**Repository :** [**https://github.com/hermawanumdp/SI5BWP2**](https://github.com/hermawanumdp/SI5BWP2)

**Pertemuan I, 10 September 2024**

Software :

1. Visual Studio code (IDE)

2. NPM (Node Package Manager)

<https://www.npmjs.com/>

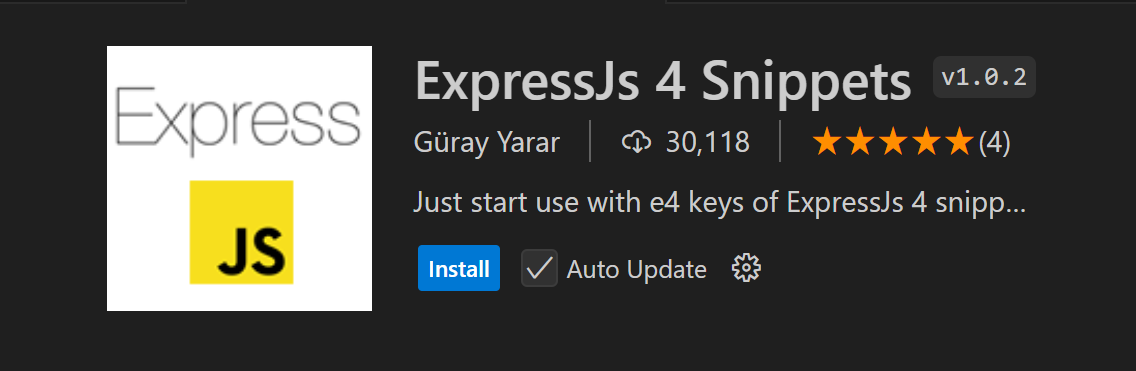
3. Node Js

<https://nodejs.org/en>

*\*Download Node JS, instalasi sudah include NPM*

**Pertemuan II, 12 September 2024**

1. **Express JS** [**https://expressjs.com**](https://expressjs.com)
2. **Tambahkan extension di VS Code (*Snippets*)**

****

1. **Inisialiasi folder project, menambahkan package JSON (Terminal)**

**npm init -y**

**-y ⇒ nilai default**

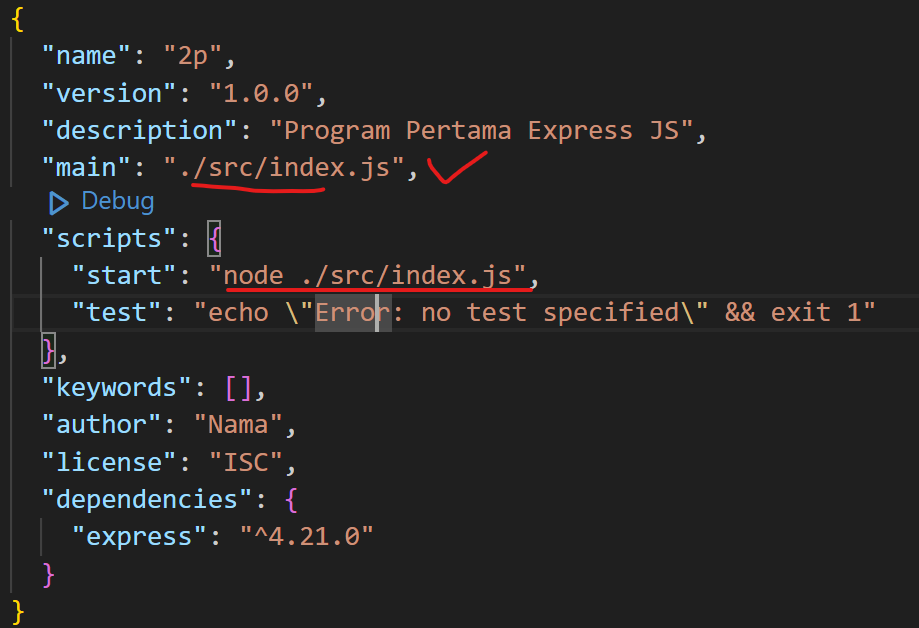
1. **Install / download framework express JS**

**npm install express**

1. **Menambahkan folder baru SRC, dan file index.js**

****

1. **Edit Package.json**

****

1. **Running**

**Di terminal dan lokasi folder, ketik :**

**npm start atau npm run start**

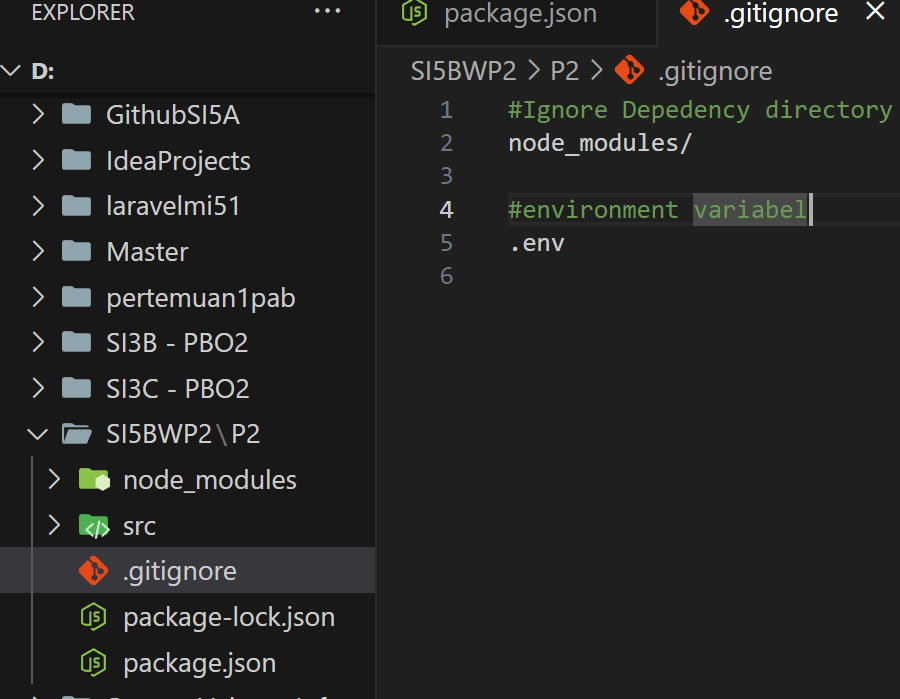
1. **Exit**

**Tekan tombol CTRL C**

1. **Optional (Gitignore)**

**Menambahkan file .gitignore yang isinya adalah daftar yang tidak akan di upload ke Github. Contoh node\_modules**

**Cloning/dowload dari github, perlu npm install lagi untuk menginstall depedency yang diperlukan**

****

1. **Final Code index.js**

| **const express= require('express')**  **const app = express()**  **const port= 3000**  **app.get('/', (req, res) => {**  **res.send('Hello world');**  **});**  **app.get('/about', (req, res) => {**  **res.send('Halaman Tentang Perusahaan');**  **});**  **app.get('/contact', (req, res) => {**  **res.send("Halaman Contact");**  **});**  **//middleware**  **app.use('\*', (req,res) =>{**  **res.status(404);**  **res.send("404, Halaman tidak ditemukan");**  **});**  **app.listen(port, () => {**  **console.log('Server started on port 3000');**  **});** |
| --- |

**Pertemuan 3, 17 September 2024**

**Routing**

1. Inisitalisasi folder

**npm init -y**

1. Install express JS

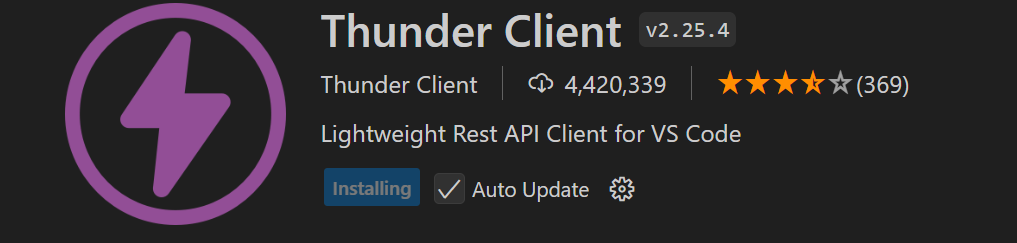
**npm install express**

1. Buat file index.js

| const express = require('express');  const app = express();  **const port =3000;**  app.get('/', (req, res) => {  res.send('Hello World!');  });  app.listen(port, () => {  console.log(**'Example app listening on port ' + port**);  }); |
| --- |

1. Software untuk menguji routing / API

* Postman (Download software)
* Thunder Client (Extension di Visual Studio)



1. Running npm run start
2. Tambahkan folder routes, dan file index.js dan mahasiswa.js
3. router/mahasiswa.js

| const express = require('express');  const mahasiswaRouter = express.Router**()**;  mahasiswaRouter  .route("/")  .get((req, res) => {  res.send("Get Data Mahasiswa");  })  .post((req, res) => {  res.send("Post Mahasiswa");  });  //localhost:3000/mahasiswa/2226240082  mahasiswaRouter  .route("/:npm")  .put((req, res) => {  res.send("Put Mahasiswa " + req.params.npm);  })  .delete((req, res) => {  res.send("Delete mahasiswa " + req.params.npm);  });  // localhost:3000/mahasiswa/kelas?idkelas=SI5B  **mahasiswaRouter.route("/kelas")**  **.get((req, res)=>{**  **const kelas = req.query.idkelas;**  **res.send("Kelasnya adalah " + kelas);**  **});**  **module.exports = mahasiswaRouter** |
| --- |

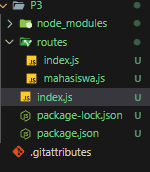
router/index.js

| const express = require('express');  const router = express.Router();  const mhsRouter = require("./mahasiswa");  router.use("/mahasiswa",mhsRouter);  module.exports = router; |
| --- |

1. File Index (Utama)

| const express = require('express');  const app = express();  const port =3000;  const route= require("./routes/index");  app.get('/', (req, res) => {  res.send('Hallo Dunia!');  });  app.use(route);  // app.get('/mahasiswa', (req, res) => {  // res.send("Get Data Mahasiswa");  // });  // app.post('/mahasiswa', (req, res) => {  // res.send("Post Mahasiswa");  // });  // app.put('/mahasiswa/:npm', (req, res) => {  // res.send("Put Mahasiswa " + req.params.npm);  // });  // app.delete('/mahasiswa/:npm', (req, res) => {  // res.send("Delete mahasiswa " + req.params.npm);  // });  app.listen(port, () => {  console.log('Example app listening on port ' + port);  });  //Run app, then load http://localhost:port in a browser to see the output. |
| --- |

1. Struktur project :

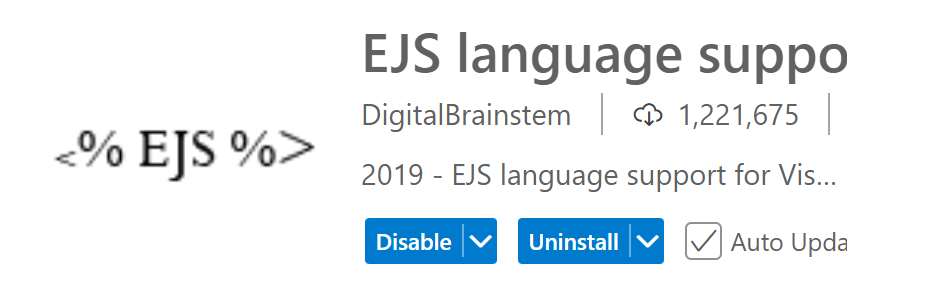


**Pertemuan 4, 19 September 2024**

**Template Engine, EJS (Embedded Javascript Template)**

**The** [**Express application generator**](https://expressjs.com/en/starter/generator.html) **uses** [**Pug**](https://pugjs.org/api/getting-started.html) **as its default, but it also supports** [**Handlebars**](https://www.npmjs.com/package/handlebars)**, and** [**EJS**](https://www.npmjs.com/package/ejs)**, among others.**

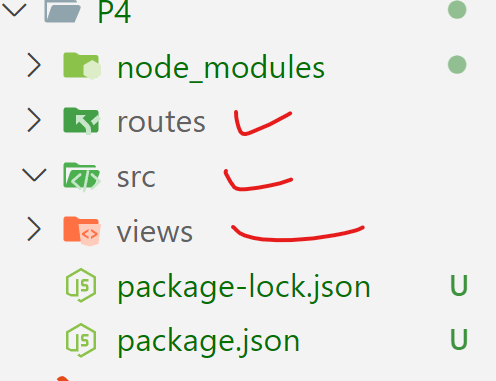
**Syarat Extension :**

****

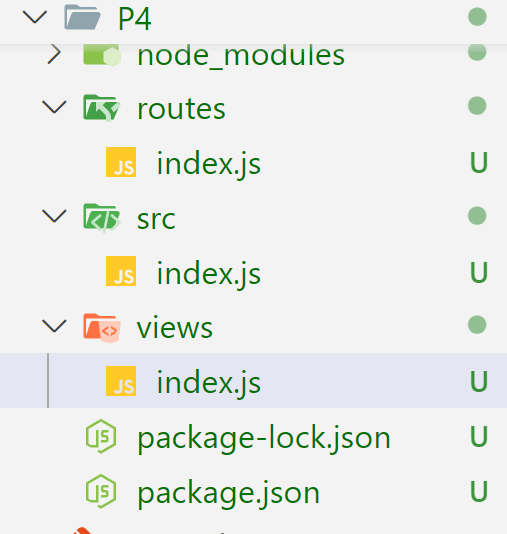
1. Inisialisasi folder

**npm init -y**

1. **npm install express**
2. Buat Folder Baru, routes, src dan views



1. Tambahkan index.js di masing routes dan src, khusus views tambahkan index.ejs



1. **Src/index.js**

| const express = require('express');  const app = express();  const port=3000;  app.get('/', (req, res) => {  res.send('Hello World!');  });  app.listen(port, () => {  console.log('Server listening on port ' + port + "!");  });  //Run app, then load http://localhost:port in a browser to see the output. |
| --- |

1. package.json

| {  "name": "p4",  "version": "1.0.0",  "description": "",  **"main": "./src/index.js",**  "scripts": {  **"start": "node ./src/index.js",**  "test": "echo \"Error: no test specified\" && exit 1"  },  "keywords": [],  "author": "",  "license": "ISC",  "dependencies": {  "express": "^4.21.0",  "nodemon": "^3.1.5"  }  } |
| --- |

1. Install EJS

**npm install ejs**

1. Update src/index.js

| const express = require('express');  const app = express();  const port=3000;  **const routes =require('../routes/index');**  **app.use(routes);**  **//Ejs**  **app.set("view engine","ejs");**  **app.set("views", "./views");**  // app.get('/', (req, res) => {  // res.send('Hello World!');  // });  app.listen(port, () => {  console.log('Server listening on port ' + port + "!");  });  //Run app, then load http://localhost:port in a browser to see the output. |
| --- |

1. routes/index.js

| const express = require('express');  const router = express.Router();  router.use("/",(req, res)=>{  res.render("index");  });  **module.exports = router;** |
| --- |

1. **Update routes/index.js**

| const express = require('express');  const router = express.Router();  router.use("/",(req, res)=>{  //format data JSON  **const data ={**  **caption: "Mahasiswa",**  **data :[**  **{**  **npm : "2226240021",**  **nama : "Budi"**  **},**  **{**  **npm : "2226240031",**  **nama : "Dewi"**  **},**  **{**  **npm : "2226240051",**  **nama : "Hendra"**  **}**  **]**  **};**  **res.render("index" , data);**  //menuju ke views/index.ejs  });  module.exports = router; |
| --- |

1. Update views/index.js

| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  </head>  <body>  <% let title ="Template engine" %>  <%= title %>  <br>  <%= caption %>  <table>  <tr>  <th>No</th>  <th>NPM</th>  </tr>  <% data.forEach((element,no) => { %>  <tr>  <td><%= no+=1 %></td>  <td><%= element.npm %></td>  </tr>  <%}); %>  </table>  </body>  </html> |
| --- |

1. Membuat template layout

**npm i express-ejs-layouts**

1. Buat folder layouts/main-layout.ejs

| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=<device-width>, initial-scale=1.0">  <title>Document</title>  </head>  <body>  <%- body %>  </body>  </html> |
| --- |

1. Update views/index.ejs

| <% let title ="Template engine" %>  <%= title %>  <br>  <%= caption %>  <table>  <tr>  <th>No</th>  <th>NPM</th>  </tr>  <% data.forEach((element,no) => { %>  <tr>  <td><%= no+=1 %></td>  <td><%= element.npm %></td>  </tr>  <%}); %>  </table> |
| --- |

1. Update routes/indes.js

| const express = require('express');  const router = express.Router();  router.use("/",(req, res)=>{  //format data JSON  const data ={  caption: "Mahasiswa",  **layout : "layout/main-layout",**  data :[  {  npm : "2226240021",  nama : "Budi"  },  {  npm : "2226240031",  nama : "Dewi"  },  {  npm : "2226240051",  nama : "Hendra"  }  ]  };  res.render("index" , data);  //menuju ke views/index.ejs  });  module.exports = router; |
| --- |

**Pertemuan 5, 24 September 2024**

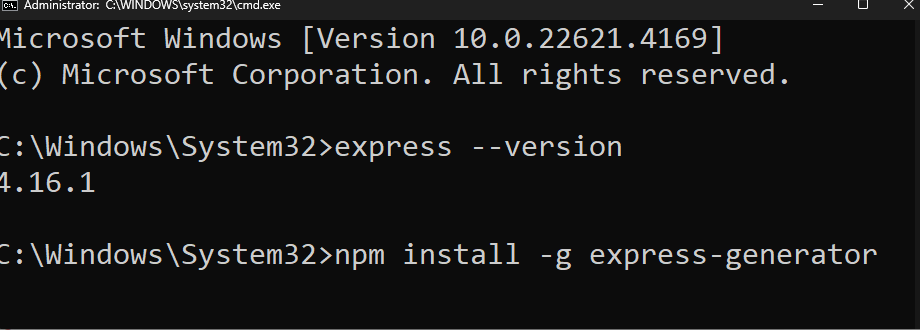
**Middleware, express generator**

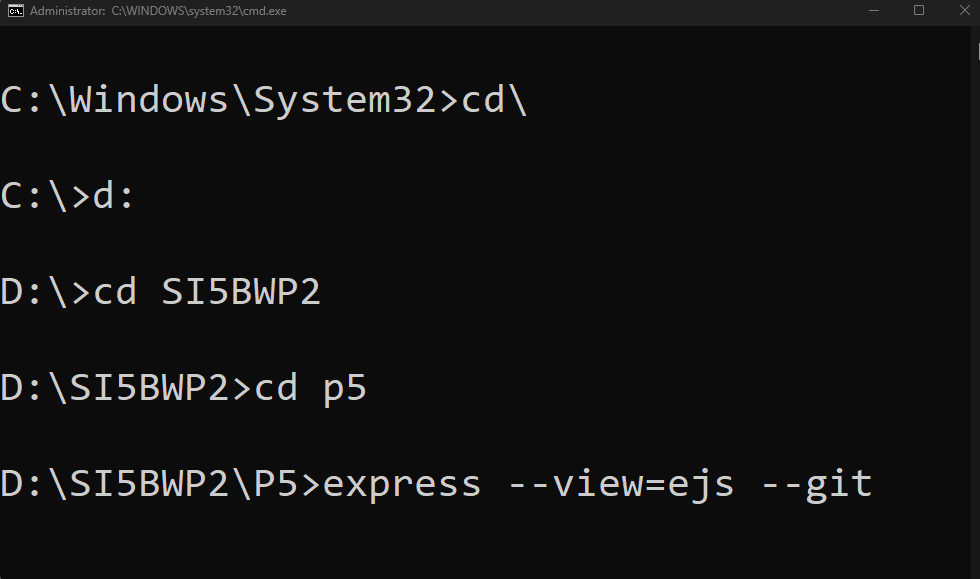
1. Cek instalasi global express generator

**Express - - version**

Kalau belum install, dengan

**npm install -g express-generator**





1. Install dependency

**npm install**

Running dng **npm start**

1. Install nodemon

Package untuk restart aplikasi setiap ada perubahan kode program

**nodemon - -version**

Install => **npm install -g nodemon**

1. Menggunakan nodemon, di package.json, tambahkan :

| {  "name": "p5",  "version": "0.0.0",  "private": true,  "scripts": {  **"start-dev" : "nodemon ./bin/www",**  "start": "node ./bin/www"  },  "dependencies": {  "cookie-parser": "~1.4.4",  "debug": "~2.6.9",  "ejs": "~2.6.1",  "express": "~4.16.1",  "http-errors": "~1.6.3",  "morgan": "~1.9.1"  }  } |
| --- |

1. Running dengan nodemon : **npm run start-dev**
2. Middleware, tambahkan di app.js

| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/index');  var usersRouter = require('./routes/users');  var app = express();  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  **app.use((req, res, next)=>{**  **console.log("Mulai log middleware");**  **next();**  **});**  **app.use((req, res, next)=>{**  **console.log("Tanggal =" + Date());**  **next();**  **});**  app.use('/', indexRouter);  app.use('/users', usersRouter);  **function logOriginalUrl (req, res, next) {**  **console.log('Request URL:', req.originalUrl)**  **next()**  **}**  **function logMethod (req, res, next) {**  **console.log('Request Type:', req.method)**  **next()**  **}**  **const logStuff = [logOriginalUrl, logMethod]**  **app.get('/appmiddle', logStuff, (req, res, next) => {**  **res.send('User Info')**  **})**  // catch 404 and forward to error handler  app.use(function(req, res, next) {  next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {  // set locals, only providing error in development  res.locals.message = err.message;  res.locals.error = req.app.get('env') === 'development' ? err : {};  // render the error page  res.status(err.status || 500);  res.render('error');  });  module.exports = app; |
| --- |

Contoh akses url :

**localhost:3000**

**localhost:3000/appmiddle**

1. Router middleware, di Router\users.js

Akses url :

<http://localhost:3000/users/user>

<http://localhost:3000/users/0>

<http://localhost:3000/users/100>

| var express = require('express');  var router = express.Router();  /\* GET users listing. \*/  router.get('/', function(req, res, next) {  res.send('respond with a resource');  });  //contoh pertama  **router.use('/user', (req, res, next) => {**  **console.log('Router middleware - Request URL:', req.originalUrl)**  **next()**  **}, (req, res, next) => {**  **console.log('Router middleware - Request Type:', req.method)**  **next()**  **});**  **router.get('/user', (req, res) => {**  **res.send("Hello dari contoh router middleware");**  **});**  //contoh kedua  **router.get('/user/:id', (req, res, next) => {**  **// if the user ID is 0, skip to the next router**  **if (req.params.id === '0') next('route')**  **// otherwise pass control to the next middleware function in this stack**  **else next()**  **}, (req, res, next) => {**  **// render a regular page**  **res.send('regular')**  **})**  **// handler for the /user/:id path, which renders a special page**  **router.get('/user/:id', (req, res, next) => {**  **console.log(req.params.id)**  **res.send('special')**  **})**  module.exports = router; |
| --- |

1. Error Handling Middleware

Update file app.js

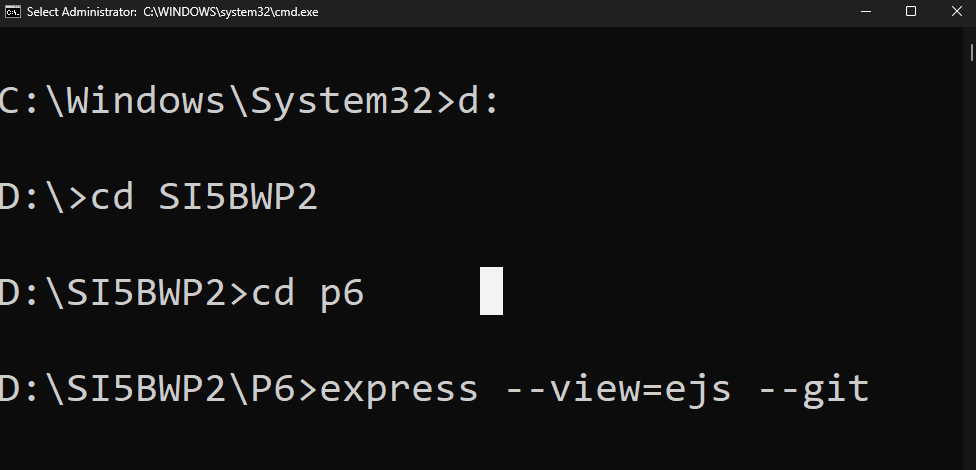
Akses URL : **localhost:3000/errmid**

| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/index');  var usersRouter = require('./routes/users');  var app = express();  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use((req, res, next)=>{  console.log("Mulai log middleware");  next();  });  app.use((req, res, next)=>{  console.log("Tanggal =" + Date());  next();  });  app.use('/', indexRouter);  app.use('/users', usersRouter);  function logOriginalUrl (req, res, next) {  console.log('Request URL:', req.originalUrl)  next()  }  function logMethod (req, res, next) {  console.log('Request Type:', req.method)  next()  }  const logStuff = [logOriginalUrl, logMethod]  app.get('/appmiddle', logStuff, (req, res, next) => {  res.send('User Info')  })  app.get('/errmid', (req, res ,next) => {  throw new Error('Ada error, hubungi Administrator');  });  app.use((err, req, res, next)=>{  console.log(err);  res.status(500).send(err.toString());  });  // catch 404 and forward to error handler  app.use(function(req, res, next) {  next(createError(404));  });  // error handler  **app.use(function(err, req, res, next) {**  **// set locals, only providing error in development**  **res.locals.message = err.message;**  **res.locals.error = req.app.get('env') === 'development' ? err : {};**  **// render the error page**  **res.status(err.status || 500);**  **res.render('error');**  **});**  module.exports = app; |
| --- |

**Pertemuan 6, 26 September 2024**

**Public, template, static site**

1. Buat project baru



1. Download dependencies yang dibutuhkan

**npm install**

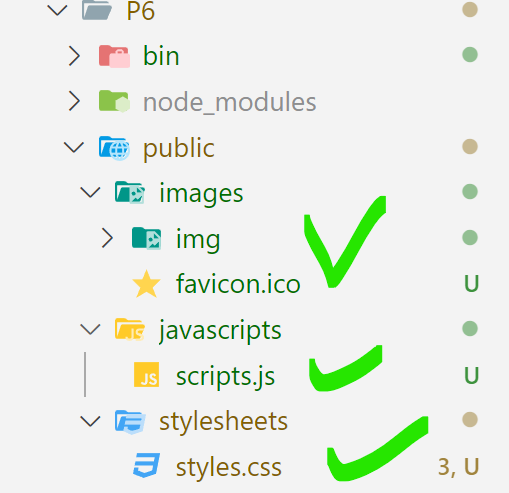
1. Ubah package.json

| {  "name": "p6",  "version": "0.0.0",  "private": true,  "scripts": {  **"start-dev" : "nodemon ./bin/www",**  "start": "node ./bin/www"  },  "dependencies": {  "cookie-parser": "~1.4.4",  "debug": "~2.6.9",  "ejs": "~2.6.1",  "express": "~4.16.1",  "http-errors": "~1.6.3",  "morgan": "~1.9.1"  }  } |
| --- |

1. **npm run start-dev**
2. <https://startbootstrap.com/>

<https://startbootstrap.com/theme/clean-blog>

1. Copy file css, js, dan image template page ke folder public di project express js



1. Penyesuaian lokasi dari CSS

| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8" />  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />  <meta name="description" content="" />  <meta name="author" content="" />  <title>Clean Blog - Start Bootstrap Theme</title>  <link rel="icon" type="image/x-icon" href="assets/favicon.ico" />  <!-- Font Awesome icons (free version)-->  <script src="https://use.fontawesome.com/releases/v6.3.0/js/all.js" crossorigin="anonymous"></script>  <!-- Google fonts-->  <link href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic" rel="stylesheet" type="text/css" />  <link href="https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700italic,800italic,400,300,600,700,800" rel="stylesheet" type="text/css" />  <!-- Core theme CSS (includes Bootstrap)-->  <link href=**"stylesheets/styles.css"** rel="stylesheet" />  </head>  <body> |
| --- |

1. Penyesuaian lokasi Javascript

| </div>  </footer>  <!-- Bootstrap core JS-->  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>  <!-- Core theme JS-->  <script src=**"javascripts/scripts.js"**></script>  </body>  </html> |
| --- |

1. Penyesuaian lokasi gambar

| <header class="masthead" style="background-image: url('**images/img/home-bg.jpg**')">  <div class="container position-relative px-4 px-lg-5">  <div class="row gx-4 gx-lg-5 justify-content-center">  <div class="col-md-10 col-lg-8 col-xl-7">  <div class="site-heading">  <h1>Clean Blog</h1>  <span class="subheading">A Blog Theme by Start Bootstrap</span>  </div>  </div>  </div>  </div>  </header> |
| --- |

1. Penambahan page about.ejs di view dan routes

Tambahkan file about.ejs di view dan copy dari template page (about.html) ke file about.ejs di express.js

Tambahkan routing di routes\index.js

| var express = require('express');  var router = express.Router();  /\* GET home page. \*/  router.get('/', function(req, res, next) {  res.render('index', { title: 'Express' });  });  **router.get('/about', (req, res) => {**  **res.render('about');**  **});**  module.exports = router; |
| --- |

1. Modifikasi navigasi di index.ejs

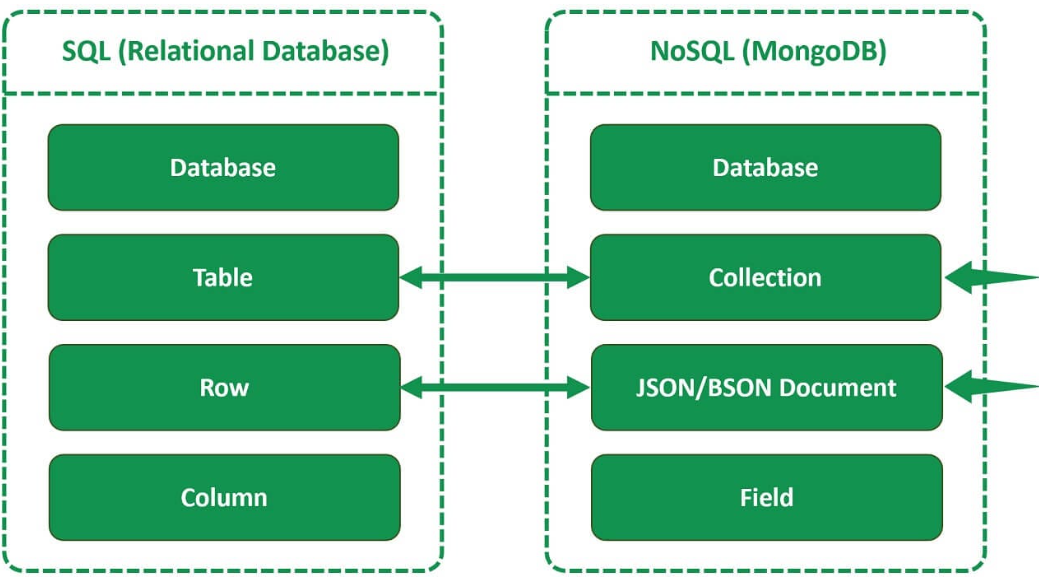
| <nav class="navbar navbar-expand-lg navbar-light" id="mainNav">  <div class="container px-4 px-lg-5">  <a class="navbar-brand" href="index.html">Start Bootstrap</a>  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">  Menu  <i class="fas fa-bars"></i>  </button>  <div class="collapse navbar-collapse" id="navbarResponsive">  <ul class="navbar-nav ms-auto py-4 py-lg-0">  <li class="nav-item"><a class="nav-link px-lg-3 py-3 py-lg-4" **href="/"**>Home</a></li>  <li class="nav-item"><a class="nav-link px-lg-3 py-3 py-lg-4" **href="/about"**>About</a></li>  <li class="nav-item"><a class="nav-link px-lg-3 py-3 py-lg-4" href="post.html">Sample Post</a></li>  <li class="nav-item"><a class="nav-link px-lg-3 py-3 py-lg-4" href="contact.html">Contact</a></li>  </ul>  </div>  </div>  </nav> |
| --- |

1. Modifikasi kembali dan buatlah template layout (pertemuan 4)

**npm install express-ejs-layouts**

**Pertemuan 7, 01 Oktober 2024**

**Pengenalan MongoDB**

****

Software :

1. MongoDB lokal

<https://www.mongodb.com/products/self-managed/community-edition>

Engine + MongoDB Compass + MongoDB Shell

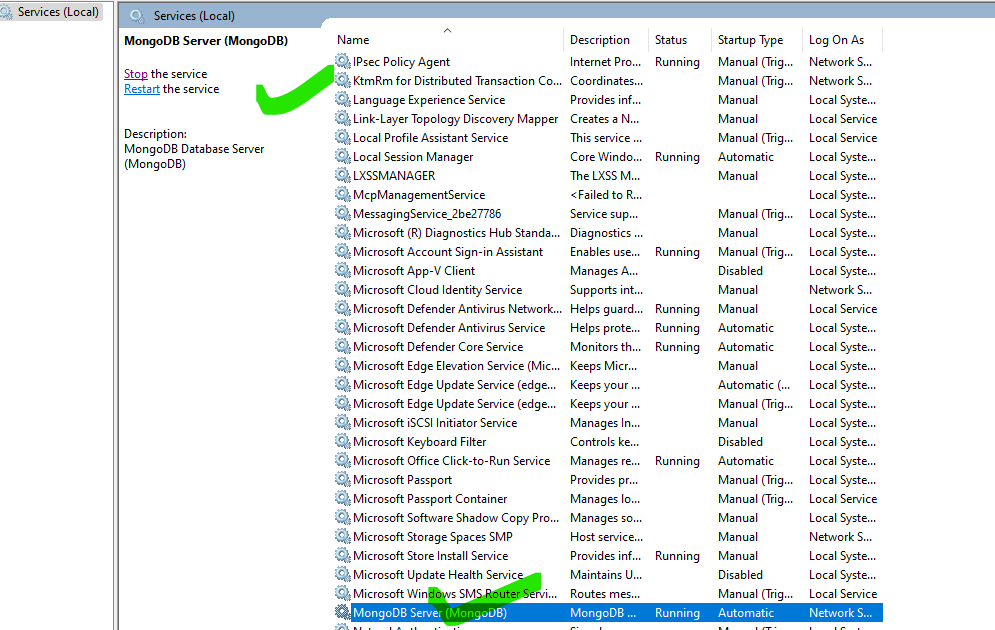
MongoDB Shell CRUD Creation :

<https://www.mongodb.com/docs/v4.4/crud/>

1. Cloud MongoDB

<https://www.mongodb.com/cloud/atlas/register>

**Pastikan engine DB aktif**



**Pertemuan 8, 3 Oktober 2024**

**MongoDB Atlas**

Buat account

<https://learn.mongodb.com/>

Project

Cluster(0)

**Security:**

Networrk Access - (Daftar White list)

Database Access - (Manajemen pengguna)

**Data Cloud(Atlas), bisa diakses melalui MongoDB Compass**

mongodb+srv://mdp:**<db\_password>**@cluster0.n214x.mongodb.net/

Contoh :

mongodb+srv://mdp:**<db\_password>**@cluster0.n214x.mongodb.net/

Error :

* Connection String salah
* Password salah
* Komunikasi (Internet)
* Koneksi diblok oleh provider internet

Filtering Data, CRUD, dll

**https://www.mongodb.com/docs/mongodb-shell/crud/**

**>= $gte (greater than equal)**

**> $gt (greater than)**

**<= $ ???**

**< $ ???**

**db.movies.find( { rated: { $in: [ "PG", "PG-13" ] } } )**

**SELECT \* FROM movies WHERE rated in ("PG", "PG-13")**

**db.movies.find( { countries: "Mexico", "imdb.rating": { $gte: 7 } } )**

**Select \* from movies where countries='Mexico' and imdb.rating >=7**

**db.movies.find( {**

**year: 2010,**

**$or: [ { "awards.wins": { $gte: 5 } }, { genres: "Drama" } ]**

**} )**

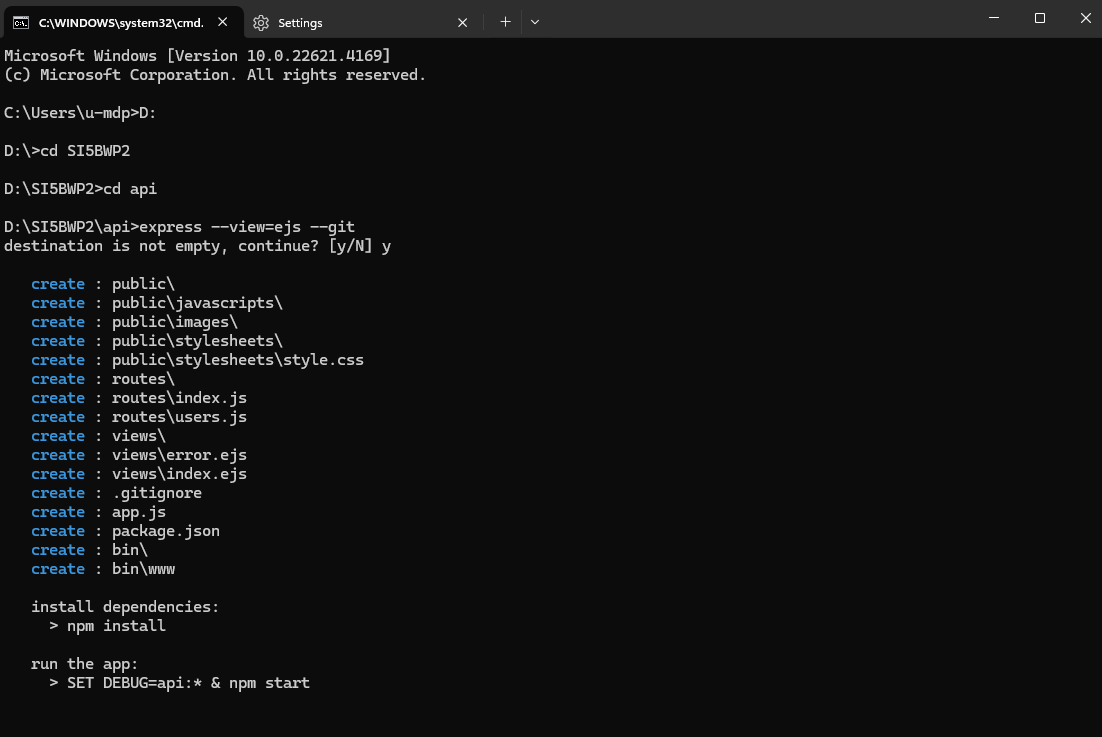
**Select \* from movies where year=2010 and**

**(awards.wins>=5 or genre='Drama')**

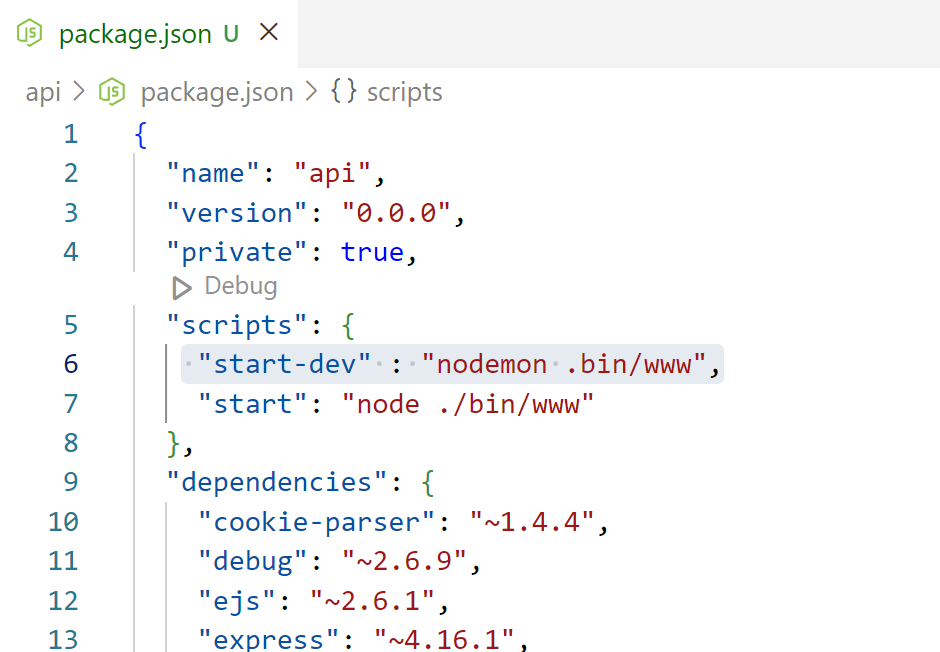
**Pertemuan 9, 08 Oktober 2024**

**API (Application Programming Interface)**

1. PS D:\SI5BWP2\api> npm init --y
2. Skeleton express-generator



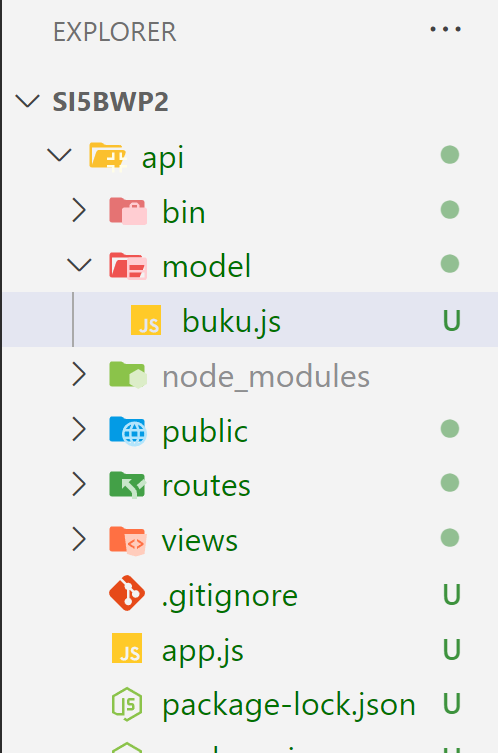
1. PS D:\SI5BWP2\api> npm install
2. Editing package.json



1. npm run start-dev
2. PS D:\SI5BWP2\api> npm install **mongoose** --save
3. App.Js

| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/index');  var usersRouter = require('./routes/users');  //CORS Enabled  //Cross Origin Resource Sharing  var app = express();  **app.use((req,res,next)=>{**  **res.setHeader("Access-Control-Allow-Origin","\*");**  **res.setHeader(**  **"Access-Control-Allow-Headers",**  **"Origin, X-Requested-With,Content-Type, Accept");**  **res.setHeader("Access-Control-Allow-Methods","GET, POST, PATCH, DELETE, OPTIONS");**  **next();**  **});**  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', indexRouter);  app.use('/users', usersRouter);  // catch 404 and forward to error handler  app.use(function(req, res, next) {  next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {  // set locals, only providing error in development  res.locals.message = err.message;  res.locals.error = req.app.get('env') === 'development' ? err : {};  // render the error page  res.status(err.status || 500);  res.render('error');  });  module.exports = app; |
| --- |

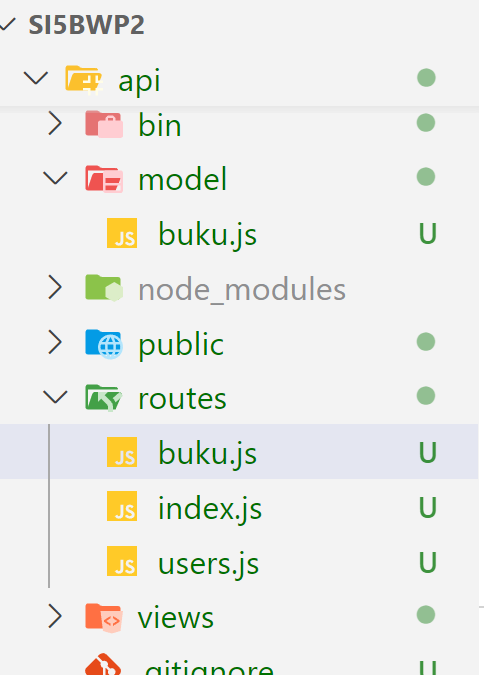
1. Tambahkan model/buku.js



1. Schema (Struktur dalam collection/table)

| const mongoose = require("mongoose");  const bukuSchema = new mongoose.Schema({  judul : {type : String, required : true},  penulis : {type : String, required : true},  genre : [{type : String, required : true}]  });  module.exports = mongoose.model("Buku", bukuSchema); |
| --- |

1. Routing : routes/buku.js



| var express = require('express');  var router = express.Router();  /\* GET users listing. \*/  router.get('/', function(req, res, next) {  res.send('respond dari BUKU router');  });  module.exports = router; |
| --- |

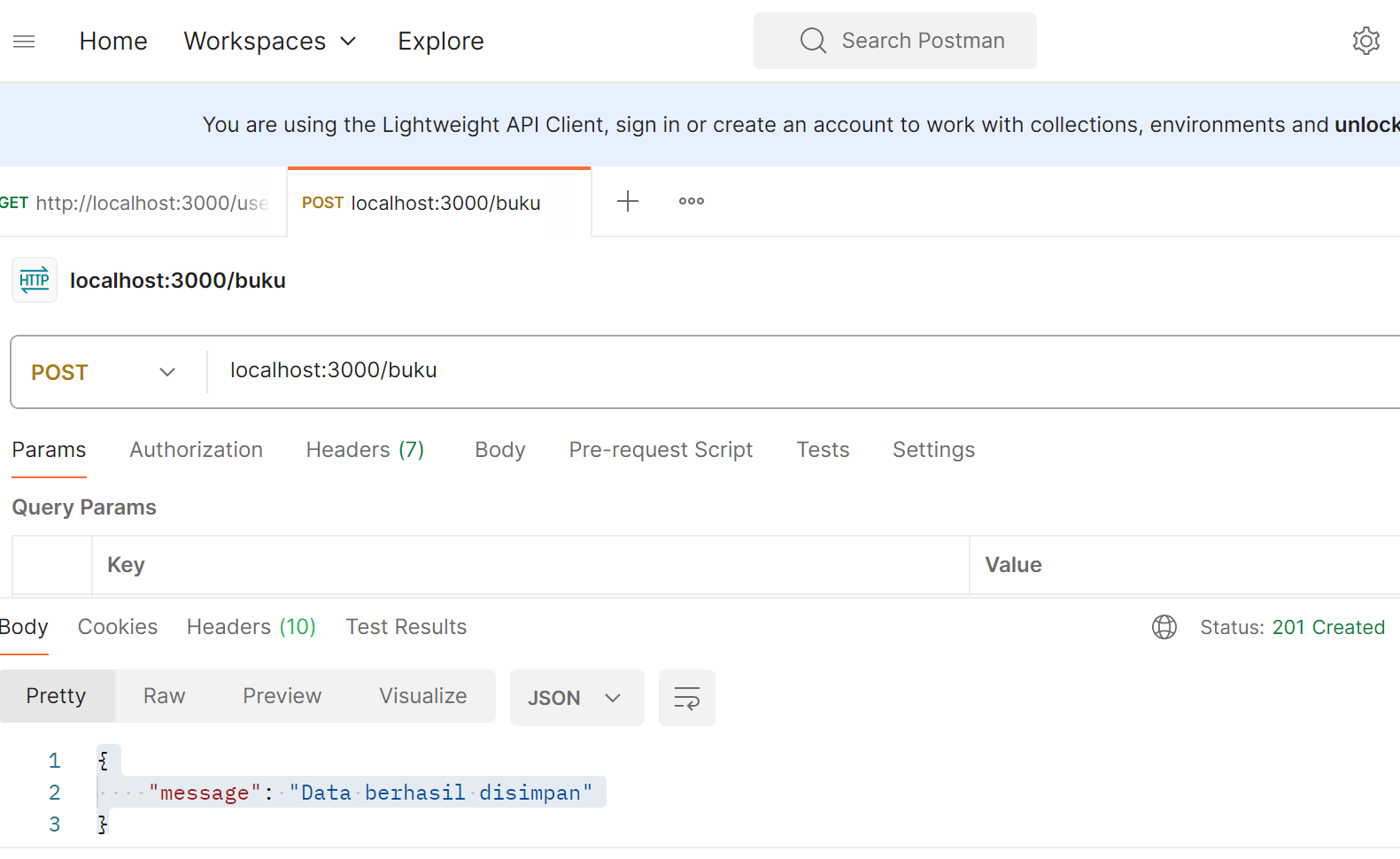
1. Editing buku.js

| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/index');  var usersRouter = require('./routes/users');  **var bukusRouter = require('./routes/bukus');**  //CORS Enabled  //Cross Origin Resource Sharing  var app = express();  app.use((req,res,next)=>{  res.setHeader("Access-Control-Allow-Origin","\*");  res.setHeader(  "Access-Control-Allow-Headers",  "Origin, X-Requested-With,Content-Type, Accept");  res.setHeader("Access-Control-Allow-Methods","GET, POST, PATCH, DELETE, OPTIONS");  next();  });  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', indexRouter);  app.use('/users', usersRouter);  **app.use('/buku', bukusRouter);**  // catch 404 and forward to error handler  app.use(function(req, res, next) {  next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {  // set locals, only providing error in development  res.locals.message = err.message;  res.locals.error = req.app.get('env') === 'development' ? err : {};  // render the error page  res.status(err.status || 500);  res.render('error');  });  module.exports = app; |
| --- |

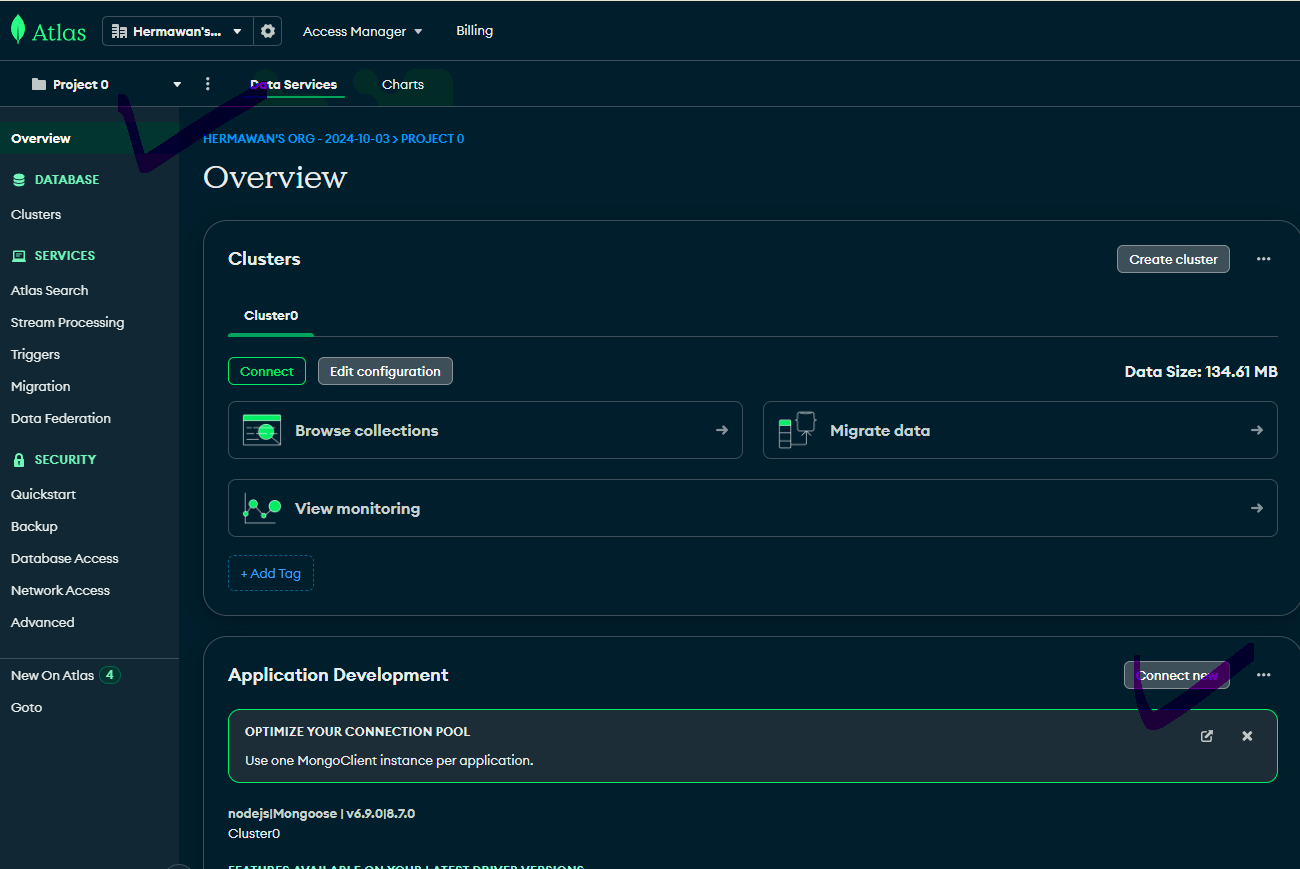
1. routes/bukus.js

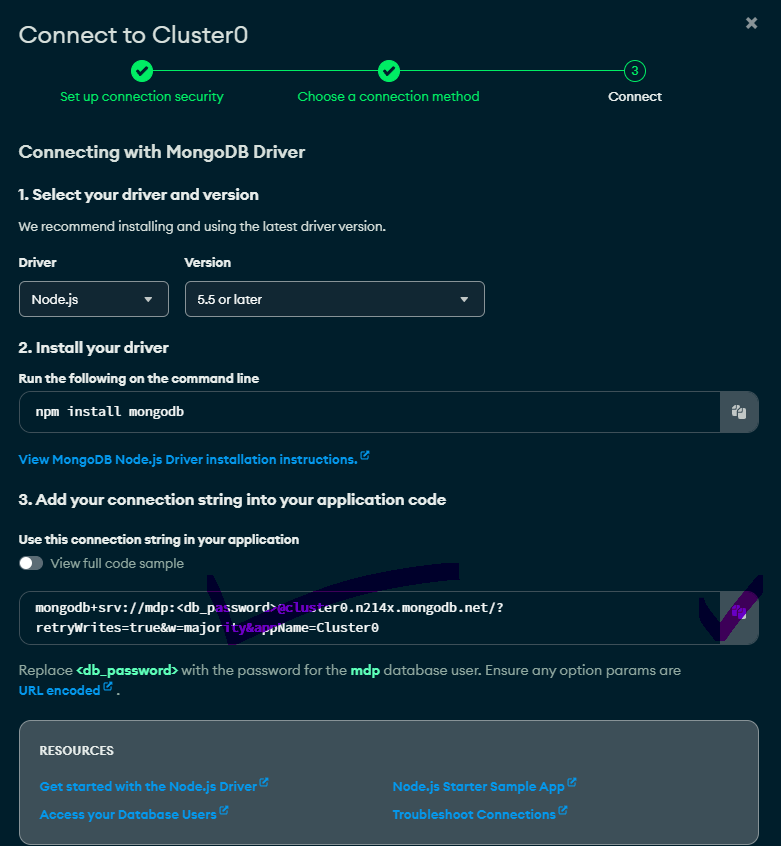
| var express = require('express');  var router = express.Router();  /\* GET users listing. \*/  // router.get('/', function(req, res, next) {  // res.send('respond dari BUKU router');  // });  //format JSON  **router.post('path', (req, res) => {**  **res.status(201).json({**  **message : "Data berhasil disimpan"**  **});**  **});**  module.exports = router; |
| --- |

1. Pengujian route dengan postman



1. Tentukan username dan password mongodb.com
2. Buatlah dbbaru dengan nama dbbuku
3. Copy Connection String :





1. Editing dengan menambahkan password dan namadatabase pada connection String
2. app.js

| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/index');  var usersRouter = require('./routes/users');  var bukusRouter = require('./routes/bukus');  const mongoose = require("mongoose");  //CORS Enabled  //Cross Origin Resource Sharing  var app = express();  app.use((req,res,next)=>{  res.setHeader("Access-Control-Allow-Origin","\*");  res.setHeader(  "Access-Control-Allow-Headers",  "Origin, X-Requested-With,Content-Type, Accept");  res.setHeader("Access-Control-Allow-Methods","GET, POST, PATCH, DELETE, OPTIONS");  next();  });  // mongoose.connect(  // "mongodb+srv://mdp:peSlThYtcsmwgvV8@cluster0.n214x.mongodb.net/dbbuku?retryWrites=true&w=majority&appName=Cluster0"  // ).then(()=>{  // console.log("Connected to Database");  // }).catch((err)=>{  // console.error('App starting error:', err.stack);  // console.log("Connection Failed");  // });  **mongoose.connect(**  **"mongodb://localhost:27017/dbbuku"**  **).then(()=>{**  **console.log("Connected to Database");**  **}).catch((err)=>{**  **// console.error('App starting error:', err.stack);**  **console.log("Connection Failed");**  **});**  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', indexRouter);  app.use('/users', usersRouter);  app.use('/buku', bukusRouter);  // catch 404 and forward to error handler  app.use(function(req, res, next) {  next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {  // set locals, only providing error in development  res.locals.message = err.message;  res.locals.error = req.app.get('env') === 'development' ? err : {};  // render the error page  res.status(err.status || 500);  res.render('error');  });  module.exports = app; |
| --- |

1. Bukus.js

| var express = require('express');  var router = express.Router();  const Buku = require("../model/buku");  /\* GET users listing. \*/  // router.get('/', function(req, res, next) {  // res.send('respond dari BUKU router');  // });  //format JSON  router.post('/', (req, res) => {  const buku = new Buku({  judul : req.body.judul,  penulis : req.body.penulis,  genre : req.body.genre  });  console.log(buku);  buku.save().then((createdBuku)=>{  res.status(201).json({  message : "Data berhasil disimpan",  bookId : createdBuku.\_id  });  });    });  module.exports = router; |
| --- |

**Pertemuan 11, 15 Oktober 2024**

**API (Application Programming Interface)**

**READ**

1. Akses Read pada routes\bukus :

| **router.get("/",(req, res)=>{**  **Buku.find()**  **.then((documents)=>{**  **res.status(201).json({**  **message : "Get Data Buku",**  **bukus : documents**  **});**  **});**  **});** |
| --- |

1. Akses Delete pada routes\bukus

| **router.delete('/:id', (req, res) => {**  **Buku.deleteOne({\_id : req.params.id})**  **.then((result)=>{**  **res.status(200).json({**  **message : "Buku berhasil dihapus ",**  **result : result**  **});**  **});**  **});** |
| --- |

1. Akses update pada routes\buku

| **router.put('/:id', (req, res) => {**    **const buku = new Buku({**  **\_id : req.params.id,**  **judul : req.body.judul,**  **penulis : req.body.penulis,**  **genre : req.body.genre**  **});**  **Buku.updateOne({\_id : req.params.id}, buku)**  **.then((hasil)=>{**  **res.status(200).json({**  **message : "Update Berhasil",**  **result : hasil**  **});**  **});**  **});** |
| --- |

**Pertemuan 12, 17 Oktober 2024**

**Controller dan Error handling**

1. Tambahkan folder baru controller dan file : buku.js di level folder paling atas
2. Editing file controller\buku.js

| const Buku = require("../model/buku");  const createBuku= (req, res) => {  const buku = new Buku({  judul : req.body.judul,  penulis : req.body.penulis,  genre : req.body.genre  });  //console.log(buku);  buku.save().then((createdBuku)=>{  res.status(201).json({  message : "Data berhasil disimpan",  bookId : createdBuku.\_id  });  });    };  const readBuku = (req, res)=>{  Buku.find()  .then((documents)=>{  res.status(201).json({  message : "Get Data Buku",  bukus : documents  });  });  };  const deleteBuku= (req, res) => {  Buku.deleteOne({\_id : req.params.id})  .then((result)=>{  res.status(200).json({  message : "Buku berhasil dihapus ",  result : result  });  });  };  const updateBuku = (req, res) => {    const buku = new Buku({  \_id : req.params.id,  judul : req.body.judul,  penulis : req.body.penulis,  genre : req.body.genre  });  Buku.updateOne({\_id : req.params.id}, buku)  .then((hasil)=>{  res.status(200).json({  message : "Update Berhasil",  result : hasil  });  });  };  module.exports ={createBuku, readBuku, deleteBuku, updateBuku} |
| --- |

1. Editing routes\buku.js

| var express = require('express');  var router = express.Router();  **const BukuController = require("../controller/buku")**  /\* GET users listing. \*/  // router.get('/', function(req, res, next) {  // res.send('respond dari BUKU router');  // });  //insert  **router.post('/',BukuController.createBuku);**  **//select**  **router.get("/",BukuController.readBuku);**  **//delete**  **router.delete('/:id',BukuController.deleteBuku);**  **//update**  **router.put('/:id',BukuController.updateBuku);**  module.exports = router; |
| --- |

1. Error handling pada request controller\buku.js

| const Buku = require("../model/buku");  const createBuku= (req, res) => {  const buku = new Buku({  judul : req.body.judul,  penulis : req.body.penulis,  genre : req.body.genre  });  //console.log(buku);  buku.save()  .then((createdBuku)=>{  res.status(201).json({  message : "Data berhasil disimpan",  bookId : createdBuku.\_id  });  })  **.catch((err)=>{**  **//console.log(err);**  **res.status(500).json({**  **message : "internal server error !"**  **//error : err**  **});**  **});**    };  const readBuku = (req, res)=>{  Buku.find()  .then((documents)=>{  res.status(201).json({  message : "Get Data Buku",  bukus : documents  });  })  **.catch((err)=>{**  **//console.log(err);**  **res.status(500).json({**  **message : "internal server error !"**  **//error : err**  **});**  **});**  };  const deleteBuku= (req, res) => {  Buku.deleteOne({\_id : req.params.id})  .then((result)=>{  res.status(200).json({  message : "Buku berhasil dihapus ",  result : result  });  })  **.catch((err)=>{**  **//console.log(err);**  **res.status(500).json({**  **message : "internal server error !"**  **//error : err**  **});**  **});**  };  const updateBuku = (req, res) => {    const buku = new Buku({  \_id : req.params.id,  judul : req.body.judul,  penulis : req.body.penulis,  genre : req.body.genre  });  Buku.updateOne({\_id : req.params.id}, buku)  .then((hasil)=>{  res.status(200).json({  message : "Update Berhasil",  result : hasil  });  }).**catch((err)=>{**  **//console.log(err);**  **res.status(500).json({**  **message : "internal server error !"**  **//error : err**  **});**  **})**;;  };  module.exports ={createBuku, readBuku, deleteBuku, updateBuku} |
| --- |

Praktikum 2 :

1. Install Library mongoose unique validator

npm install --save mongoose-unique-validator@5.0.0

the unique validator will check for duplicate database entries and report them just like any other validation error

1. Di model\user.js

| **const mongoose = require("mongoose");**  **const uniqueValidator = require("mongoose-unique-validator");**  **const userSchema = new mongoose.Schema({**  **email : {type : String, required : true, unique : true},**  **password : {type : String, required:true}**  **});**  **userSchema.plugin(uniqueValidator);**  **module.exports = mongoose.model("User",userSchema);** |
| --- |

1. npm install --save mongoose-unique-validator@5.0.0

**Pertemuan 13, 24 Oktober 2024**

**Users**

1. controller\user.js

| const User = require("../model/user");  const signUp= (req, res)=>{  const user = new User({  email : req.body.email,  password : req.body.password  });  user.save()  .then((result)=>{  res.status(202).json({  message : "User Created",  // result : result  });  })  **.catch((err)=>{**  **res.status(501).json({**  **message : "Internal Server Error",**  **error : err**  **});**  **});**    };  module.exports = { signUp} |
| --- |

1. routes\users.js

| var express = require('express');  var router = express.Router();  const UserController = require("../controller/user");  /\* GET users listing. \*/  // router.get('/', function(req, res, next) {  // res.send('respond with a resource');  // });  router.post('/',UserController.signUp);  module.exports = router; |
| --- |

1. **npm install --save bcrypt**

<https://github.com/hermawanumdp/SI5BWP2/tree/main/api>

1. **Controller\user.js**

| const User = require("../model/user");  const bcrypt = require("bcrypt");  const signUp= (req, res)=>{  bcrypt.hash(req.body.password,10)  .then((hash)=>{  const user = new User({  email : req.body.email,  password : hash  });  user.save()  .then((result)=>{  res.status(202).json({  message : "User Created",  // result : result  });  })  .catch((err)=>{  res.status(501).json({  message : "Internal Server Error",  // error : err  });  });  });      };  const login = (req, res)=>{  let fetchedUser;  User.findOne({ email : req.body.email})  .then((user)=>{  if(!user){  return res.status(401).json({  message : "Auth failed, email not exists !"  });  }  fetchedUser= user;  return bcrypt.compare(req.body.password, user.password);  });  }  module.exports = { signUp, login} |
| --- |