



**polines**  
politeknik negeri semarang  
*committed to quality*

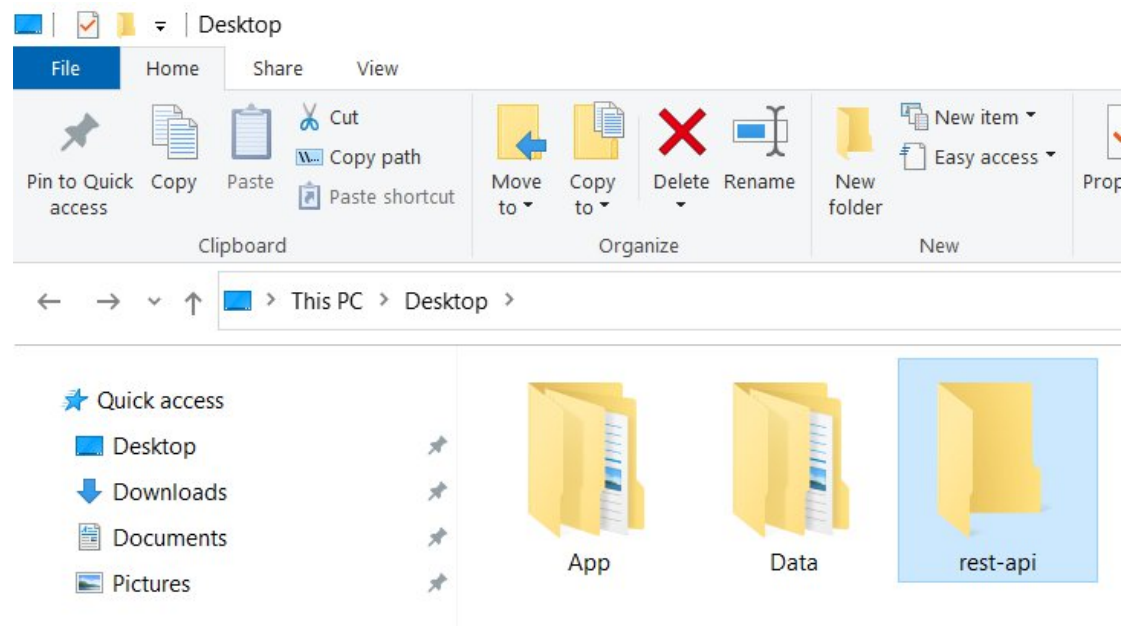


# PCC CLASS

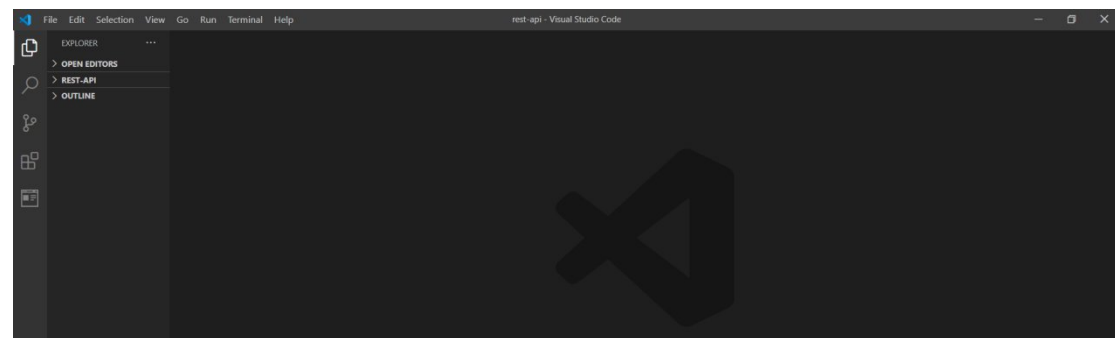
## Membuat RESTful API dengan Node.js dan MongoDB



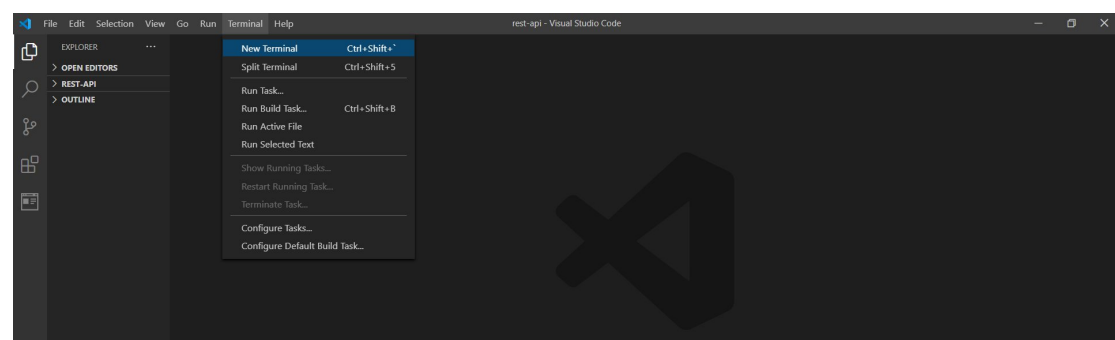
1. Buat Folder Baru Dengan Nama rest-api



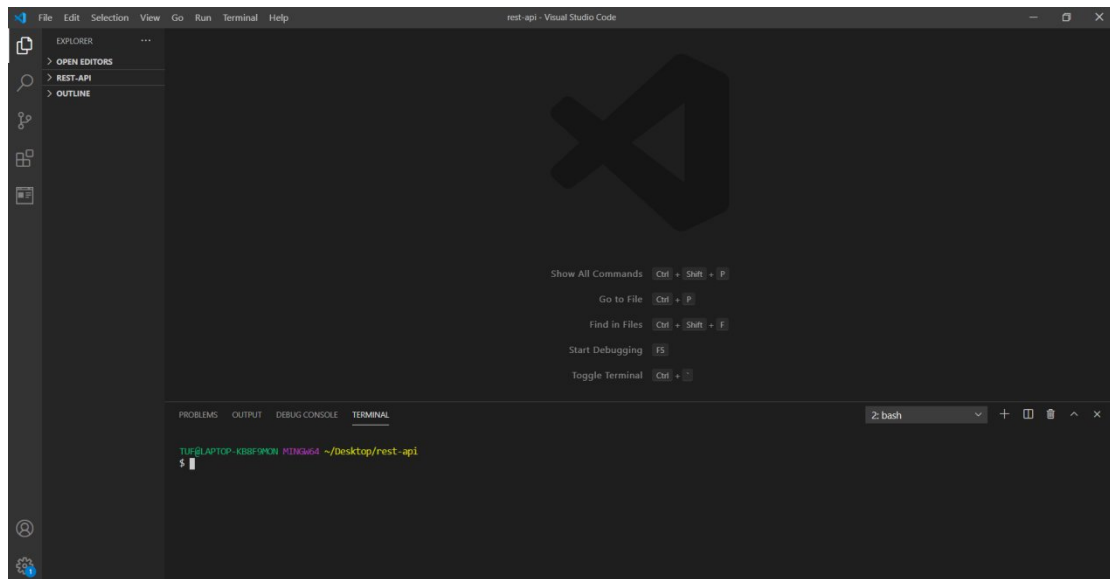
2. Buka Folder rest-api di visual studio code



3. Buka terminal dengan cara ctrl+shift+`

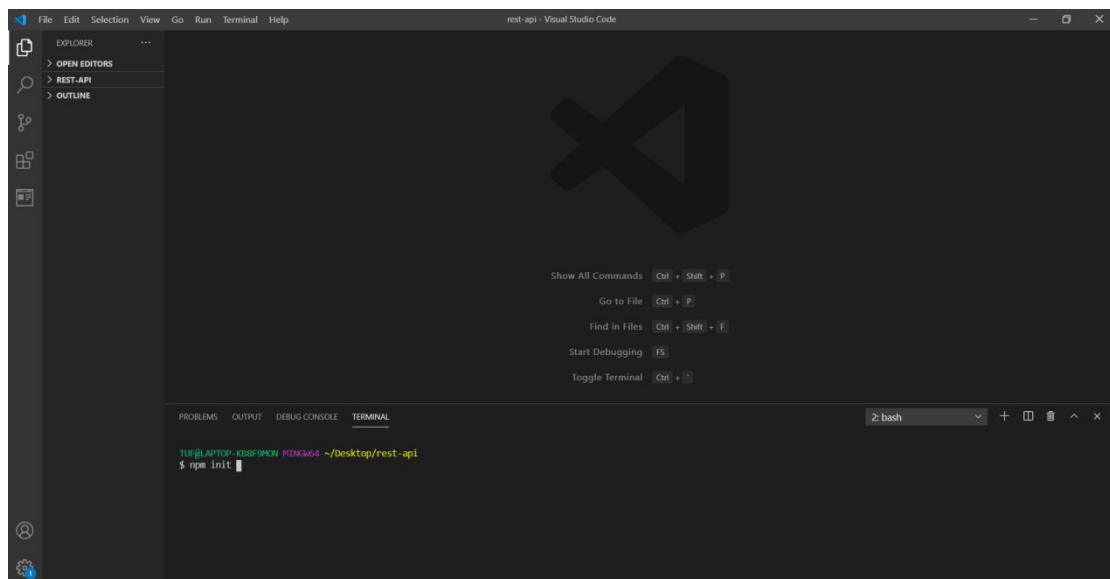


Hasil

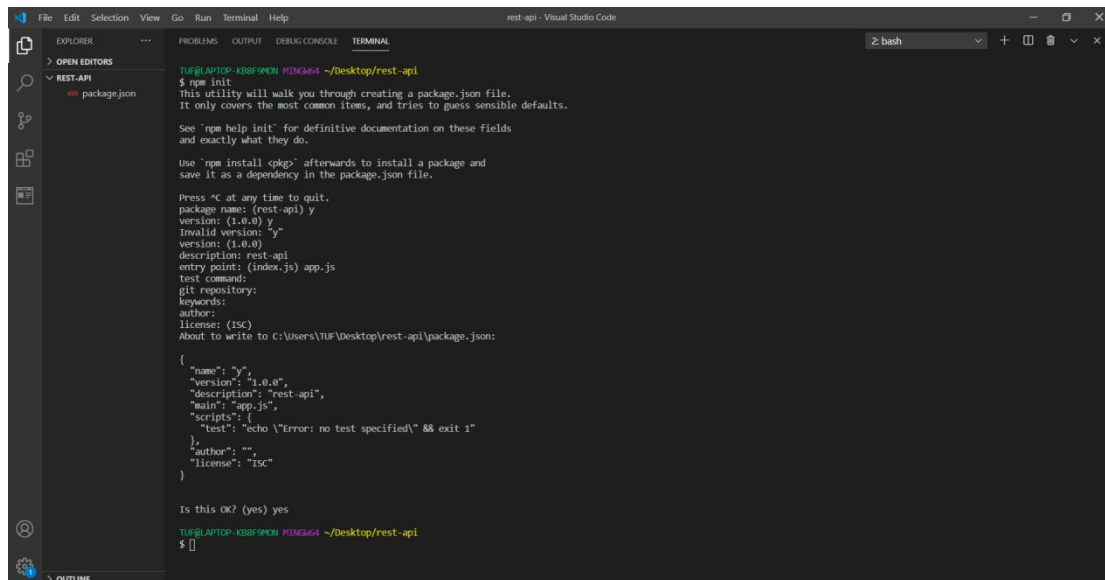


4. Ketik ***npm init*** pada terminal visual studio code

Npm init digunakan untuk membuat file ***package.json***



Hasil



```
rest-api - Visual Studio Code
TUF@LAPTOP-KB8F5WCH MINGW64 ~/Desktop/rest-api
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

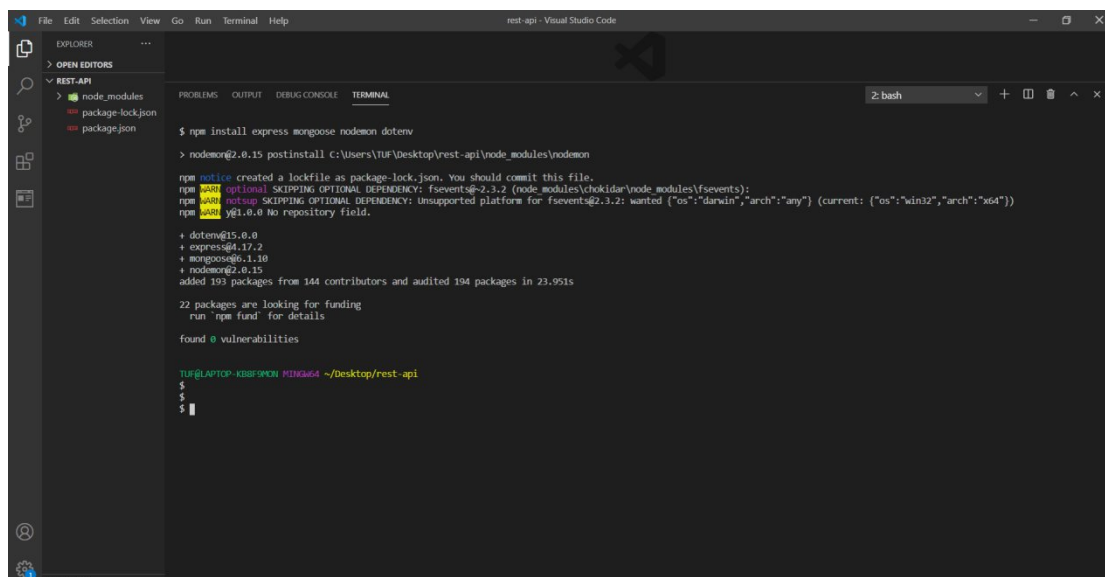
See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (rest-api) y
version: (1.0.0) y
invalid version: "y"
version: (1.0.0)
description: rest-api
entry point: (index.js) app.js
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\TUF\Desktop\rest-api\package.json:
{
  "name": "y",
  "version": "1.0.0",
  "description": "rest-api",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
TUF@LAPTOP-KB8F5WCH MINGW64 ~/Desktop/rest-api
$
```

5. Selanjutnya install beberapa dependencies yang akan digunakan dengan cara mengetikkan ***npm install express mongoose nodemon dotenv***



```
rest-api - Visual Studio Code
TUF@LAPTOP-KB8F5WCH MINGW64 ~/Desktop/rest-api
$ npm install express mongoose nodemon dotenv

> nodemon@2.0.15 postinstall C:\Users\TUF\Desktop\rest-api\node_modules\nodemon
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os": "darwin", "arch": "any"} (current: {"os": "win32", "arch": "x64"})
npm WARN @1.0.0 No repository field.

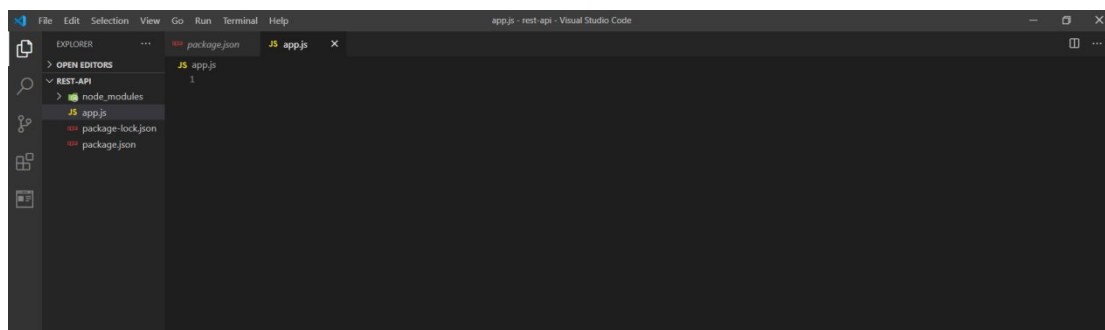
+ dotenv@15.4.0
+ express@4.17.2
+ mongoose@6.1.10
+ nodemon@2.0.15
added 193 packages from 144 contributors and audited 194 packages in 23.951s

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

found 0 vulnerabilities

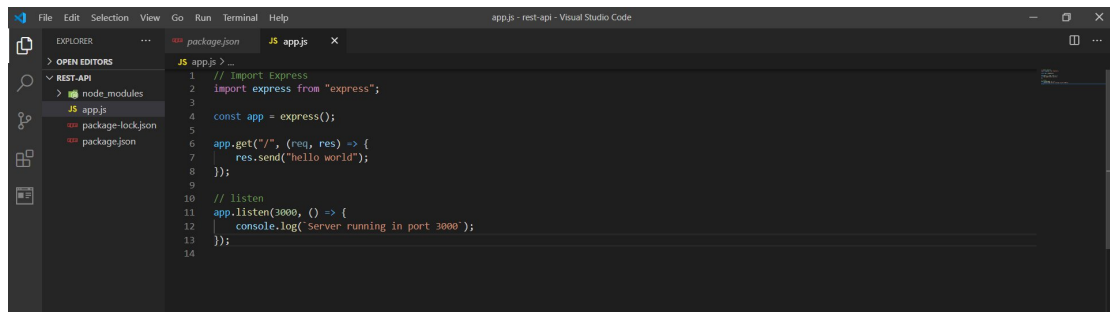
TUF@LAPTOP-KB8F5WCH MINGW64 ~/Desktop/rest-api
$
$
$
```

6. Buat file bernama app.js

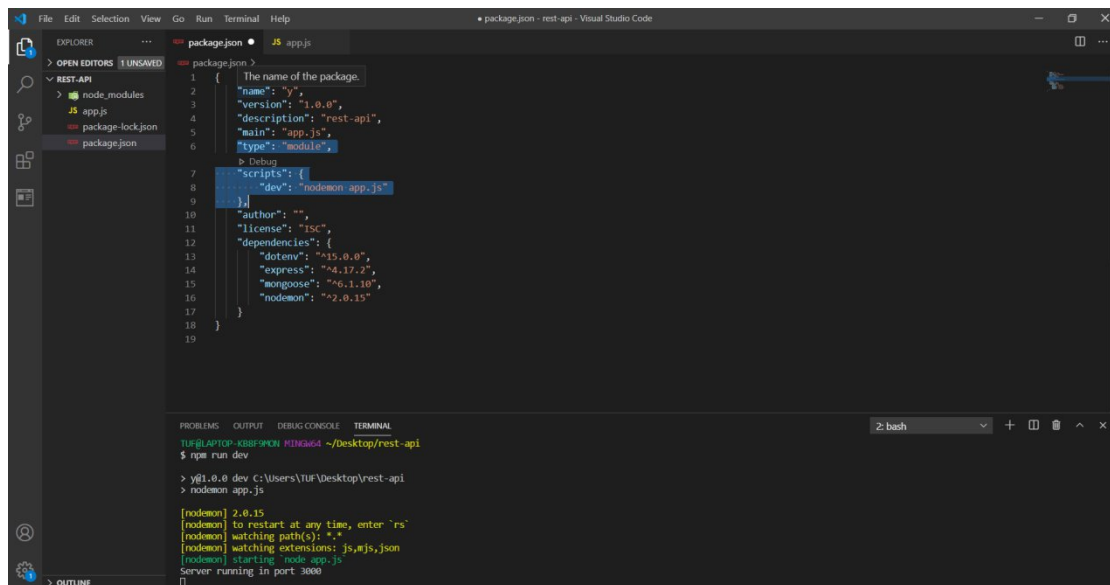


```
rest-api - Visual Studio Code
EXPLORER
  REST-API
    package-lock.json
    package.json
    node_modules
      JS app.js
      package-lock.json
      package.json
```

7. kemudian tambahkan main file dari ***express.js*** dan menambahkan satu route untuk mencoba di browser



8. Selanjutnya konfigurasi file package.json dengan menambahkan source code berikut dan jalankan servernya dengan mengetikkan perintah ***npm run dev***



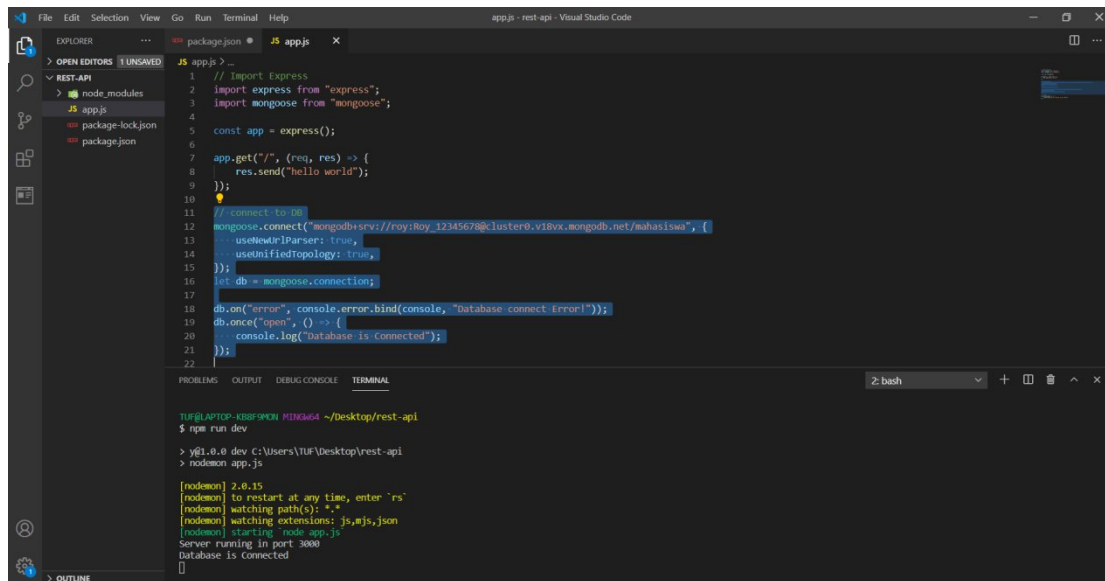
9. Kemudian buka browser dan ketik url localhost:3000 untuk mengecek konfigurasi route yang sudah dibuat



10. Import package mongoose dengan menggunakan perintah

```
import mongoose from "mongoose";
```

11. Koneksikan express ke database mongodb dengan mengetikkan perintah berikut



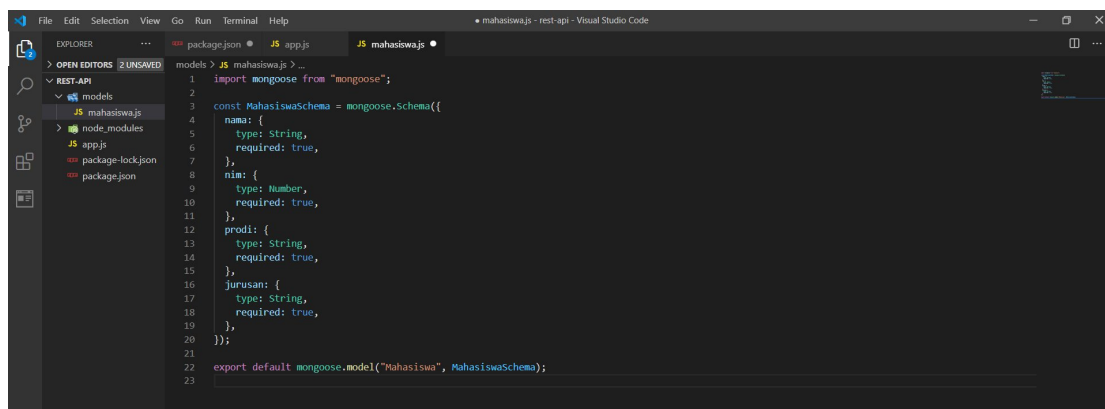
```
1 // Import Express
2 import express from "express";
3 import mongoose from "mongoose";
4
5 const app = express();
6
7 app.get("/", (req, res) => {
8   res.send("hello world");
9 });
10
11 // connect to db
12 mongoose.connect("mongodb+srv://roy:Roy_12345678@cluster0.v18vx.mongodb.net/mahasiswa", {
13   useNewUrlParser: true,
14   useUnifiedTopology: true,
15 });
16 const db = mongoose.connection;
17
18 db.on("error", console.error.bind(console, "Database connect Error!"));
19 db.once("open", () => {
20   console.log("Database is Connected");
21 });
22
```

```
TUF@LAPTOP-KB8F5MCH MINKAW4 ~/Desktop/rest-api
$ npm run dev

> y@1.0.0 dev C:\Users\TUF\Desktop\rest-api
> nodemon app.js

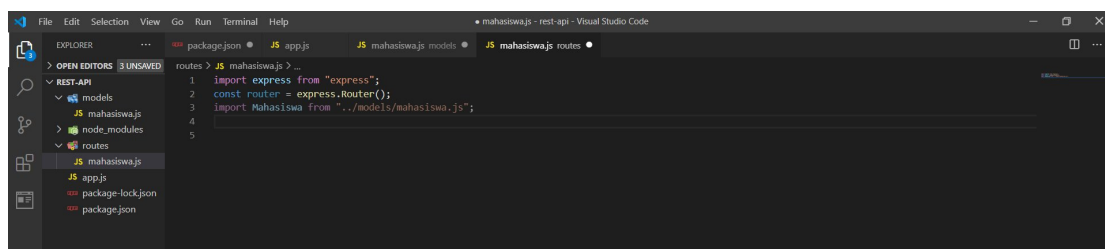
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting node app.js
Server running in port 3000
Database is Connected
```

12. Buat folder **models** kemudian buat file **mahasiswa.js** untuk membuat schema database pada mongodb lalu ketikkan script berikut



```
1 import mongoose from "mongoose";
2
3 const MahasiswaSchema = mongoose.Schema({
4   nama: {
5     type: String,
6     required: true,
7   },
8   nim: {
9     type: Number,
10    required: true,
11  },
12  prodi: {
13    type: String,
14    required: true,
15  },
16  jurusan: {
17    type: String,
18    required: true,
19  },
20 });
21
22 export default mongoose.model("Mahasiswa", MahasiswaSchema);
23
```

13. Buat folder bernama **routes** kemudian buat file **mahasiswa.js** folder routes ini yang nantinya akan diakses sebagai rest-api didalam project ini



```
1 import express from "express";
2 const router = express.Router();
3 import Mahasiswa from "../models/mahasiswa.js";
4
5
```

14. Selanjutnya tambahkan perintah create ke dalam source code **mahasiswa.js**

```
24
25 // CREATE
26 router.post("/", async (req, res) => {
27   const reqMhs = new Mahasiswa({
28     nama: req.body.nama,
29     nim: req.body.nim,
30     prodi: req.body.prodi,
31     jurusan: req.body.jurusan,
32   });
33
34   try {
35     const mahasiswa = await reqMhs.save();
36     res.json(mahasiswa);
37   } catch (err) {
38     res.json({ message: err });
39   }
40 });
41
42 export default router;
43
```

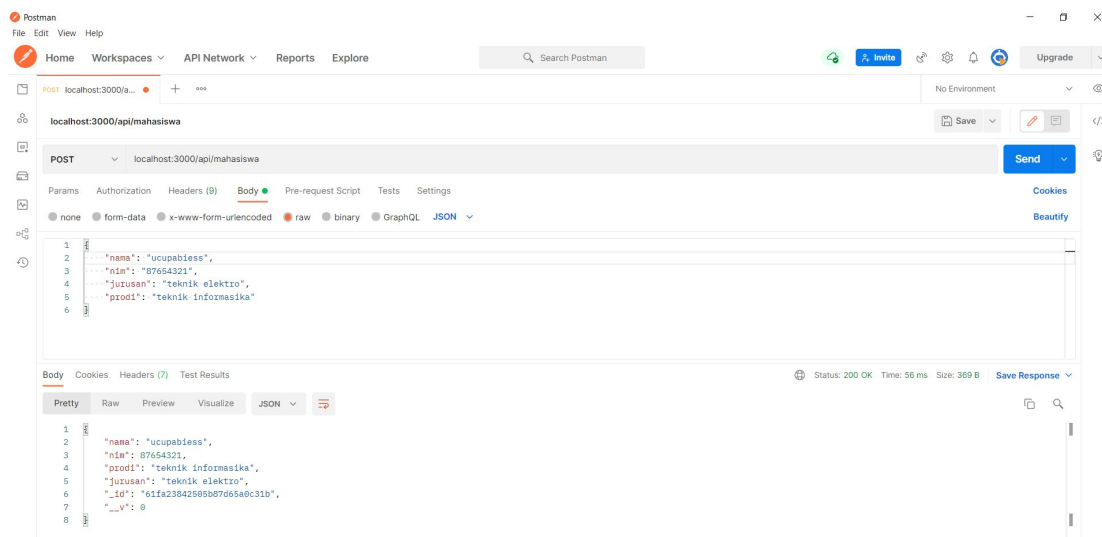
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Database is connected

15. Import package bodyparser dan mahasiswaRoutes dan tambahkan script middleware di **app.js**

```
5 import mahasiswaRoutes from "../routes/mahasiswa.js";
6 import bodyParser from "body-parser";
7
8 // Middleware
9 app.use(bodyParser());
10
11 // routes example
12 app.use("/api/mahasiswa", mahasiswaRoutes);
13
```

16. Kemudian uji coba create tersebut menggunakan postman



17. Selanjutnya kita akan menampilkan data dari database

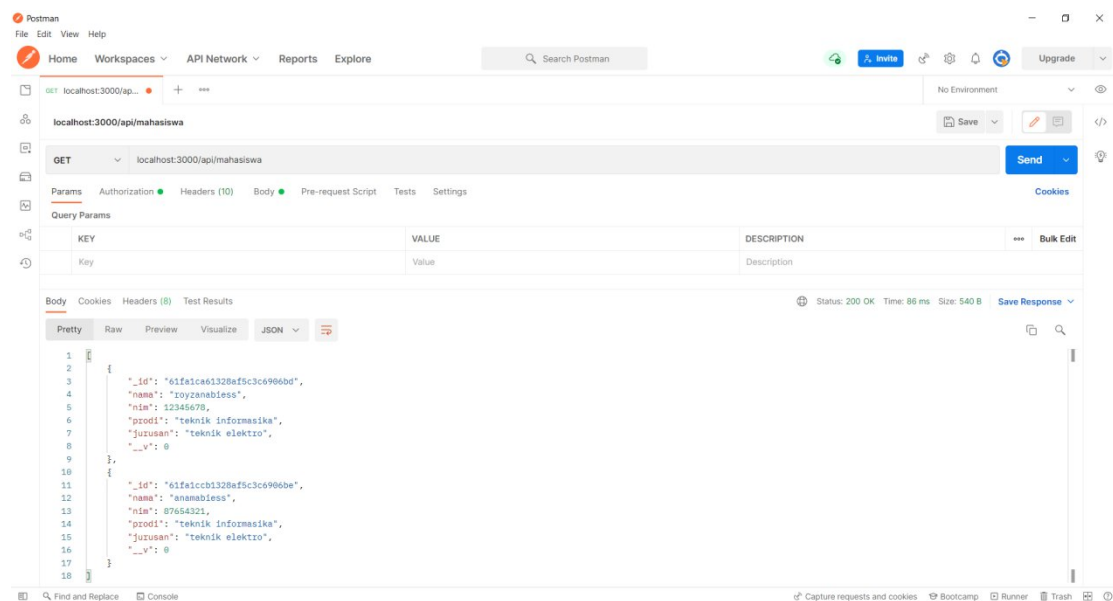
```

//GET DATA MAHASISWA
router.get("/", async (req, res) => {
  try {
    const mahasiswa = await Mahasiswa.find();
    res.json(mahasiswa);
  } catch (err) {
    res.json({ message: err });
  }
});

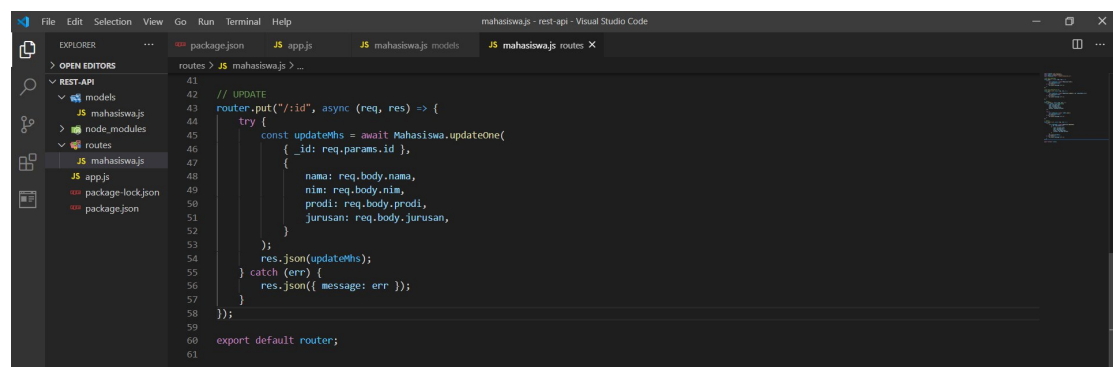
//GET DATA MAHASISWA BY ID
router.get("/:id", async (req, res) => {
  try {
    const mahasiswa = await Mahasiswa.findOne({ _id: req.params.id });
    res.json(mahasiswa);
  } catch (err) {
    res.json({ message: err });
  }
});

```

18. Lakukan uji coba rest-api menampilkan data

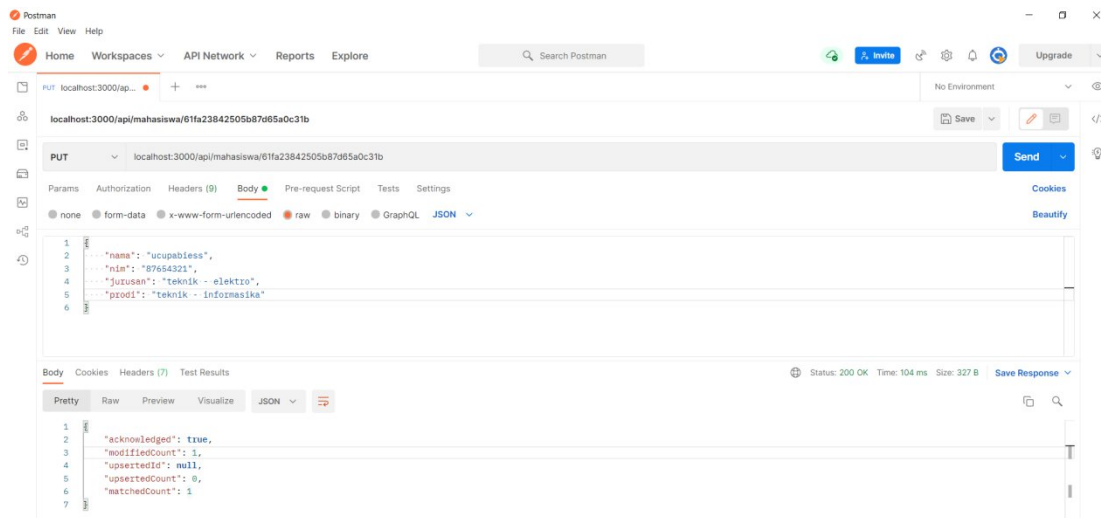


19. Tambahkan perintah update ke dalam source code ***mahasiswa.js*** seperti berikut

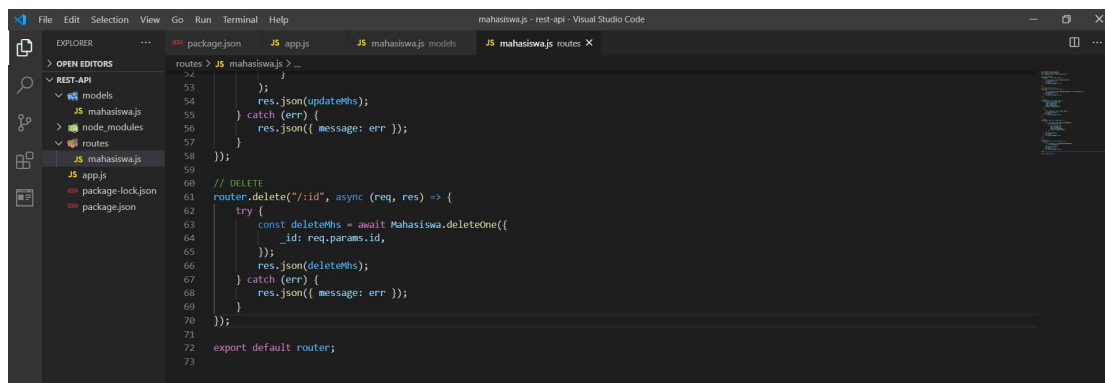


20. Lalu uji coba perintah yang telah dibuat menggunakan postman, tambahkan parameter berupa id mahasiswa kemudian send untuk melihat hasilnya





21. Selanjutnya tambahkan perintah delete ke dalam source code *mahasiswa.js*



22. Lalu uji coba menggunakan postman, tambahkan parameter berupa id mahasiswa kemudian send untuk melihat hasilnya

