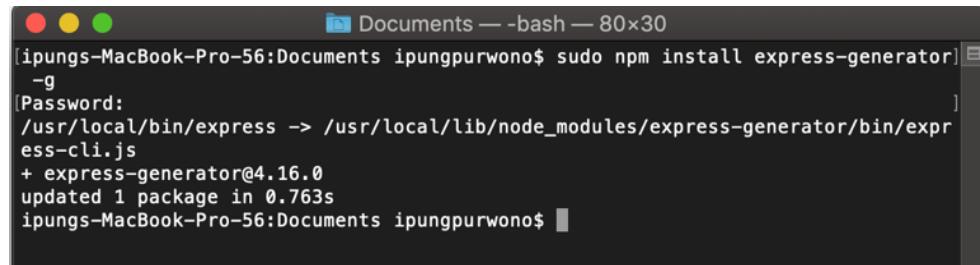
BAB III

Express.Js & Mongoose

Pada bab ini kita akan mengenal sebuah framework yang cukup populer dikalangan pecinta Node.js yaitu Express. Express adalah kerangka aplikasi web Node.js yang minimalist dan fleksibel yang menyediakan serangkaian fitur canggih untuk aplikasi web dan seluler. Dengan Express kita akan diberikan kerangka web yang sangat membantu dalam mengembangkan project web Node.js. Agar lebih memahami penggunaan framework ini kita akan membuat Project sederhana yaitu membuat **CRUD(Create Read Update Delete) Operation** menggunakan Express dan Mongoose. Mongoose memudahkan kita dalam melakukan pemodelan database MongoDB. Sebagai contoh untuk Project Pertama kita adalah membuat sebuah operasi CRUD untuk membuat sebuah pendataan warga berdasarkan desanya. Jadi kita akan membuat dua buah collection yaitu warga dan desa. Dimana dengan bantuan express kita akan mudah membuat aplikasi CRUDnya.

A. Instalasi Express

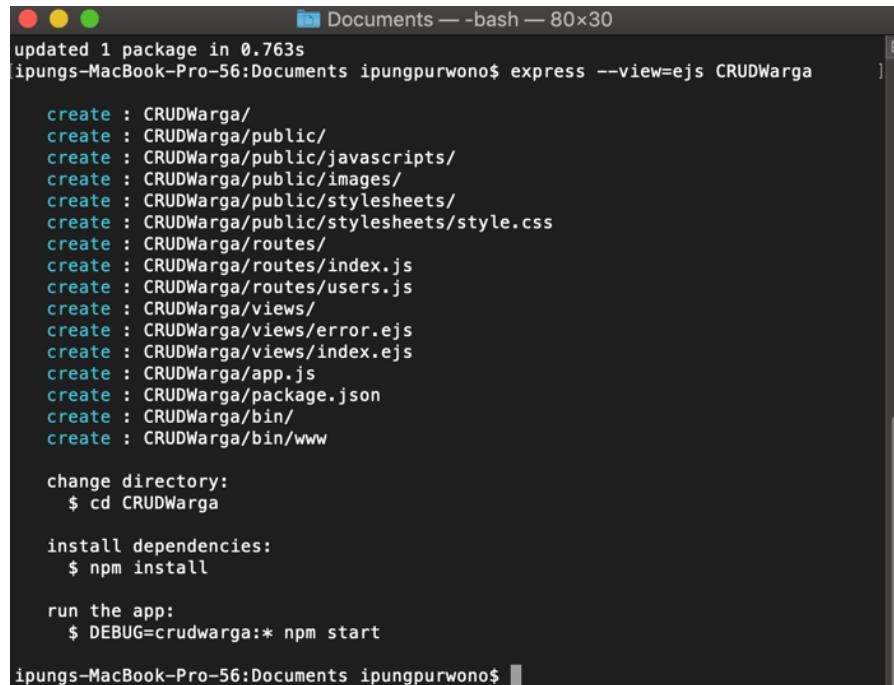
Kita bisa menggunakan Express generator untuk menciptakan kerangka project web seacara cepat. Terlebih dahulu tentukan lokasi dimana anda akan menyimpan file project. Sebagai contoh kami menggunakan Documents sebagai tempat dimana kita akan menginstal generator Express. Arahkan posisi terminal ke Documents dengan mengetikkan perintah “**cd Documents**” pada terminal atau command prompt anda. Anda bebas menaruh project bisa dihardisk D / E disesuaikan dengan kebutuhan anda. Setelah posisi terminal sudah masuk pada Documents, sekarang gunakan perintah “**npm install express-generator -g**” lalu tekan Enter. Jika anda pengguna Mac, gunakan tambahan perintah “**sudo**” didepannya. Hal itu bertujuan agar installasi langsung dilakukan oleh administrator. Karena jika tidak menggunakan sudo maka dipastikan bahwa Express akan error diinstall, jadi gunakan “**sudo**”.



```
[ipungs-MacBook-Pro-56:Documents ipungpurwono$ sudo npm install express-generator]
-g
[Password:
/usr/local/bin/express -> /usr/local/lib/node_modules/express-generator/bin/expr
ess-cli.js
+ express-generator@4.16.0
updated 1 package in 0.763s
ipungs-MacBook-Pro-56:Documents ipungpurwono$ ]
```

Gambar 66. Install Express Generator

Setelah berhasil menginstalnya kita akan membuat kerangka web dengan Express dengan mengetikan perintah “**express --view=ejs CRUDWarga**” kemudian tekan enter.



```
updated 1 package in 0.763s
[ipungs-MacBook-Pro-56:Documents ipungpurwono$ express --view=ejs CRUDWarga
create : CRUDWarga/
create : CRUDWarga/public/
create : CRUDWarga/public/javascripts/
create : CRUDWarga/public/images/
create : CRUDWarga/public/stylesheets/
create : CRUDWarga/public/stylesheets/style.css
create : CRUDWarga/routes/
create : CRUDWarga/routes/index.js
create : CRUDWarga/routes/users.js
create : CRUDWarga/views/
create : CRUDWarga/views/error.ejs
create : CRUDWarga/views/index.ejs
create : CRUDWarga/app.js
create : CRUDWarga/package.json
create : CRUDWarga/bin/
create : CRUDWarga/bin/www

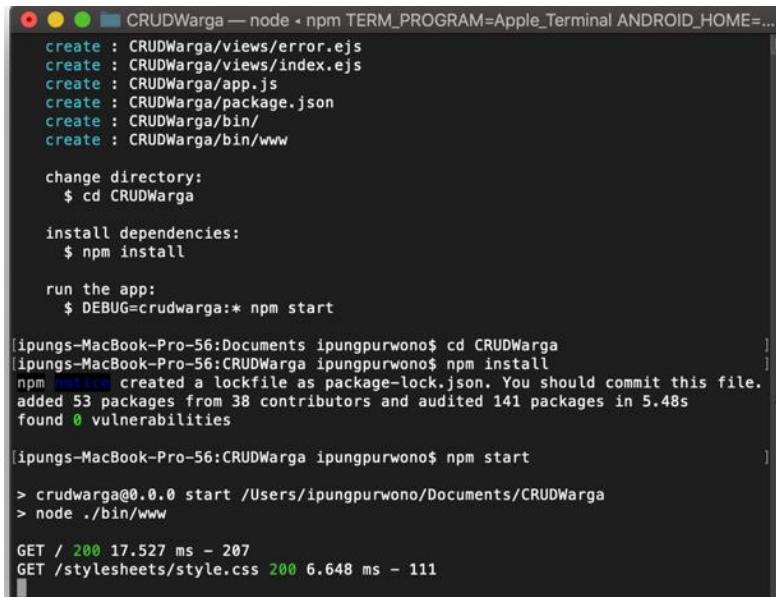
change directory:
$ cd CRUDWarga

install dependencies:
$ npm install

run the app:
$ DEBUG=crudwarga:* npm start
ipungs-MacBook-Pro-56:Documents ipungpurwono$ ]
```

Gambar 67. Membuat kerangka aplikasi CRUDWarga

Pada perintahnya kita diminta melakukan change directory (cd) CRUDWarga artinya kita diminta mengarahkan ke folder tersebut. Setelah berpindah ke folder CRUDWarga, lalu jalankan perintah “**npm install**” kemudian “**npm start**”.



```

CRUDWarga — node -e npm TERM_PROGRAM=Apple_Terminal ANDROID_HOME=...
create : CRUDWarga/views/error.ejs
create : CRUDWarga/views/index.ejs
create : CRUDWarga/app.js
create : CRUDWarga/package.json
create : CRUDWarga/bin/
create : CRUDWarga/bin/www

change directory:
$ cd CRUDWarga

install dependencies:
$ npm install

run the app:
$ DEBUG=crudwarga:* npm start

[ipungs-MacBook-Pro-56:Documents ipungpurwono$ cd CRUDWarga
[ipungs-MacBook-Pro-56:CRUDWarga ipungpurwono$ npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
added 53 packages from 38 contributors and audited 141 packages in 5.48s
found 0 vulnerabilities

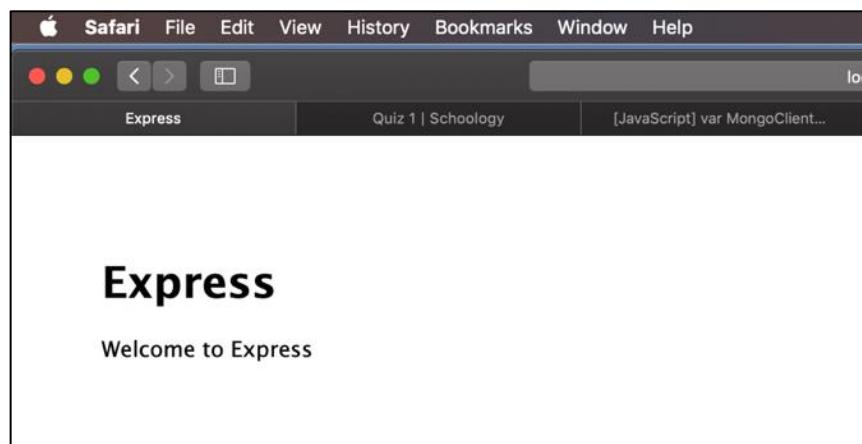
[ipungs-MacBook-Pro-56:CRUDWarga ipungpurwono$ npm start
> crudwarga@0.0.0 start /Users/ipungpurwono/Documents/CRUDWarga
> node ./bin/www

GET / 200 17.527 ms - 207
GET /stylesheets/style.css 200 6.648 ms - 111

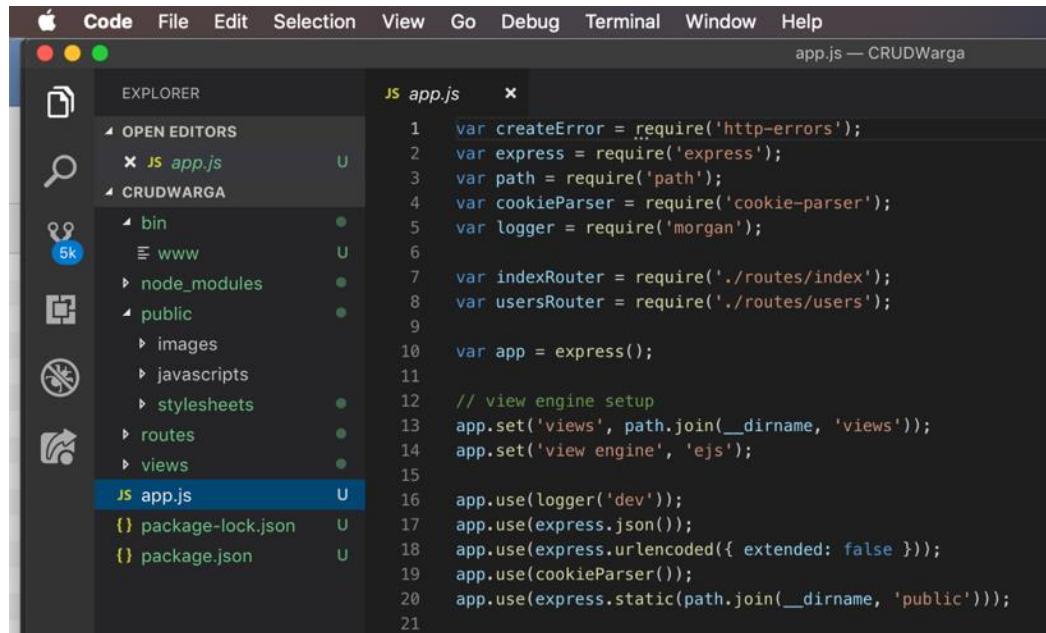
```

Gambar 68. Menjalankan aplikasi CRUDWarga

Jika anda melakukannya dengan benar, maka anda bisa melihat di browser anda pada url <http://localhost:3000> atau <http://127.0.0.1:3000>.

**Gambar 69.** Tampilan Express Pertama Kali

Anda sekarang sudah memiliki sebuah struktur project web CRUDWarga. Langkah selanjutnya adalah Tambahkan project tersebut ke Visual Studio Code Editor. Anda bisa menambahkan folder CRUDWarga dengan cara Add Project to Workspace.



The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "CRUDWARGA". The "bin" folder contains "www". The "public" folder contains "images", "javascripts", "stylesheets", "routes", and "views". The "app.js" file is selected.
- Code Editor (Right):** Displays the content of "app.js". The code initializes an Express application, sets up view engine setup, and uses various middleware and static file handling.

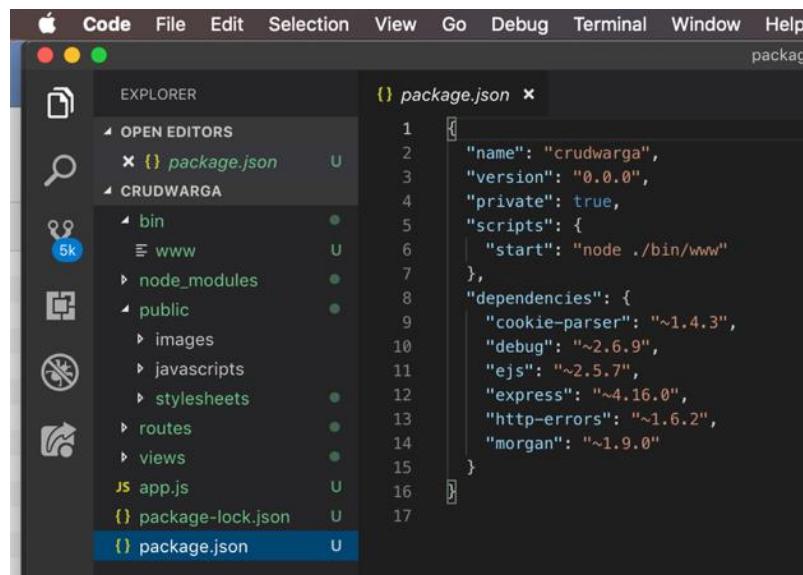
```

1 var createError = require('http-errors');
2 var express = require('express');
3 var path = require('path');
4 var cookieParser = require('cookie-parser');
5 var logger = require('morgan');
6
7 var indexRouter = require('./routes/index');
8 var usersRouter = require('./routes/users');
9
10 var app = express();
11
12 // view engine setup
13 app.set('views', path.join(__dirname, 'views'));
14 app.set('view engine', 'ejs');
15
16 app.use(logger('dev'));
17 app.use(express.json());
18 app.use(express.urlencoded({ extended: false }));
19 app.use(cookieParser());
20 app.use(express.static(path.join(__dirname, 'public')));
21

```

Gambar 70. Struktur Project CRUD Warga

Bisa terlihat pada gambar, anda sudah memiliki struktur project, jika anda membuka app.js maka beberapa packages secara otomatis langsung panggil. Anda bisa melihat berbagai package yang terpasang pada project anda dengan membuka file package.json



The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "CRUDWARGA". The "bin" folder contains "www". The "public" folder contains "images", "javascripts", "stylesheets", "routes", and "views". The "app.js" and "package.json" files are selected.
- Code Editor (Right):** Displays the content of "package.json". It defines the project name, version, private status, scripts (start command), and dependencies (including express, http-errors, morgan, cookie-parser, debug, and ejs).

```

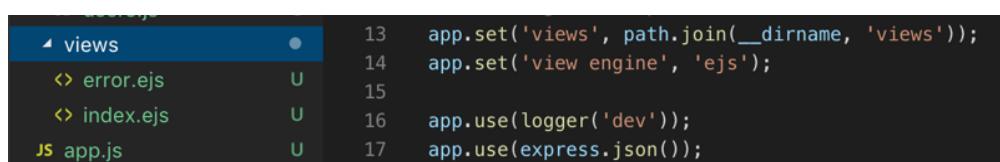
1 {
2   "name": "crudwarga",
3   "version": "0.0.0",
4   "private": true,
5   "scripts": {
6     "start": "node ./bin/www"
7   },
8   "dependencies": {
9     "cookie-parser": "~1.4.3",
10    "debug": "~2.6.9",
11    "ejs": "~2.5.7",
12    "express": "~4.16.0",
13    "http-errors": "~1.6.2",
14    "morgan": "~1.9.0"
15  }
16 }
17

```

Gambar 71. Package.json

B. View Engine

Di dalam project CRUD Warga ini kita menggunakan view engine “ejs” dimana hal tersebut bisa dijadikan alat mendesain tampilan webnya. Ejs dipilih karena struktur codenya yang familiar seperti halnya html. Sebetulnya Express bisa menggunakan alternative view engine seperti pug ataupun jade, namun penulis lebih memilih ejs atas pertimbangan memudahkan pemahaman terutama bagi yang baru proses migrasi dari php ataupun framework seperti codeigniter. Semua penulisan file template berextensi dot ejs dan ditempatkan pada folder views.



```

13  app.set('views', path.join(__dirname, 'views'));
14  app.set('view engine', 'ejs');
15
16  app.use(logger('dev'));
17  app.use(express.json());

```

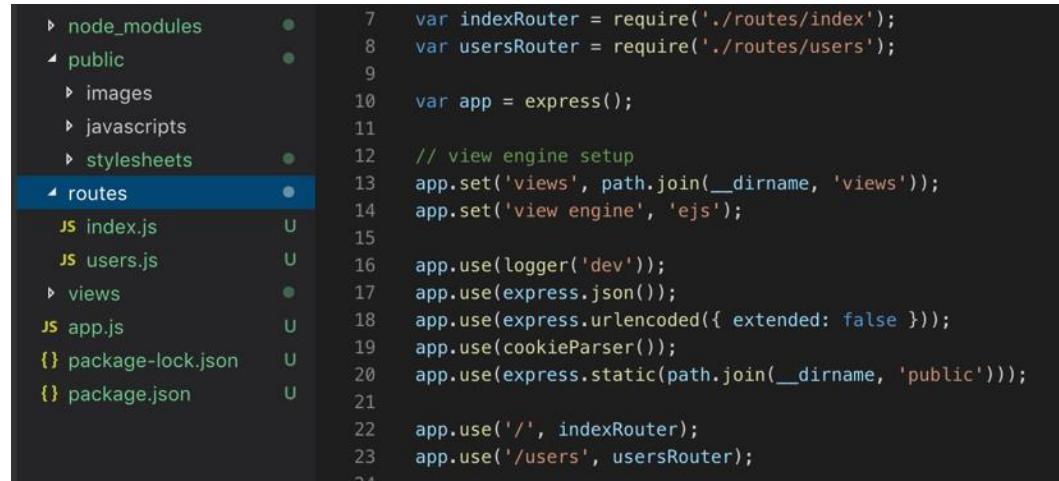
Gambar 72. View Engine ejs

Dapat dilihat pada code baris 13 pada file app.js yang sudah otomatis digenerate oleh Express generator pas installasi, terdapat fungsi bahwa app tersebut menge set views yang akan digunakan untuk templating yang menggunakan nama direktori “views”. Dan pada baris 14 gunakan view engine “ejs”. Nah semua file yang berbentuk tampilan yang mengarah frontend misalnya seperti form login, form registrasi, form tambah barang dan sebagainya akan ditempatkan di folder views.

C. Routing Express

Routing mengacu pada penentuan bagaimana aplikasi merespons permintaan klien ke titik akhir tertentu, yang merupakan URI (atau jalur) dan metode permintaan HTTP spesifik (GET, POST, dan sebagainya). Routing bahsa sederhananya adalah alamat link yang akan dibuat seperti `http://localhost:3000/about`, nah **about** tersebut adalah routing yang bisa dibuat dengan mudah dengan Express. Setiap rute dapat memiliki satu atau lebih fungsi handler, yang dieksekusi ketika rute dicocokkan. Express sudah menyediakannya secara langsung untuk memudahkan kita ketika membuat sebuah

routing oleh karenanya, kita bisa melihat sebuah folder khusus dengan nama “routes” pada struktur project anda.



```

    ▶ node_modules      ●  7  var indexRouter = require('./routes/index');
    ▶ public           ●  8  var usersRouter = require('./routes/users');
    ▶   images          9
    ▶   javascripts    10 var app = express();
    ▶   stylesheets     11
    ▷ routes          ● 12 // view engine setup
    JS index.js        U 13 app.set('views', path.join(__dirname, 'views'));
    JS users.js        U 14 app.set('view engine', 'ejs');
    ▶ views            ● 15 app.use(logger('dev'));
    JS app.js          U 16 app.use(express.json());
    {} package-lock.json U 17 app.use(express.urlencoded({ extended: false }));
    {} package.json    U 18 app.use(cookieParser());
    20 app.use(express.static(path.join(__dirname, 'public')));
    21
    22 app.use('/', indexRouter);
    23 app.use('/users', usersRouter);
    24
  
```

Gambar 73. Routes Express

Bisa anda lihat bahwa pada struktur project terdapat nama folder routes, disana kita bisa membuat berbagai routes yang didalamnya bisa dimasukan berbagai macam fungsi termasuk memanipulasi database MongoDB. Bisa lihat pada baris 7 dan 8 adalah membuat variable untuk mengambil file routes yang sudah dibuat dan kemudian diexport sebagai sebuah modul. Jika masih belum paham modul apa anda bisa buka pada bab pertama. Setelah routes dibuat kemudian gunakan routes tersebut sebagai alamat route di aplikasi web anda. Bisa menggunakan “use” lalu nama routenya. Anda bisa cek pada baris ke 22 dan 23 pada file app.js.

D. Static Folder

Static folder bisa kita gunakan untuk membantu dalam proses templating misalkan kita hendak menggunakan framework Bootstrap sebagai template webnya atau bisa juga menggunakan template yang sudah ada. Anda bisa memasukannya pada folder static ini, disini Express sudah langsung generate nama foldernya yaitu “public”.

```

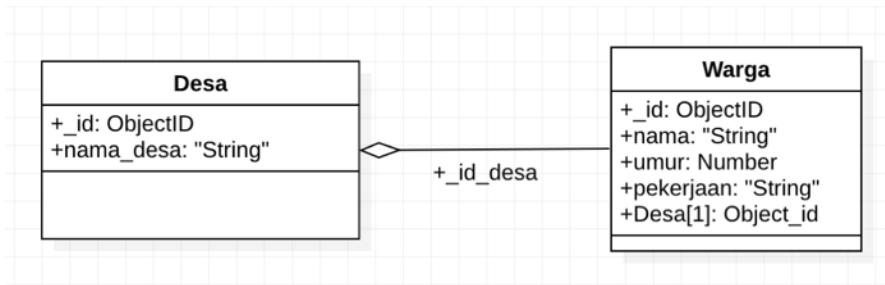
  ↗ public          • 19 app.use(cookieParser());
  ↗ images          20 app.use(express.static(path.join(__dirname, 'public')));
  ↗ javascripts    21
  ↗ stylesheets     22 app.use('/', indexRouter);
  ↗ routes          23 app.use('/users', usersRouter);
  ↗ routes          24
  ↗ routes          25 // catch 404 and forward to error handler

```

Gambar 74. Static Files

E. Membuat Project CRUD Warga

Tahap installasi Express sudah dilakukan, selanjutnya kita akan merancang aplikasi web sederhana yang didalamnya terdapat operasi CRUD dalam pengolahan data warga desa, jadi gambaran class diagramnya adalah sebagai berikut:

**Gambar 75.** Class Diagram

Kita bisa melihat pada class diagram dimana, collection Warga akan menggunakan id_desa sebagai localField yang akan dijadikan foreignField dari _id collection Desa.

1. Koneksi Express dan MongoDB

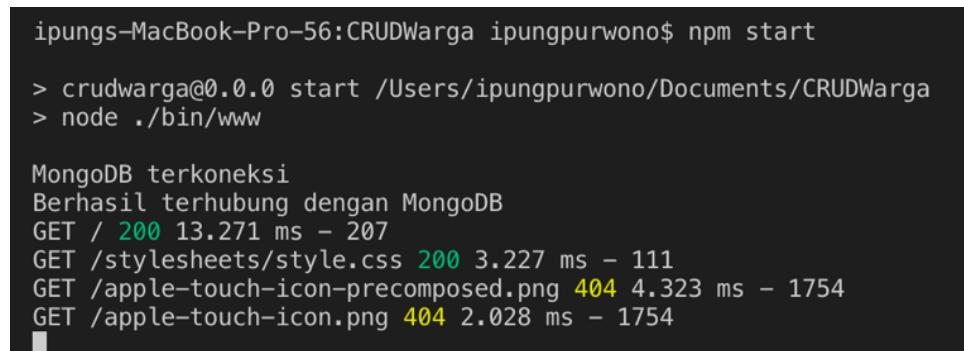
Agar aplikasi yang kita bangun dapat berjalan dengan baik, kita perlu untuk mengkoneksikannya. Jika anda sudah membaca pada bab kedua, sudah kami sertakan bagaimana cara untuk melakukan koneksi antara Node.js dan MongoDB. Caranya sebetulnya sama, namun karena ini pada framework jadi anda cukup mendefinisikan konesinya pada app.js. Sebelum membuat code konesninya kita terlebih dahulu menginstall mongoose. Mongoose adalah sebuah package atau modul yang dapat mempermudah kita dalam membuat skema database mongodb. Pada terminal visual studio code anda ketikan perintah “**npm install mongoose**”. Tunggu hingga proses installasi selesai. Setelah terinstall kita akan

menambahkan beberapa baris code pada file app.js sebagai berikut. Agar tidak salah pastikan anda sudah membuat database baru menggunakan MongoDB Compass dengan nama “**cruddb**”.

```

1. //koneksi ke mongodb
2. var mongoose = require('mongoose');
3. mongoose.Promise = global.Promise;
4.
5. mongoose.connect('mongodb://localhost/cruddb', { useNewUrlParser: true })
6.   .then(() => console.log('Berhasil terhubung dengan
    MongoDB'))
7.   .catch((err) => console.error(err));
8.
9. console.log('MongoDB terkoneksi');
```

Tambahkan 8 baris code tersebut yang anda masukan pada file **app.js**. Code tersebut akan memanggil modul mongoose yang sebelumnya sudah kita install, lalu gunakan Promise agar kita bisa mengoneksikannya dengan nama database **cruddb**. Sekarang pada terminal jalankan perintah “**npm start**”.



```

ipungs-MacBook-Pro-56:CRUDWarga ipungpurwono$ npm start
> crudwarga@0.0.0 start /Users/ipungpurwono/Documents/CRUDWarga
> node ./bin/www

MongoDB terkoneksi
Berhasil terhubung dengan MongoDB
GET / 200 13.271 ms - 207
GET /stylesheets/style.css 200 3.227 ms - 111
GET /apple-touch-icon-precomposed.png 404 4.323 ms - 1754
GET /apple-touch-icon.png 404 2.028 ms - 1754
```

Gambar 76. Menghubungkan ke MongoDB

2. Membuat Models

Kita sudah menghubungkan Node.js dengan MongoDB menggunakan modul mongoose. Selanjutnya kita akan membuat folder baru yaitu **models**. Pastikan anda sudah memiliki folder dengan nama “**models**”. Pada folder models kita akan membuat schema database MongoDB.

**Gambar 77.** Membuat folder models

Pada folder models kita buat dua buah files yaitu Desa.js dan Warga.js. Kedua file tersebut akan menciptakan sebuah scheme MongoDB yang bias dipanggil pada modul lain. Ketikan code seperti berikut :

Desa.js

```

1. var mongoose = require('mongoose');
2. var Schema = mongoose.Schema;
3. var DesaSchema = new Schema(
4.   {
5.     nama: {type: String, required: true}
6.   }
7. );
8.
9. //Export model
10.module.exports = mongoose.model('Desa', DesaSchema);

```

Penjelasan Code :

- Pada baris ke 1 kita panggil modul mongoose
- Baris 2 kita buat sebuah schema, schema ini fungsi bawaan dari mongoose
- Baris 3 buat sebuah variable schema dengan nama DesaSchema dimana akan menyimpan document nama. Field Document nama tersebut bertipe String, dan wajib diinput.
- Baris 10 lakukan export agar dikenali sebagai sebuah modul sehingga bisa dijadikan object yang bisa dipanggil oleh file lain.

Warga.js

```

1. var mongoose = require('mongoose');
2. var Schema = mongoose.Schema;
3. var WargaSchema = new Schema(
4.   {
5.     nama: { type: String, required: true },
6.     umur: { type: String, required: true },
7.     pekerjaan: { type: String, required: true },
8.     desa_id: { type: Schema.Types.ObjectId, ref: 'Desa', required: true }
9.   }
10.);
11.
12. //Export model
13.module.exports = mongoose.model('Warga', WargaSchema);

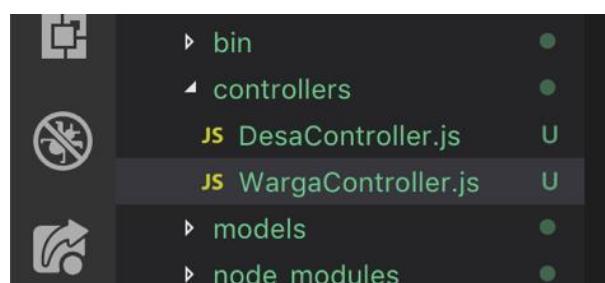
```

Penjelasan Code :

- Konsepnya sama dengan models Desa.js
- Baris 3 kita buat schema baru yaitu WargaSchema
- Baris 8 adalah sebuah id yang mewakili nama desa yang akan kita gunakan saat menginputkan warga baru untuk nama desanya.

3. Membuat Controllers

Folder models sudah kita buat, selanjutnya adalah folder controllers, folder ini akan kita gunakan untuk mengelompokan file yang akan melakukan control terhadap berbagai bentuk pengolahan data. Setelah anda membuat folder ‘controllers’ selanjutnya adalah kita akan membuat dua buah files yaitu **DesaController.js** dan **WargaController.js**.



Gambar 78. Membuat folder controllers

DesaController.js

```

1. var mongoose = require("mongoose");
2. var Desa = require("../models/Desa");
3.
4. var desaController = {};
5.
6. //control menampilkan data dari collection desa yang akan ditampilkan pad form input warga
7. desaController.find = function(req, res) {
8.   var desa = new Desa(req.body);
9.
10.  desa.find({}, function (err, desa) {
11.    console.log(desa);
12.    res.render('warga', {desa: desa ,title: 'CRUD Desa'});
13.  }).select('_id');
14. };
15.
16. // controll untuk melakukan penyimpanan data inputan ke collection desa
17. desaController.save = function(req, res) {
18.   var desa = new Desa(req.body);
19.
20.   desa.save(function(err) {
21.     if(err) {
22.       console.log(err);
23.       res.render('desa');
24.     } else {
25.       console.log("Sukses menyimpan data desa.");
26.       //kembalikan ke halaman home
27.       res.redirect('../');
28.     }
29.   });
30. };
31.
32. module.exports = desaController;

```

Penjelasan Code :

- Baris 1 kita panggil modul mongoose.
- Baris 2 panggil / include Models Desa
- Baris 4 buat sebuah array Controller

- Baris 7 -14 buat sebuah fungsi yang akan menampilkan semua data desa yaitu _id dan nama desa. Hal ini diperlukan saat kita menginput warga desa, yang nota bene list nama desanya pada select box.
- Baris 17 -30 buat sebuah fungsi untuk proses menyimpan data ke database.
- Baris 27 lakukan redirect ke halaman home ketika berhasil terinput
- Baris 32 jadikan sebuah module dengan exports.

WargaController.js

```

1. var mongoose = require("mongoose");
2. var Warga = require("../models/Warga");
3.
4. var wargaController = {};
5.
6. // control untuk melakukan penyimpanan data inputan ke
   collection warga
7. wargaController.save = function(req, res) {
8.   var warga = new Warga(req.body);
9.
10.  warga.save(function(err) {
11.    if(err) {
12.      console.log(err);
13.      res.render('index');
14.    } else {
15.      console.log("Successfully created an warga.");
16.      res.redirect('../');
17.    }
18.  });
19.};
20.
21.module.exports = wargaController;

```

Penjelasan Code :

- Baris 1 kita panggil modul mongoose.
- Baris 2 panggil / include Models Warga
- Baris 3 buat sebuah array Controller
- Baris 7 – 19 buat sebuah fungsi untuk menyimpan data ke database.
- Baris 6 lakukan redirect ke halaman home ketika berhasil terinput
- Baris 21 jadikan sebuah module dengan exports.

4. Membuat Routes

Agar kita bisa mengakses sebuah url atau alamat kita wajib menciptakan routes.

Routes disini akan memanggil dua alamat web berikut yaitu :

- Localhost:3000/warga
- Localhost:3000/desa
- Localhost:3000

Untuk membuat routes harus disimpan pada folder routes yang notabene foldernya sudah auto generate pas kita instalasi Express.js. Sekarang kita buat tiga buah routes yaitu index.js, warga.js dan desa.js.

Index.js

```

1. var express = require('express');
2. var router = express.Router();
3.
4. // GET home page.
5. router.get('/', function(req, res, next) {
6.   res.render('index', { title: 'CRUD Warga' });
7. });
8.
9. module.exports = router;

```

warga.js

```

1. var express = require('express');
2. var router = express.Router();
3. var mongoose = require('mongoose');
4. var warga = require("../controllers/WargaController.js");
5. var Desa = require("../models/Desa");
6.
7. //ambil tampilan render dari folder view berformat ejs
8. router.get('/', function(req, res) {
9.   Desa.find({}, function (err, desa) {
10.     console.log(desa);
11.     res.render('warga', {desa: desa ,title: 'CRUD Desa'});
12.   }).select('_id nama');
13. });
14.
15.//fungsi POST menambah data desa
16. router.post('/tambah', function(req, res) {
17.   warga.save(req, res);
18. });
19.
20. module.exports = router;

```

Penjelasan Code :

- Baris 1 kita panggil modul express.
- Baris 2 aktifkan fungsi router
- Baris 3 panggil modul mongoose
- Baris 4 include kan controller WargaController.js
- Baris 5 include kan models Desa
- Baris 8 – 13 buat sebuah url, hal ini bertujuan untuk menciptakan url localhost:3000/warga. Kemudian di baris 9 lakukan find untuk menyeleksi dan menampilkan semua data pada collection Desa. Setelah terseleksi lalu tampilkan pada console log (Baris 10). Baris 11 lakukan rendering dimana akan memanggil sebuah template / tampilan ejjs yang berfungsi menampilkan isi documents dari collection Desa yaitu _id dan nama ke file views warga yang akan kita buat di folder views.
- Baris 16 – 18 adalah sebuah url yang akan kita gunakan sebagai action pada form input data warga.

desa.js

```

1. var express = require('express');
2. var router = express.Router();
3. var mongoose = require('mongoose');
4. var desa = require("../controllers/DesaController.js");
5.
6. //ambil tampilan render dari folder view berformat ejjs
7. router.get('/', function(req, res, next) {
8.   res.render('desa', { title: 'Desa' });
9. });
10.
11. //fungsi POST menambah data desa
12. router.post('/tambah', function(req, res) {
13.   desa.save(req, res);
14. });
15.
16. module.exports = router;

```

Penjelasan Code :

- Secara konsep sama hanya terjadi perbedaan saat get akan memanggil form views untuk input data desa.

- Pada baris 12 – 14 adalah sebuah proses action yang berfungsi menyimpan data desa. Akan digunakan pada action form input desa.

5. Panggil routes pada app.js

Agar routes dapat dipanggil dan menampilkan sebuah tampilan web yaitu :

- Localhost:3000/warga
- Localhost:3000/desa

Perlu kita daftarkan pada app.js. Tambahkan beberapa baris code seperti berikut pada app.js

App.js

```
1. var indexRouter = require('./routes/index');
2. var desaRouter = require('./routes/desa');
3. var wargaRouter = require('./routes/warga');
```

Ketiga baris code tersebut berfungsi untuk mengincludekan semua router yang kita butuhkan. Tambahkan kembali code berikut pada app.js

```
1. app.use('/', indexRouter);
2. app.use('/desa', desaRouter);
3. app.use('/warga', wargaRouter);
```

Ketiga baris code tersebut berfungsi untuk mendaftarkan sebuah router pada app.js. Kode lengkap app.js adalah sebagai berikut.

app.js

```
1. var createError = require('http-errors');
2. var express = require('express');
3. var path = require('path');
4. var cookieParser = require('cookie-parser');
5. var logger = require('morgan');
6.
7. var indexRouter = require('./routes/index');
8. var desaRouter = require('./routes/desa');
9. var wargaRouter = require('./routes/warga');
10.
```

```

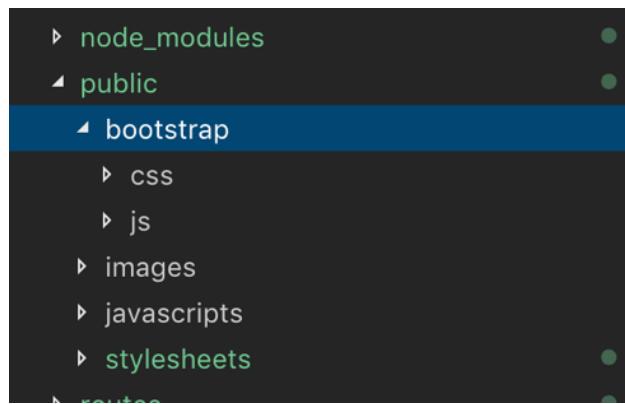
11. var app = express();
12.
13. // view engine setup
14. app.set('views', path.join(__dirname, 'views'));
15. app.set('view engine', 'ejs');
16.
17. app.use(logger('dev'));
18. app.use(express.json());
19. app.use(express.urlencoded({ extended: false }));
20. app.use(cookieParser());
21. app.use(express.static(path.join(__dirname, 'public')));
22.
23. //koneksi ke mongodb
24. var mongoose = require('mongoose');
25. mongoose.Promise = global.Promise;
26.
27. mongoose.connect('mongodb://localhost/cruddb', { useNewUrlParser: true })
28.   .then(() => console.log('Berhasil terhubung dengan
    MongoDB'))
29.   .catch((err) => console.error(err));
30.
31. console.log('MongoDB terkoneksi');
32.
33. app.use('/', indexRouter);
34. app.use('/desa', desaRouter);
35. app.use('/warga', wargaRouter);
36.
37. // catch 404 and forward to error handler
38. app.use(function(req, res, next) {
39.   next(createError(404));
40. });
41.
42. // error handler
43. app.use(function(err, req, res, next) {
44.   // set locals, only providing error in development
45.   res.locals.message = err.message;
46.   res.locals.error = req.app.get('env') === 'development' ? e
      rr : {};
47.
48.   // render the error page
49.   res.status(err.status || 500);
50.   res.render('error');
51. });
52.
53. module.exports = app;

```

6. Membuat View Input Data

Setelah berhasil membuat controller, models dan routes, selanjutnya adalah membuat template aplikasinya kalau dalam keseharian ya membuat template htmlnya. Template html tersebut akan disimpan dalam ekstensi ejs. Dalam hal ini

kita akan berlatih membuat tampilan web sederhana dengan bantuan bootstrap. Bootstrap adalah framework css yang mampu membantu proses pembuatan tampilan web menjadi lebih cepat dan rapih. Bootstrap saat ini menjadi framework css yang sangat terkenal karena dokumentasinya yang lengkap dan mudah dipelajari. Anda bisa mendownload bootstrap pada <https://getbootstrap.com/docs/4.0/getting-started/download/>. Anda cukup mengambil file **bootstrap.min.css** dan **bootstrap.min.js** sebagai alat bantu dalam membangun tampilan web. File yang telah anda download tersebut kemudian tempatkan pada folder public.



Gambar 79. Memindahkan file bootstrap ke static folder public

Selanjutnya kita akan membuat dua buah file ejs , yaitu **index.ejs**, **warga.ejs** dan **desa.ejs** Kedua file tersebut berisi form untuk menambahkan data warga dan desa. Perlu diingat bahwa file berformat .ejs wajib disimpan di folder **views**.

index.ejs

```

1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title><%= title %></title>
5.     <link rel='stylesheet' href='/stylesheets/style.css' />
6.     <link
      rel='stylesheet' href='/bootstrap/css/bootstrap.min.css' />
7.
8.   </head>
9.   <body>
```

```

10.    <h1><%= title %></h1>
11.    <p>Welcome to <%= title %></p>
12.</html>

```

warga.ejs

```

1.  <!DOCTYPE html>
2.  <html>
3.    <head>
4.      <title><%= title %></title>
5.      <link rel='stylesheet' href='/stylesheets/style.css' />
6.      <link
7.        rel='stylesheet' href='/bootstrap/css/bootstrap.min.css' />
8.    </head>
9.    <body>
10.      <h1><%= title %></h1>
11.      <p>Welcome to <%= title %></p>
12.      <form action="warga/tambah" method="post" role="form">
13.        <div class="form-group">
14.          <label for="exampleInputEmail1">Nama</label>
15.          <input type="text" name="nama" class="form-control" id="" required placeholder="Enter nama">
16.        </div>
17.        <div class="form-group">
18.          <label for="exampleInputPassword1">Umur</label>
19.          <input type="number" name="umur" class="form-control" id="" required placeholder="Umur">
20.        </div>
21.        <div class="form-group">
22.          <label for="exampleInputPassword1">Pekerjaan</label>
23.          <input type="text" name="pekerjaan" class="form-control" id="" required placeholder="Pekerjaan">
24.        </div>
25.        <div class="form-group">
26.          <label for="exampleInputPassword1">Desa</label>
27.          <select type="text" name="desa_id" class="form-control" id="" required>
28.            <% for(var i = 0; i < desa.length; i++) {%
29.              <option value="<%= desa[i]._id
30.                %>"><%= desa[i].nama %></option>
31.            <% } %
32.          </select>
33.        </div>
34.        <button type="submit" class="btn btn-primary">Submit</button>
35.      </form>
36.    </body>
37.    <script
38.      type="javascript" src='/booststrap/js/bootstrap.min.js'>
39.</html>

```

Penjelasan Code :

- Baris 11 pada form action kita masukan sebuah url yaitu warga/tambah. Url tersebut adalah routes yang sudah kita buat pada routes warga yang didalamnya terdapat sebuah fungsi tambah.
- Baris 27 kita lakukan looping untuk menampilkan sebuah value dari data collection desa, yang kita sudah set di routes warga. Anda melihat code berikut pada routes warga.

```
//ambil tampilan render dari folder view berformat ejs
router.get('/', function(req, res) {
  Desa.find({}, function (err, desa) {
    console.log(desa);
    res.render('warga', {desa: desa ,title: 'CRUD Desa'});
    }).select('_id nama');
});
```

- Kita menyeleksi _id dan nama dimana akan di tampilkan pada nilai sebuah value dari select box yang terdapat pada baris ke 28.
- Awalnya kita lakukan looping untuk mengambil semua data dari collection desa. Lalu tampilkan nilainya pada baris ke 28 yaitu untuk option value. Dan isi optionnya.



Gambar 80. Menampilkan nilai dari database ke select box

desa.ejs

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title><%= title %></title>
```

```

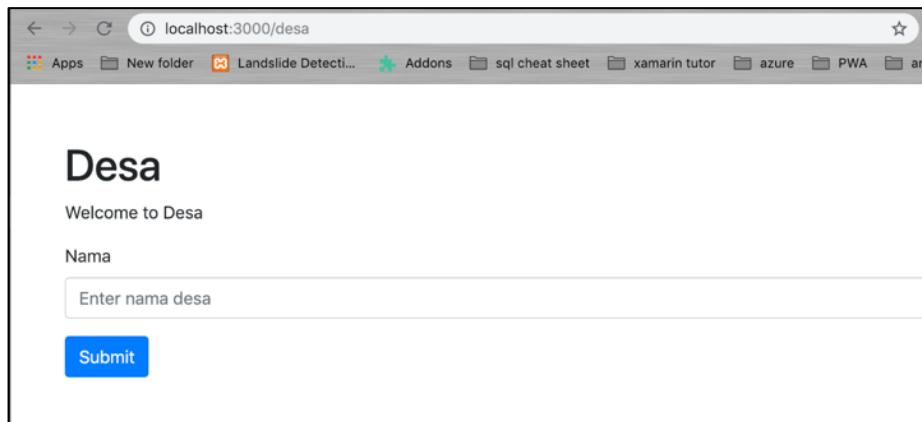
5.      <link rel='stylesheet' href='/stylesheets/style.css' />
6.      <link
7.          rel='stylesheet' href='/bootstrap/css/bootstrap.min.css' />
8.    </head>
9.    <body>
10.      <h1><%= title %></h1>
11.      <p>Welcome to <%= title %></p>
12.      <form action="desa/tambah" method="post" role="form">
13.        <div class="form-group">
14.          <label for="exampleInputEmail1">Nama</label>
15.          <input type="text" name="nama" class="form-
    control" id="" required placeholder="Enter nama desa">
16.        </div>
17.
18.        <button type="submit" class="btn btn-
    primary">Submit</button>
19.      </form>
20.    </body>
21.
22.    <script
23.      tyle="javascript" src='/booststrap/js/bootstrap.min.js'>
23.</html>

```

Penjelasan Code :

- Baris 12 form action kita masukan sebuah url yaitu desa/tambah. Url tersebut adalah routes yang sudah kita buat pada routes warga yang didalamnya terdapat sebuah fungsi tambah.

Beberapa file sudah kita buat sekarang kita akan menjalankan aplikasi websitenya. Untuk menjalankannya kita cukup mengetikan perintah pada terminal yaitu “**npm start**”. Tunggu hingga server Node.js berjalan. Jika sudah berjalan silahkan akses halaman **localhost:3000/desa**. Lakukan input nama desa lalu simpan. Jika anda mengikuti proses pembuatan dengan benar maka data akan tersimpan di database.



Gambar 81. Menampilkan tampilan input desa

Jika anda memasukan sebuah data inputan maka akan masuk ke dalam database MongoDB anda bias cek pada MongoDB Compass pada collection desas.

A screenshot of the MongoDB Compass interface showing the 'cruddb.desas' collection. It displays two documents in 'LIST' view. The first document has '_id: ObjectId("5be69006e83eee58de958dcb")', 'nama: "Tlaga"', and '_v: 0'. The second document has '_id: ObjectId("5be6900ee83eee58de958dcc")', 'nama: "Gumelar"', and '_v: 0'.

Gambar 82. Data desa masuk pada database

Lakukan uji coba juga dengan mengetikan alamat **localhost:3000/warga** maka anda akan dihadapkan pada form input warga.

CRUD Desa

Welcome to CRUD Desa

Nama
Enter nama

Umur
Umur

Pekerjaan
Pekerjaan

Desa
Tlaga

Submit

Gambar 83. Form input warga

Bisa anda lihat pada form input desa, anda bisa memilih beberapa desa yang sudah anda inputkan sebelumnya. Masukan sebuah data maka jika ditekan tombol submit maka data akan masuk pada database.

```
_id: ObjectId("5bf7ff0d63628ee933efb65b")
nama: "Andri"
umur: "23"
pekerjaan: "Programmer"
desa_id: ObjectId("5be69006e83eee58de958dc")
__v: 0
```

Gambar 83. Data warga masuk pada database

Anda bisa melihat pada desa_id kita berhasil memasukan sebuah _id yang akan kita gunakan untuk menampilkan data menyesuaikan dengan _id yang telah ditentukan. Misalkan ingin menampilkan data warga berdasarkan kelompok _id

desa tertentu misalnya. Hal ini bias kita gunakan konsepnya seperti join pada MySQL.

Anda bisa menampilkan data dari semua isi collection warga, kita akan merubah code pada routes index.js dan view index.ejs.

Index.js

```

1. var express = require('express');
2. var router = express.Router();
3. var Warga = require("../models/Warga");
4.
5. // GET home page.
6. router.get('/', function(req, res, next) {
7.   Warga.find({}, function (err, warga) {
8.     console.log(warga);
9.     res.render('index', {warga: warga ,title: 'CRUD Warga'});
10.    }).select('_id nama umur pekerjaan');
11. });
12.
13. module.exports = router;

```

Penjelasan Code :

- Baris ke 3 kita includekan models Warga karena kita hendak menampilkan data Warga.
- Baris 7 kita tampilkan data menggunakan fungsi find.
- Baris 9 tampilkan semua data warga pada index.ejs dimana yang ditampilkan adalah _id nama umur pekerjaan.

Index.ejs

```

1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title><%= title %></title>
5.     <link rel='stylesheet' href='/stylesheets/style.css' />
6.     <link
      rel='stylesheet' href='/bootstrap/css/bootstrap.min.css' />
7.   </head>
8.   <body>
9.     <h1><%= title %></h1>

```

```

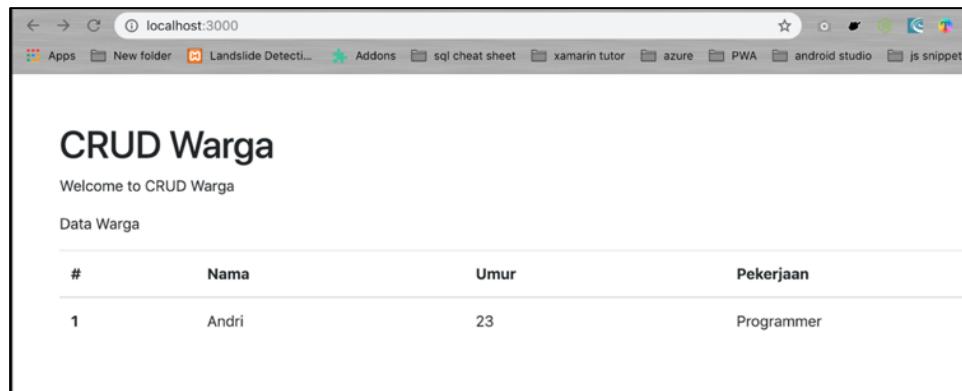
10.    <p>Welcome to <%= title %></p>
11.    <p> Data Warga</p>
12.    <table class="table">
13.      <thead>
14.        <tr>
15.          <th scope="col">#</th>
16.          <th scope="col">Nama</th>
17.          <th scope="col">Umur</th>
18.          <th scope="col">Pekerjaan</th>
19.        </tr>
20.      </thead>
21.      <tbody>
22.      <% var no=1 %>
23.      <% for(var i = 0; i < warga.length; i++) { %>
24.        <tr>
25.          <th scope="row"><%= no++ %></th>
26.          <td><%= warga[i].nama %></td>
27.          <td><%= warga[i].umur %></td>
28.          <td><%= warga[i].pekerjaan %></td>
29.        </tr>
30.      <% } %>
31.      </tbody>
32.    </table>
33.  </html>

```

Penjelasan Code :

- Baris ke 23 kita bisa melihat adanya sebuah proses looping dari data warga yang sudah dirender.
- Baris 26 – 28 tampilkan data tersebut pada table.
- Baris 30 tutup looping.

Sekarang silahkan akses halaman localhost:3000. Jika terjadi error lakukan stop server terlebih dahulu dengan cara **ctrl + c** untuk stop Node.js selanjutnya restart kembali server Node.js tersebut dengan mengetikan perintah “**npm start**”. Silahkan akses halaman “localhost:3000” pada browser anda. Seharusnya anda akan melihat sebuah table dengan data dari collection MongoDB.



Gambar 84. Menampilkan data warga pada tabel

7. Melakukan Edit & Hapus Data

Selain data bisa kita input maka data juga harus bisa kita edit. Konsep edit adalah mengambil sebuah `_id` dengan metode get lalu kita update datanya berdasarkan id tersebut. Agar kita bisa mengedit data kita perlu membuat sebuah tombol / link aksi yaitu edit dan delete. Kami juga menambahkan icon fontawesome yang ditanam di header. Tambahkan code berikut di bagian header pada file **index.ejs** agar kita bisa menambahkan icon pada button / link.

```
<link
rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.css" integrity="sha384-B4diYHKNBt8Bc12p+WXckhzCICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3B1XeVIU"
crossorigin="anonymous">
```

Sebetulnya anda bisa melihat dokumentasi penggunaan font awesome pada link berikut <https://fontawesome.com/start>. Setelah anda memasang font awesome pada header index.ejs maka tambahkan beberapa baris berikut pada **index.ejs** yaitu setelah baris ke 28. Sehingga code index.ejs anda menjadi seperti berikut.

Index.ejs

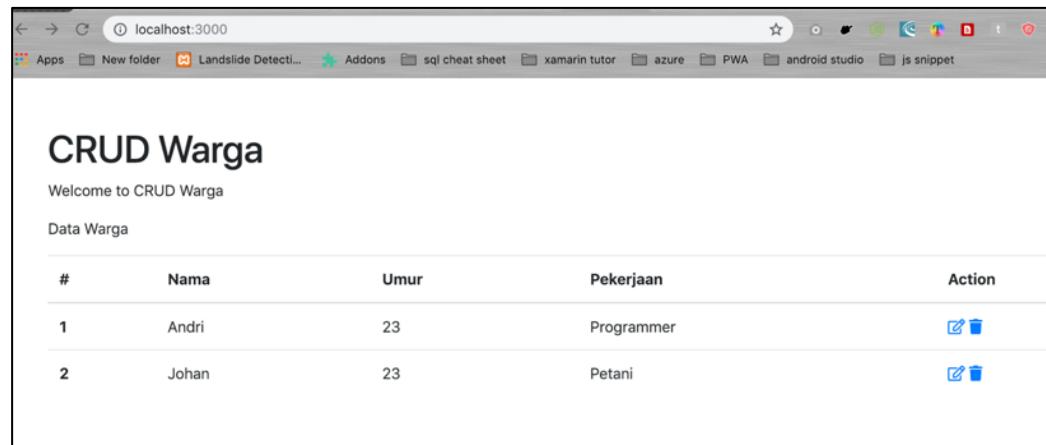
```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title><%= title %></title>
```

```

5.      <link rel='stylesheet' href='/stylesheets/style.css' />
6.  <link
   rel='stylesheet' href='/bootstrap/css/bootstrap.min.css' />
7.  <link
   rel="stylesheet" href="https://use.fontawesome.com/releases/v
5.5.0/css/all.css" integrity="sha384-
B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3B1Xe
VIU" crossorigin="anonymous">
8.  </head>
9.  <body>
10.   <h1><%= title %></h1>
11.   <p>Welcome to <%= title %></p>
12.   <p> Data Warga</p>
13.   <table class="table">
14.     <thead>
15.       <tr>
16.         <th scope="col">#</th>
17.         <th scope="col">Nama</th>
18.         <th scope="col">Umur</th>
19.         <th scope="col">Pekerjaan</th>
20.         <th scope="col">Action</th>
21.       </tr>
22.     </thead>
23.     <tbody>
24.       <% var no=1 %>
25.       <% for(var i = 0; i < warga.length; i++) {%
26.         <tr>
27.           <th scope="row"><%= no++ %></th>
28.           <td><%= warga[i].nama %></td>
29.           <td><%= warga[i].umur %></td>
30.           <td><%= warga[i].pekerjaan %></td>
31.           <!-- menambahkan aksi edit dan delete dengan ambil _id -
->
32.           <td><a href='/warga/edit/<%= warga[i]._id
%>'> <i class="far fa-edit"></i> </a>
33.           <a href='/warga/delete/<%= warga[i]._id
%>' onclick="return confirm('Are you sure you want to delete
this item?');">
34.             <i class="fas fa-trash"></i></a></td>
35.         </tr>
36.       <% } %>
37.     </tbody>
38.   </table>
39. </html>

```

Jalankan perintah “**npm start**” pada terminal visual studio code anda, kemudian buka browser dan ketikan alamat “**localhost:3000**” maka anda akan melihat ada dua buah icon yaitu edit dan hapus, dan ketika anda sorot aka nada link di mana untuk melakukan edit ataupun hapus data berdasarkan **_id** dari collection warga.



Gambar 85. Menampilkan button aksi edit dan hapus data warga

Button untuk melakukan aksi edit dan hapus sudah kita buat, selanjutnya yang perlu dipersiapkan adalah sebuah form untuk menampilkan form edit ditambah sebuah fungsi untuk melakukan update data serta fungsi untuk menghapus data. Form edit akan kita buat pada folder views sedangkan fungsi update atau delete akan kita buat pada folder routes yang didalamnya dipanggil dari file yang tersimpan di controllers. Pertama kita buat dulu fungsi yang akan kita gunakan untuk mengarahkan ke form edit yang akan kita buat di file WargaController.js.

WargaController.js

```

1. var mongoose = require("mongoose");
2. var Warga = require("../models/Warga");
3. var Desa = require("../models/Desa");
4.
5. var wargaController = {};
6.
7. // control untuk melakukan penyimpanan data inputan ke
   collection warga
8. wargaController.save = function (req, res) {
9.   var warga = new Warga(req.body);
10.
11.  warga.save(function (err) {
12.    if (err) {
13.      console.log(err);
14.      res.render('index');

```

```

15.     } else {
16.       console.log("Successfully created an warga.");
17.       res.redirect('../');
18.     }
19.   });
20. });
21.
22. // controll untuk melakukan edit data warga
23. wargaController.edit = function (req, res) {
24.   var id = req.params._id;
25.   Warga.findOne({ _id: id }, function (err, warga) {
26.     if (warga) {
27.       console.log(warga);
28.       //tampilkan semua nama desa
29.       Desa.find({}, function (err, desa) {
30.         console.log(desa);
31.         //tampilkan desa sesuai dengan desa_id warganya
32.         Desa.findOne({ _id: warga.desa_id }, function (err, x)
33.         {
34.           console.log(x);
35.           res.render('editwarga', { warga: warga, title: 'CRUD
Desa', desa: desa, x:x });
36.         })
37.       });
38.     } else {
39.       res.redirect('../');
40.     }
41.   });
42. });
43.
44. module.exports = wargaController;

```

Penjelasan Code :

- Baris ke 2-3 kita include kan models agar kita bisa mengambil data dari collection desa dan warga.
- Baris 23 – 42 kita buat sebuah fungsi yang dicontroll untuk mengarahkan ke form edit. Pada form edit tersebut akan kita isi value input nya dari _id warga yang terseleksi / dipilih.
- Baris 25 jalankan fungsi findOne dimana berfungsi akan menampilkan satu isi document yang terpilih _idnya.
- Baris 29 agar list desa tampil dalam dropdown selectbox ketika akan mengedit data maka kita gunakan fungsi find().
- Baris 32 bertujuan agar desa terpilih akan terseleksi pada selectbox.

- Baris 34 tampilkan query hasil pemilahan fungsi diatas ke dalam view yaitu file editwarga.ejs.

Fungsi router edit sudah kita buat selanjutnya karena pada fungsi tersebut merender ke template yaitu editwarga.ejs maka kita wajib membuat file editwarga.ejs yang akan kita buat dengan code berikut.

editwarga.ejs

```

1. 1. <!DOCTYPE html>
2. 2. <html>
3. 3.   <head>
4. 4.     <title><%= title %></title>
5. 5.     <link rel='stylesheet' href='/stylesheets/style.css' />
6. 6.     <link
7.       rel='stylesheet' href='/bootstrap/css/bootstrap.min.css' />
8. 7.   </head>
9. 8.   <body>
10. 9.     <h1><%= title %></h1>
11. 10.    <p>Edit Data</p>
12. 11.    <form action="warga/update/<%= warga._id
13.       %>" method="post" role="form">
14.      <div class="form-group">
15.        <label for="exampleInputEmail1">Nama</label>
16.        <input type="text" name="nama" class="form-
17.          control" id="" required placeholder="Enter nama" value="<%
18.            warga.nama %>">
19.      </div>
20.      <div class="form-group">
21.        <label for="exampleInputPassword1">Umur</label>
22.        <input type="number" name="umur" class="form-
23.          control" id="" required placeholder="Umur" value="<%
24.            warga.umur %>">
25.      </div>
26.      <div class="form-group">
27.        <label for="exampleInputPassword1">Pekerjaan</label>
28.        <input type="text" name="pekerjaan" class="form-
29.          control" id="" required placeholder="Pekerjaan" value="<%
            warga.pekerjaan %>">
30.      </div>
31.      <div class="form-group">
32.        <label for="exampleInputPassword1">Desa</label>
33.        <select type="text" name="desa_id" class="form-
34.          control" id="" required>
35.          <!-- set selected value nama desanya -->
36.          <option value="<%= x._id
37.            %>" selected><%= x.nama %></option>
38.          <!-- tampilkan dropdown semua desa -->
```

```

30.      <% for(var i = 0; i < desa.length; i++) {%
31.          <option value="<%= desa[i]._id
32.                  %>"><%= desa[i].nama %></option>
33.      <% } %>
34.      </select>
35.      </div>
36.      <button type="submit" class="btn btn-
primary">Submit</button>
37.  </body>
38.  <script
39.    type="javascript" src='/bootstrap/js/bootstrap.min.js'>
39.</html>

```

Penjelasan Code :

- Pada setiap baris anda bisa melihat code value= “”. Denga nisi disesuaikan dengan nama isi fieldnya. Misalkan nama maka diisi dengan <%=warga.nama%> misalnya. Isi warga.nama berasal dari code controller yang sudah kita buat pada WargaController.js ketika anda memanggil render view. Silahkan cek code WargaController.js baris ke 34.
- Untuk form action pada baris ke 11 kita akan menggunakan fungsi update berdasarkan _id nya.
- Baris 26 -33 pada baris ini kita melihat select box yang dimana data yang terselect (selected) adalah data desa yang _id nya sama dengan desa_id pada collection warga.

Agar fungsi yang sudah kita buat di controller dapat berjalan maka kita harus mengubah code pada router warga.js dengan menambahkan beberapa fungsi berikut.

```

router.get('/edit/:_id', function (req, res) {
  warga.edit(req,res);
});

```

Untuk code lengkapnya adalah sebagai berikut.

warga.js

```
1. var express = require('express');
2. var router = express.Router();
3. var mongoose = require('mongoose');
4. var warga = require("../controllers/WargaController.js");
5. var Desa = require("../models/Desa");
6.
7. //ambil tampilan render dari folder view berformat ejs
8. router.get('/', function(req, res) {
9.   Desa.find({}, function (err, desa) {
10.     console.log(desa);
11.     res.render('warga', {desa: desa ,title: 'CRUD Desa'});
12.   }).select('_id nama');
13. });
14.
15. //fungsi POST menambah data desa
16. router.post('/tambah', function(req, res) {
17.   warga.save(req, res);
18. });
19.
20. //fungsi EDIT data warga
21. router.get('/edit/:_id', function (req, res) {
22.   warga.edit(req,res);
23. });
24.
25. module.exports = router;
```

Penjelasan Code :

- Fungsi ditambahkan pada baris ke 21 – 23 dimana variable warga kita ambil dari isi controller yang kita includekan pada baris ke 4.
- Pada get fungsi edit tersebut kita melakukan get _id dari isi document collection warga. Misalkan ada tiga buah baris data pada collection warga dan kita memilih baris ke 2 misalnya, baris ke 2 tersebut memiliki _id, nah _id tersebut merupakan kunci yang akan mengarahkan kita ke halaman editwarga disesuaikan datanya berdasarkan _id tersebut.
- Jalankan **npm start** pada terminal, lalu akses browser anda pada port 3000

#	Nama	Umur	Pekerjaan	Action
1	Andri	23	Programmer	
2	Johan	23	Petani	

Gambar 86. Halaman home yang akan kita edit atau hapus

CRUD Desa

Edit Data

Nama

Umur

Pekerjaan

Desa

Gambar 87. Halaman form edit jika kita pilih button edit

Pada gambar 87 kita melihat form edit dimana valuenya berisi sesuai dengan nama warganya. Hal tersebut sudah kita akali pada fungsi yang kita buat di file WargaController.js. Pada kolom select box desa, maka langsung terseleksi nama desa warganya.

Agar setiap kali kita melakukan perubahan data pada form warga, maka kita harus membuat sebuah fungsi kembali untuk mengupdate. Anda pernah belajar melakukan update data pada bab ke dua, mungkin anda bias melakukan kilas balik agar lebih paham. Nah agar data warga bias berubah alias update maka kita akan menambahkan beberapa fungsi code pada file WargaController.js seperti berikut.

Tambahan Code WargaController.js

```

1. // control untuk melakukan penyimpanan data inputan ke
   collection warga
2. wargaController.update = function (req, res) {
3.   Warga.findByIdAndUpdate(req.params._id, {
4.     $set: {
5.       nama: req.body.nama,
6.       umur: req.body.umur,
7.       pekerjaan: req.body.pekerjaan,
8.       desa_id: req.body.desa_id
9.     }
10.   }, { new: true }, function (err, warganya) {
11.     if (err) {
12.       console.log(err);
13.       res.render('editwarga', { warganya: req.body });
14.     }
15.     res.redirect('http://localhost:3000');
16.   });
17. };

```

Penjelasan Code:

- Baris 2 kita menciptakan turunan fungsi yang akan kita panggil di routes yaitu fungsi update.
- Baris 3 kita cari terlebih dahulu _id dari documents yang hendak kita ubah,
- Baris 4 setelah _id nya kita dapatkan lalu set datanya sesuai dengan data baru yang diinputkan hal ini ditandai dengan fungsi \$set
- Jika berhasil terupdate maka akan diarahkan ke halaman home.

Kode lengkap WargaController.js

```

1. var mongoose = require("mongoose");
2. var Warga = require("../models/Warga");
3. var Desa = require("../models/Desa");
4.
5. var wargaController = {};
6.
7. // control untuk melakukan penyimpanan data inputan ke
   collection warga
8. wargaController.save = function (req, res) {
9.   var warga = new Warga(req.body);
10.
11. warga.save(function (err) {

```

```

12.    if (err) {
13.      console.log(err);
14.      res.render('index');
15.    } else {
16.      console.log("Successfully created an warga.");
17.      res.redirect('../');
18.    }
19.  });
20.});
21.
22.// control untuk melakukan edit data warga
23.wargaController.edit = function (req, res) {
24.  var id = req.params._id;
25.  Warga.findOne({ _id: id }, function (err, warga) {
26.    if (warga) {
27.      console.log(warga);
28.      //tampilkan semua nama desa
29.      Desa.find({}, function (err, desa) {
30.        console.log(desa);
31.        //tampilkan desa sesuai dengan desa_id warganya
32.        Desa.findOne({ _id: warga.desa_id }, function (err, x
33.        ) {
34.          console.log(x);
35.          res.render('editwarga', { warga: warga, title: 'CRU
D Desa', desa: desa, x: x });
36.        });
37.      }
38.    } else {
39.      res.redirect('../');
40.    }
41.  });
42.});
43.
44.// control untuk melakukan penyimpanan data inputan ke
collection warga
45.wargaController.update = function (req, res) {
46.  Warga.findByIdAndUpdate(req.params._id, {
47.    $set: {
48.      nama: req.body.nama,
49.      umur: req.body.umur,
50.      pekerjaan: req.body.pekerjaan,
51.      desa_id: req.body.desa_id
52.    }
53.  }, { new: true }, function (err, warganya) {
54.    if (err) {
55.      console.log(err);
56.      res.render('editwarga', { warganya: req.body });
57.    }
58.    res.redirect('http://localhost:3000');
59.  });
60.});
61.
62.module.exports = wargaController;

```

Panggil fungsi yang sudah kita tambahkan di controller tersebut pada file warga.js yang terdapat pada folder routes.

warga.js

```

1. var express = require('express');
2. var router = express.Router();
3. var mongoose = require('mongoose');
4. var warga = require("../controllers/WargaController.js");
5. var Desa = require("../models/Desa");
6.
7. //ambil tampilan render dari folder view berformat ejs
8. router.get('/', function(req, res) {
9.   Desa.find({}, function (err, desa) {
10.     console.log(desa);
11.     res.render('warga', {desa: desa ,title: 'CRUD Desa'});
12.   }).select('_id nama');
13. });
14.
15. //fungsi POST menambah data desa
16. router.post('/tambah', function(req, res) {
17.   warga.save(req, res);
18. });
19.
20. //fungsi EDIT data warga
21. router.get('/edit/:_id', function (req, res) {
22.   warga.edit(req,res);
23. });
24.
25. //fungsi POST mengupdate data desa
26. router.post('/update/:_id', function(req, res) {
27.   warga.update(req, res);
28. });
29.
30. module.exports = router;

```

Penjelasan Code:

- Baris 26 – 28 adalah fungsi post untuk mengupdate data warga jika terjadi perubahan berdasarkan _id terpilih.
- Jalankan npm start pada terminal lalu lakukan uji coba melakukan perubahan data.

CRUD Warga				
Welcome to CRUD Warga				
Data Warga				
#	Nama	Umur	Pekerjaan	Action
1	Agustinus Hendra	29	Programmer T	
2	Johan	23	Petani	
3	Johan Simima	23	Petani	
4	Yaka Andri	23	Programmer	

Gambar 88. Halaman home yang belum di edit

Pilihlah fungsi edit data misalkan kita hendak mengubah nama Johan, lalu kita klik icon editnya, secara otomatis akan diarahkan ke halaman edit data nama Johan.

CRUD Desa				
Edit Data				
Nama	<input type="text" value="Johan"/>			
Umur	<input type="text" value="23"/>			
Pekerjaan	<input type="text" value="Petani"/>			
Desa	<input type="text" value="Paningkaban"/>			
	<input type="button" value="Submit"/>			

Gambar 89. Halaman form edit nama Johan

Kita coba edit datanya dengan diganti menjadi nama Johan Saimima dan Profesinya adalah Desainer Grafis, lalu tekan Submit maka data Johan akan berubah.

CRUD Warga

Welcome to CRUD Warga

Data Warga

#	Nama	Umur	Pekerjaan	Action
1	Agustinus Hendra	29	Programmer T	
2	Johan Saimima	23	Desainer Grafis	
3	Johan Simima	23	Petani	
4	Yaka Andri	23	Programmer	

Gambar 90. Nama Johan sudah terupdate

Fungsi tambah dan edit data sudah kita buat, langkah terakhir adalah langkah untuk menghapus data. Kita akan membuat tambahan fungsi pada controller yaitu menghapus data berdasarkan _id warga yang akan kita hapus datanya. Buka file WargaController.js lalu tambahkan code berikut.

```

1. // control untuk melakukan hapus data
2. wargaController.delete = function (req, res) {
3.   Warga.findOne({ _id: req.params._id }, function (err, row) {
4.     if (row) {
5.       console.log(row);
6.       Warga.remove({ _id: req.params._id }, function (err) {
7.         // If error
8.         if (err) {
9.           console.log("Delete Warga Error", err);
10.        }
11.        else {
12.          console.log("Warga deleted!");
13.          res.redirect('http://localhost:3000');
14.        }
15.      });
16.    }
17.    else {
18.      res.redirect('http://localhost:3000');
19.    }
20.  });
21. }
```

Penjelasan Code:

- Baris 2 kita akan membuat fungus delete data

- Baris 3 gunakan findOne untuk mengambil satu data warga yang hendak kita hapus berdasarkan _id nya.
- Baris 4 jika menemukan datanya
- Baris 6 Lakukan penghapusan data berdasarkan _id terpilih
- Jika benar arahkan ke halaman home

Langkah terakhir adalah memanggil fungsi yang sudah kita buat di controller tersebut pada router warga.js. Code pada warga.js kita ubah menjadi seperti berikut ini.

warga.js

```

1. var express = require('express');
2. var router = express.Router();
3. var mongoose = require('mongoose');
4. var warga = require("../controllers/WargaController.js");
5. var Desa = require("../models/Desa");
6.
7. //ambil tampilan render dari folder view berformat ejs
8. router.get('/', function(req, res) {
9.   Desa.find({}, function (err, desa) {
10.     console.log(desa);
11.     res.render('warga', {desa: desa ,title: 'CRUD Desa'});
12.   }).select('_id nama');
13. });
14.
15.//fungsi POST menambah data desa
16.router.post('/tambah', function(req, res) {
17.   warga.save(req, res);
18. });
19.
20.//fungsi EDIT data warga
21.router.get('/edit/:_id', function (req, res) {
22.   warga.edit(req,res);
23. });
24.
25.//fungsi POST mengupdate data desa
26.router.post('/update/:_id', function(req, res) {
27.   warga.update(req, res);
28. });
29.
30.//fungsi DELETE warga
31.router.get('/delete/:_id', function (req, res) {
32.   warga.delete(req,res);
33. });
34.
35.module.exports = router;

```

Penjelasan Code:

- Fungsi delete data kita tempatkan pada baris ke 31-33 dimana bias anda lihat kita membuat fungsi menghapus satu data berdasarkan _idnya.

Kode WargaController.js selengkapnya.

WargaController.js

```

1. var mongoose = require("mongoose");
2. var Warga = require("../models/Warga");
3. var Desa = require("../models/Desa");
4.
5. var wargaController = {};
6.
7. // control untuk melakukan penyimpanan data inputan ke
   collection warga
8. wargaController.save = function (req, res) {
9.   var warga = new Warga(req.body);
10.
11.  warga.save(function (err) {
12.    if (err) {
13.      console.log(err);
14.      res.render('index');
15.    } else {
16.      console.log("Successfully created an warga.");
17.      res.redirect('../');
18.    }
19.  });
20.};
21.
22.// control untuk melakukan edit data warga
23.wargaController.edit = function (req, res) {
24.  var id = req.params._id;
25.  Warga.findOne({ _id: id }, function (err, warga) {
26.    if (warga) {
27.      console.log(warga);
28.      //tampilkan semua nama desa
29.      Desa.find({}, function (err, desa) {
30.        console.log(desa);
31.        //tampilkan desa sesuai dengan desa_id warganya
32.        Desa.findOne({ _id: warga.desa_id }, function (err, x
33. ) {
34.          console.log(x);
35.          res.render('editwarga', { warga: warga, title: 'CRU
   D Desa', desa: desa, x: x });
36.        })
37.      });
38.    }
39.  });
40.
41.  res.render('editwarga', { warga: warga, title: 'CRU
   D Desa', desa: desa, x: x });
42.};
43.
44. module.exports = wargaController;

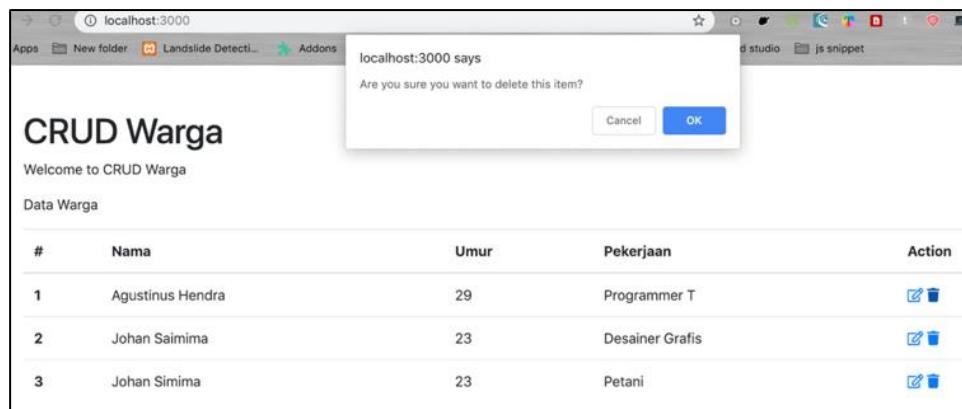
```

```

37.      }
38.    else {
39.      res.redirect('../');
40.    }
41.  });
42.});
43.
44.// controll untuk melakukan update data inputan ke collection
45.wargaController.update = function (req, res) {
46.  Warga.findByIdAndUpdate(req.params._id, {
47.    $set: {
48.      nama: req.body.nama,
49.      umur: req.body.umur,
50.      pekerjaan: req.body.pekerjaan,
51.      desa_id: req.body.desa_id
52.    }
53.  }, { new: true }, function (err, warganya) {
54.    if (err) {
55.      console.log(err);
56.      res.render('editwarga', { warganya: req.body });
57.    }
58.    res.redirect('http://localhost:3000');
59.  });
60.});
61.
62.// controll untuk melakukan hapus data
63.wargaController.delete = function (req, res) {
64.  Warga.findOne({ _id: req.params._id }, function (err, row) {
65.    if (row) {
66.      console.log(row);
67.      Warga.remove({ _id: req.params._id }, function (err) {
68.        // If error
69.        if (err) {
70.          console.log("Delete Warga Error", err);
71.        }
72.        else {
73.          console.log("Warga deleted!");
74.          res.redirect('http://localhost:3000');
75.        }
76.      });
77.    }
78.    else {
79.      res.redirect('http://localhost:3000');
80.    }
81.  });
82.}
83.
84.
85.module.exports = wargaController;

```

Untuk melakukan test jalankan perintah “npm start” pada terminal visual studio code anda, dan akses alamat browser localhost:3000 lalu coba anda pilih salah satu file yang hendak anda hapus. Semoga berhasil.



Gambar 91. Menghapus Data Warga

BAB IV

TENTANG PENULIS



Ipung Purwono, lahir di Banyumas, 16 Mei 1989 adalah seorang yang senang mempelajari dunia programming baik website, desktop ataupun mobile. Saat ini aktif mengelola Banyu Center di Kota Purwokerto <https://banyu.center>.

Rutinitas sehari-hari adalah melakukan riset dan mempelajari segala hal tentang bahasa pemrograman dan sering bekerja sama dengan berbagai kampus dimana hasil riset tersebut diimplementasikan menjadi sebuah aplikasi yang diharapkan berguna bagi masyarakat.

Penulis juga sering mengisi berbagai seminar ataupun workshop yang berkaitan dengan dunia pemrograman.

Penulis dapat dihubungi pada kontak berikut:

- WhatsApp : 0821-1394-0427
- Email : ipungofficial@gmail.com
- Web : <https://banyu.center>
- Facebook : <https://facebook.com/ipungz.purwono>
- Instagram : <https://instagram.com/ipungdev>

Daftar Pustaka

<https://nodejs.org/en/docs/>

<https://www.w3schools.com/nodejs/>

https://www.w3schools.com/nodejs/nodejs_mongodb.asp

<https://expressjs.com/>

<https://ejs.co/#docs>

<https://docs.mongodb.com>