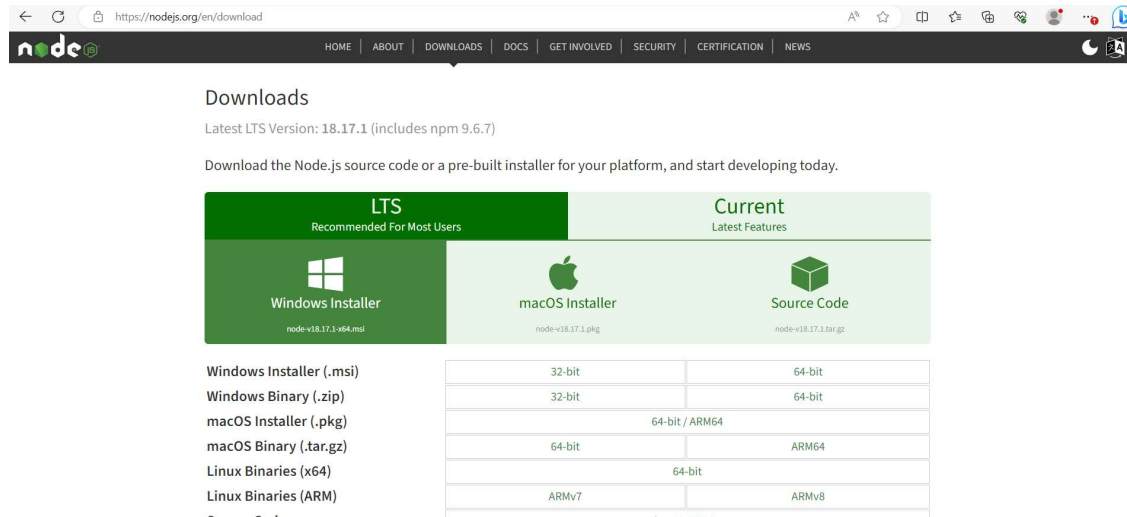


Backend ExpressJs

1. Pastikan sudah menginstall nodes.js

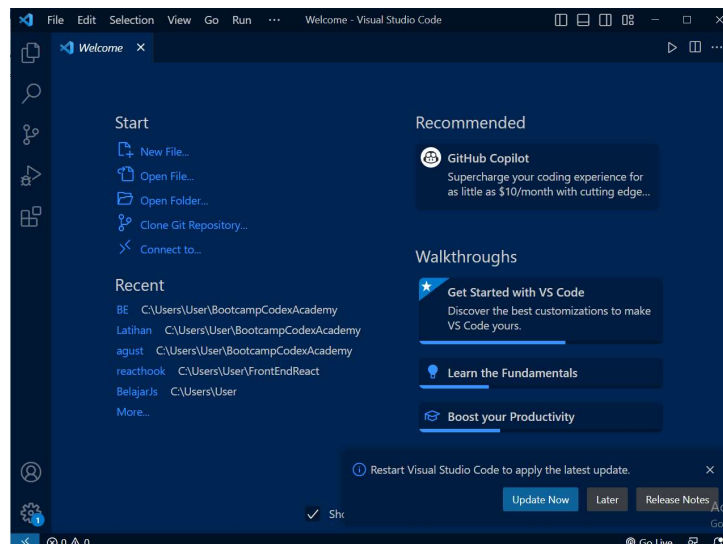


2. Cek apakah instalasi sudah dilakukan.

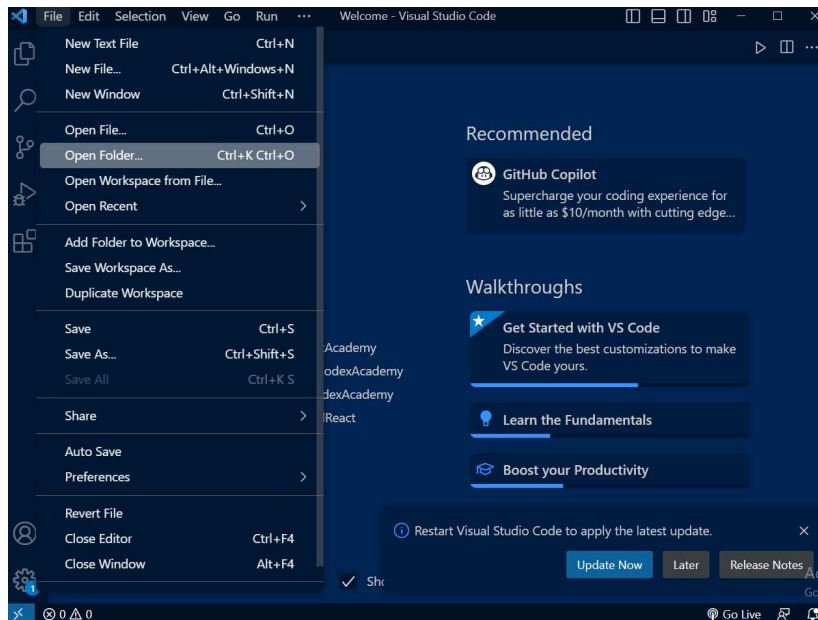
```
C:\Users\User>node -v
v18.12.1
```

Jika sudah tampil maka sudah berhasil terinstall.

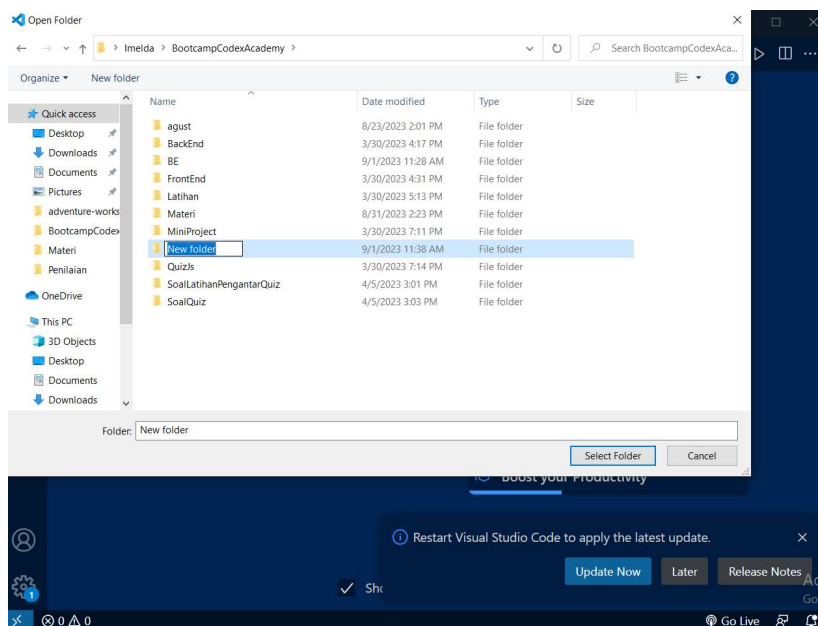
3. Buka editor misalnya visual studio code untuk membuat backend menggunakan ExpressJs



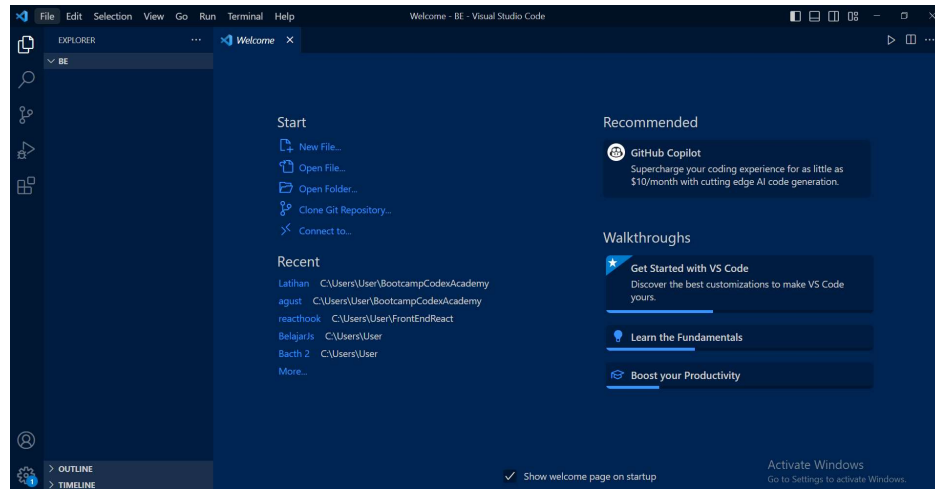
- a.
- b. Buat folder kerja



- c. Pilih direktori dimana buat folder baru, lalu klik new folder atau klik kanan lalu pilih new folder, berikan nama folder untuk menyimpan backend yang dibuat.



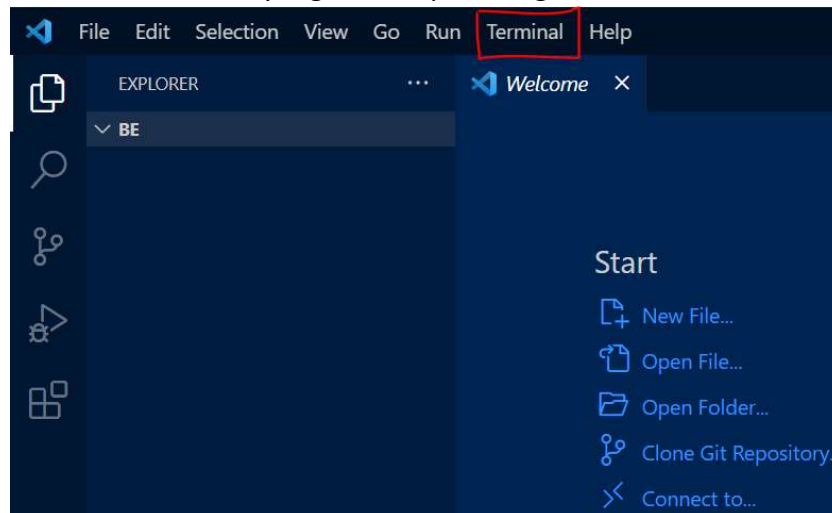
- d. Sekarang visual studio code berada pada direktori yang baru buat.



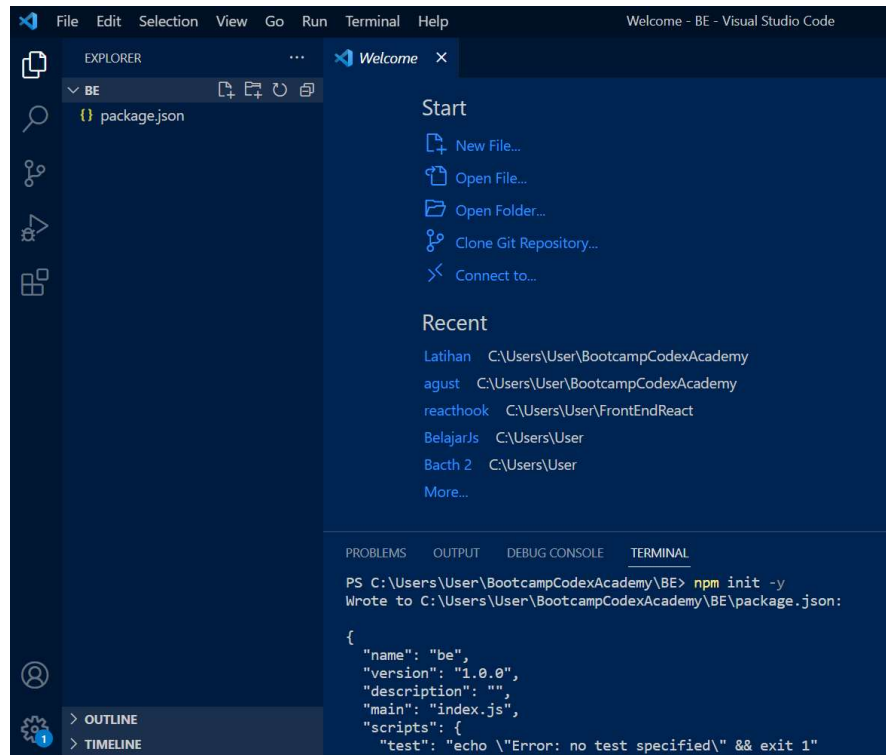
4. Membuat Backend menggunakan library expressjs

- Tetap berada pada direktori kerja tampilkan terminal untuk membuat file package.json.

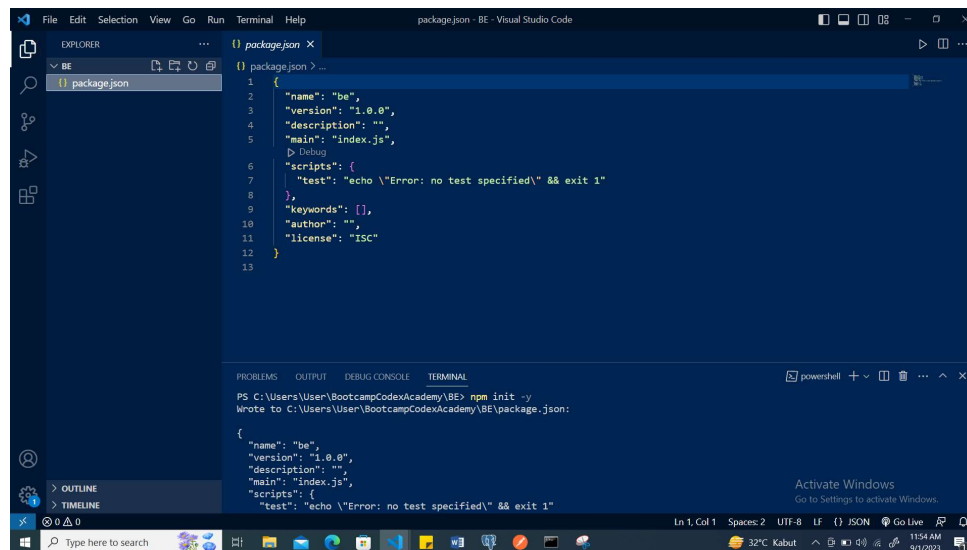
Klik menu terminal yang berada pada bagian atas:



- Pada terminal ketikkan: `npm init` atau `nmp init -y` jika ingin secara default semua pertanyaan akan diberi nilai yes, lalu tekan enter.

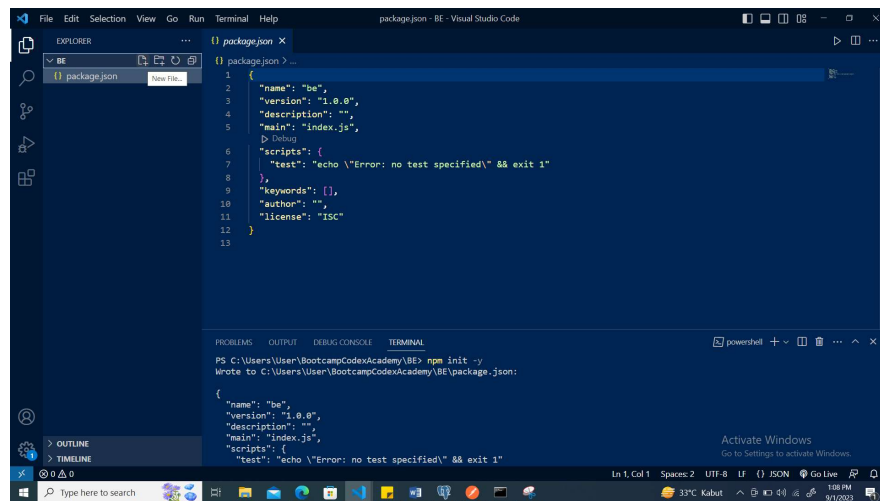


Atas perintah tadi akan terbentuk file `package.json`. File ini berisi deskripsi mengenai backend yang dibangun termasuk daftar library-library yang digunakan. File ini bisa diubah kemudian informasi didalamnya sesuai kebutuhan.



C.

5. Membuat file yang akan melayani setiap request dari client. Misalnya Namanya `server.js`
 - a. Pastikan folder direktori kerja terselect lalu pilih icon buat file baru.

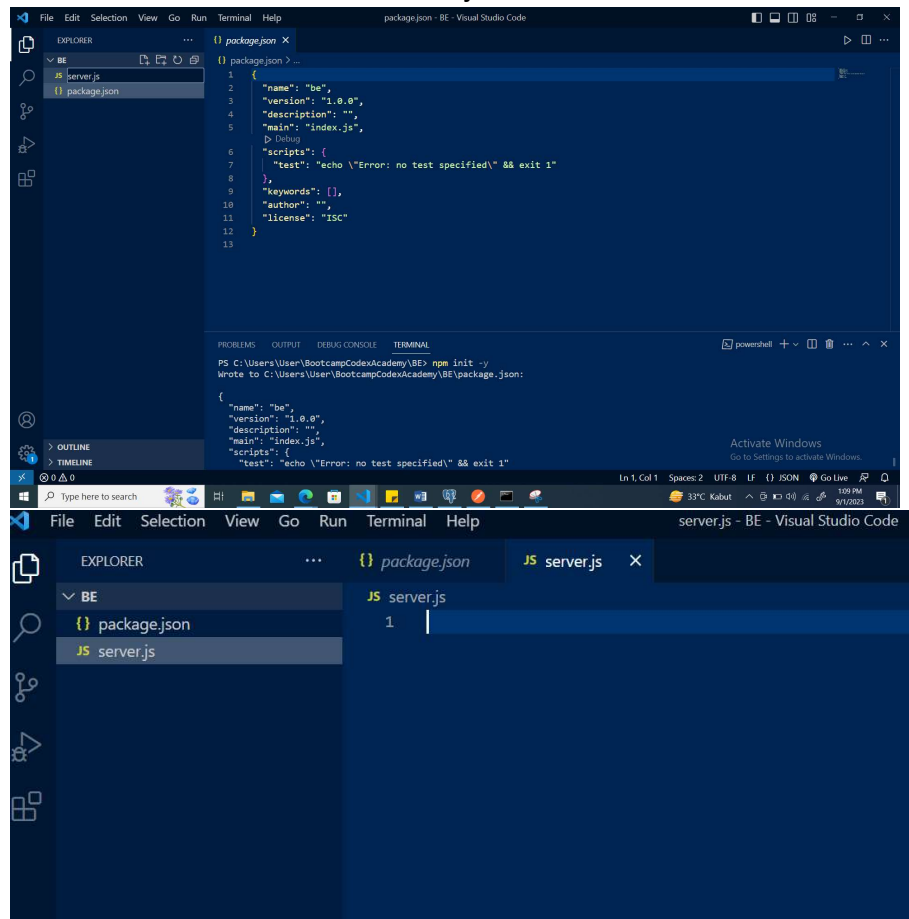


The screenshot shows the Visual Studio Code interface with the `package.json` file open. The file contains the following JSON:

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

The terminal at the bottom shows the output of the `npm init -y` command, which has written the `package.json` file to the current directory.

Setelah klik new file, ketikkan nama file `server.js` lalu enter maka selanjutnya akan masuk ke halaman `server.js`



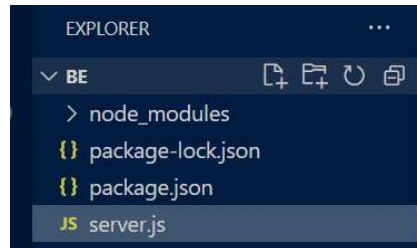
The top screenshot shows the Explorer view on the left with a new file named `server.js` created. The bottom screenshot shows the `server.js` file open in the editor, with the cursor at the first line.

b. Install library `express` dengan perintah

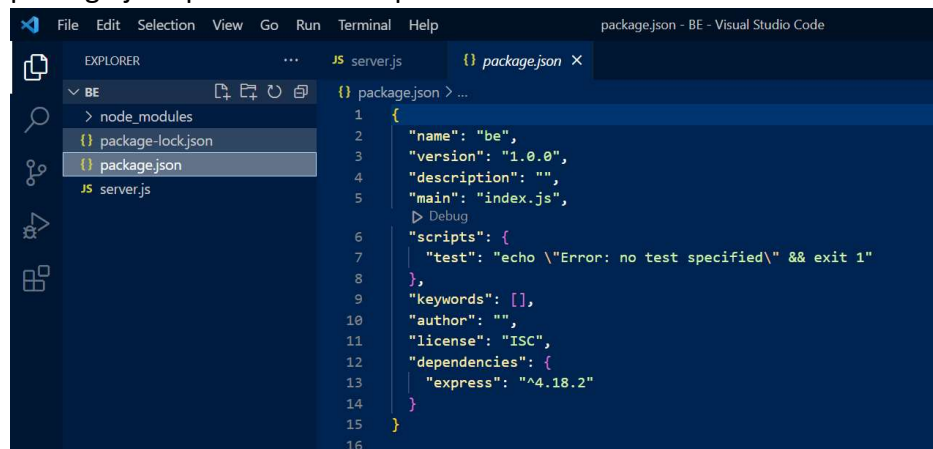
```
PS C:\Users\User\BootcampCodexAcademy\BE> npm i express
```

, pada saat install library pertama sekali maka akan terbentuk folder `node_modules` sebagai lokasi library yang

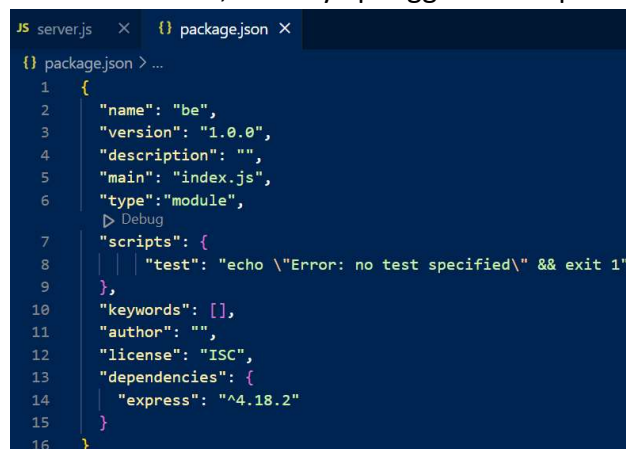
diinstall dan file package-lock.json yg merupakan file yang berisi informasi penginstallan library



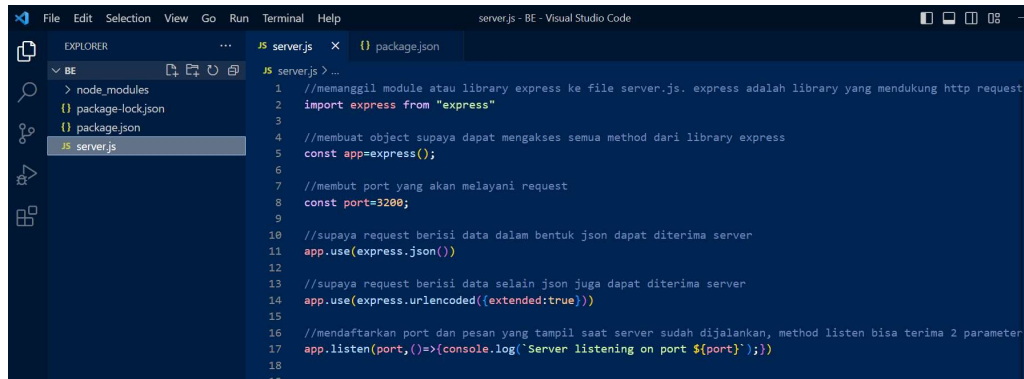
- c. Setelah berhasil install maka library yang baru diinstall akan terdaftar di file package.json pada elemen dependencies.



- d. Sebelum isi file server.js tambahkan elemen pada file package.json: `“type”:”module”` untuk membuat nodejs support perintah-perintah ecmascriptt atau ES6 ke atas, misalnya penggunaan import dan export, seperti berikut:

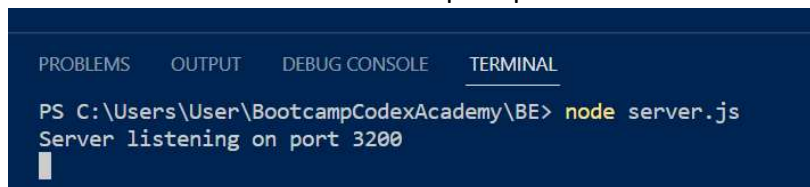


Isi file server seperti berikut, untuk membuat file server.js sebagai server yang akan melayani setiap ada request dari client.



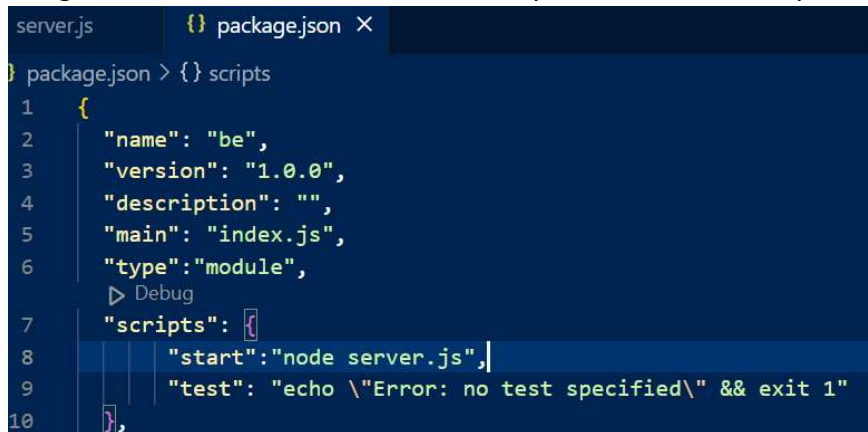
```
1 //mengambil module atau library express ke file server.js. express adalah library yang mendukung http request
2 import express from "express"
3
4 //membuat object supaya dapat mengakses semua method dari library express
5 const app=express();
6
7 //membuat port yang akan melayani request
8 const port=3200;
9
10 //supaya request berisi data dalam bentuk json dapat diterima server
11 app.use(express.json())
12
13 //supaya request berisi data selain json juga dapat diterima server
14 app.use(express.urlencoded({extended:true}))
15
16 //mendaftarkan port dan pesan yang tampil saat server sudah dijalankan, method listen bisa terima 2 parameter
17 app.listen(port,()=>{console.log("Server listening on port ${port}");})
18
```

- e. Run server dengan cara ketikkan perintah: `node server.js` pada terminal. Jika code sudah benar maka akan tampil seperti berikut:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\User\BootcampCodexAcademy\BE> node server.js
Server listening on port 3200
```

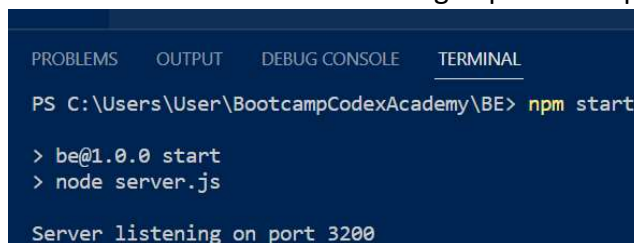
- f. Buat alias untuk perintah/script `node server.js` pada `package.json` bagian script dengan menambahkan alias baru misalnya bernama `start`, seperti berikut:



```
server.js {} package.json X
package.json > {} scripts
1 {
2   "name": "be",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "start": "node server.js",
9     "test": "echo \"Error: no test specified\" && exit 1"
10  },

```

- g. Setelah itu server bisa di run dengan perintah `npm start`, seperti berikut:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\User\BootcampCodexAcademy\BE> npm start

> be@1.0.0 start
> node server.js

Server listening on port 3200
```

- h. Jika ingin setiap perubahan pada code otomatis direload sehingga tidak perlu restart server maka perlu install nodemon. Lakukan install dengan cara `npm i --D nodemon`, seperti berikut:


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\User\BootcampCodexAcademy\BE> npm i --D nodemon
added 33 packages, and audited 92 packages in 1s

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

found 0 vulnerabilities
```

--D artinya library yang diinstall didaftarkan pada package.json bagian devDependencies, artinya library ini hanya diperlukan pada saat development saja, tidak dibutuhkan setelah diproduction, sehingga tidak harus diinstall di production.

```
server.js package.json X
package.json > {} scripts
1 {
2   "name": "be",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "start": "node server.js",
9     "test": "echo \"Error: no test specified\" && exit 1"
10  },
11  "keywords": [],
12  "author": "",
13  "license": "ISC",
14  "dependencies": {
15    "express": "^4.18.2"
16  },
17  "devDependencies": {
18    "nodemon": "^3.0.1"
19  }
20 }
```

- i. Setelah install nodemon berhasil, update bagian script alias start, yang sebelumnya node server.js menjadi nodemon server.js, seperti berikut:

```
server.js package.json X
package.json > {} dependencies
1 {
2   "name": "be",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "start": "nodemon server.js",
9     "test": "echo \"Error: no test specified\" && exit 1"
10  }
11 }
```

- j. Run kembali file server.js dengan perintah npm start, seperti berikut:

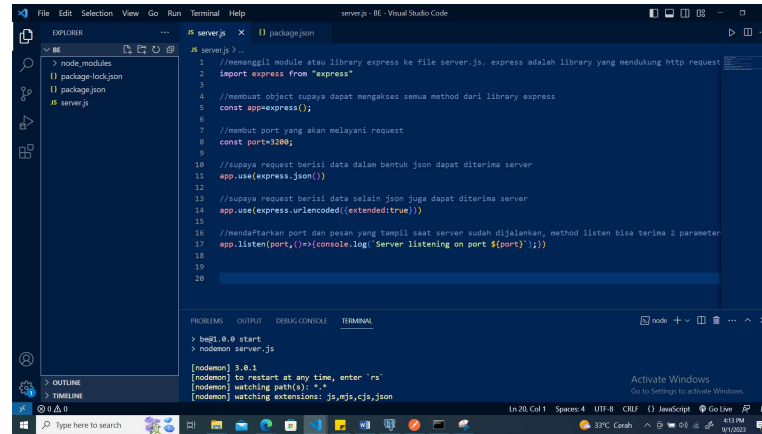
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\User\BootcampCodexAcademy\BE> npm start

> be@1.0.0 start
> nodemon server.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server listening on port 3200
```


Dengan demikian server tidak perlu direstart untuk melihat hasil perubahan pada code, akan otomatis reload.

6. Membuat file .env yang berguna untuk menyimpan parameter atau variable global seperti konfigurasi database dan port, seperti berikut:
 - a. Klik dalam kotak lokasi file project berada lalu pilih icon new file.



The screenshot shows the Visual Studio Code interface with the 'server.js' file open in the editor. The file contains the following code:

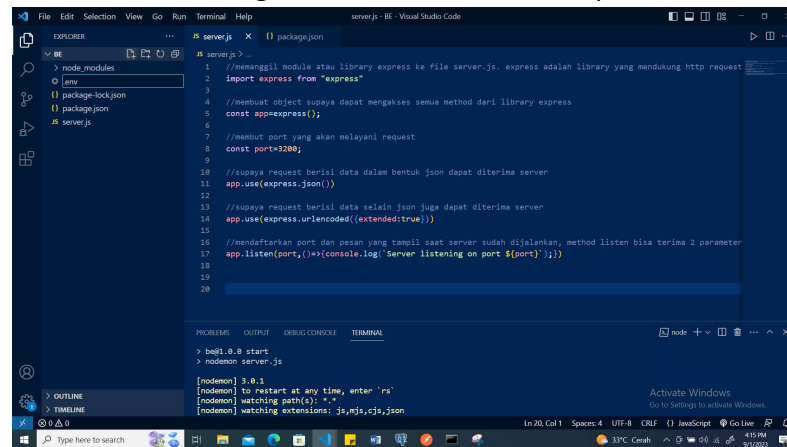
```
1 //memanggil module atau library express ke file server.js. express adalah library yang mendukung http request
2 import express from "express"
3
4 //membuat object supaya dapat mengakses semua method dari library express
5 const app=express();
6
7 //membuat port yang akan melayani request
8 const port=3200;
9
10 //supaya request berisi data dalam bentuk json dapat diterima server
11 app.use(express.json())
12
13 //supaya request berisi data selain json juga dapat diterima server
14 app.use(express.urlencoded({extended:true}))
15
16 //mendaftarkan port dan pesan yang tampil saat server sudah dijalankan, method listen bisa terima 2 parameter
17 app.listen(port,()=>{console.log("Server listening on port ${port}");})
18
19
20
```

The terminal at the bottom shows the following output:

```
> be@1.0.0 start
> nodemon server.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
```

Ketikkan .env sebagai nama file lalu enter, seperti berikut:



The screenshot shows the Visual Studio Code interface with the '.env' file open in the editor. The file contains the following code:

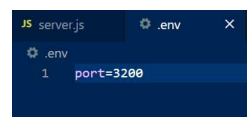
```
1 port=3200
```

The terminal at the bottom shows the following output:

```
> be@1.0.0 start
> nodemon server.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
```

Pada file .env isi variable/parameter yang menampung port server, seperti berikut:



The screenshot shows the '.env' file in the Visual Studio Code editor. The file contains the following code:

```
1 port=3200
```

- b. Load atau panggil file .env pada file server.js dan ubah pada file server.js pada bagian port menggunakan parameter pada file.env, seperti berikut:

```

serverjs > ...
1 //meload file .env
2 import "dotenv/config"
3 //memanggil module atau library express ke file server.js. express adalah library yang mendukung http request
4 import express from "express"
5
6 //membuat object supaya dapat mengakses semua method dari library express
7 const app=express();
8
9 //membuat port yang akan melayani request
10 const port=process.env.PORT || 2100;
11
12 //supaya request berisi data dalam bentuk json dapat diterima server
13 app.use(express.json())
14
15 //supaya request berisi data selain json juga dapat diterima server
16 app.use(express.urlencoded({extended:true}))
17
18 //mendaftarkan port dan pesan yang tampil saat server sudah dijalankan, method listen bisa terima 2 parameter
19 app.listen(port,()=>{console.log(`Server listening on port ${port}`);})
20

```

Pada contoh bagian port diatas pakai or sebagai kondisi jika tidak berhasil baca dari env maka port defaultnya 2100. Parameter yang ada pada env tidak case sensitive artinya tidak masalah huruf besar atau kecil.

- c. Sebelum run server.js silahkan install library dotenv dengan cara: npm i dotenv, seperti berikut:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Node.js v18.12.1
PS C:\Users\User\BootcampCodexAcademy\BE> npm i dotenv
added 1 package, and audited 93 packages in 4s
12 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

```

- d. Setelah selesai run kembali server.js dengan npm start, seperti berikut:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\User\BootcampCodexAcademy\BE> npm start
> be@1.0.0 start
> nodemon server.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server listening on port 3200

```

- e. Server untuk melayani request sudah selesai dibuat
7. Membuat method-method untuk memproses setiap request.
Server memiliki endpoint atau alamat API untuk bisa digunakan client sebagai alamat request. Ada 4 method yang umum digunakan pada request API yaitu: Get,Post,Delete,Put. Setiap Request dari client akan diresponse oleh server, contoh sebagai berikut:

```

app.use(express.json())

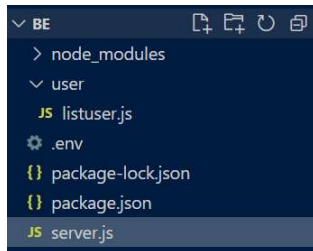
//supaya request berisi data selain json juga dapat diterima server
app.use(express.urlencoded({extended:true}))

app.get("/",(req,res)=>{
  res.send("selamat datang di web saya")
  //res.send({message:'selamat datang di web saya'})
})

```

Berdasarkan code program diatas, endpoint atau alamat API nya adalah "/" atau root direktori dengan method Get.

Cara kedua dengan memindahkan response dalam sebuah method, seperti berikut:



```

server.js  X  JS listuser.js  X  .env
ser > JS listuser.js > [0] getUser
1  const getUser = async (req, res) => {
2    try {
3
4      res.send("Hello world")
5    }
6    catch (err) {
7      res.send(err)
8    }
9
10 }
11
12 export default {
13   getUser
14 }

```

Selanjutnya import file atau module listuser.js ke dalam server.js, seperti berikut:

```

import listuser from "./user/listuser.js";

//membuat object supaya dapat mengakses semua method dari library express
const app=express();

//membuat port yang akan melayani request
const port=process.env.PORT || 2100;

//supaya request berisi data dalam bentuk json dapat diterima server
app.use(express.json())

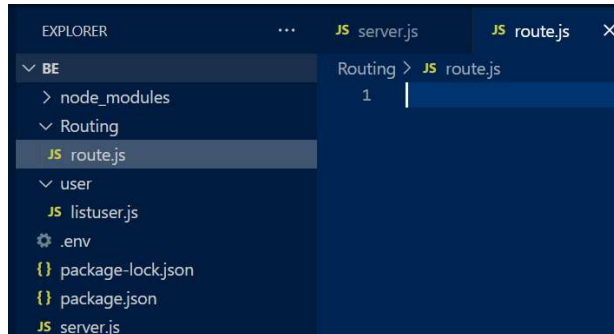
//supaya request berisi data selain json juga dapat diterima server
app.use(express.urlencoded({extended:true}))

app.get("/",
  listuser.getUser
)

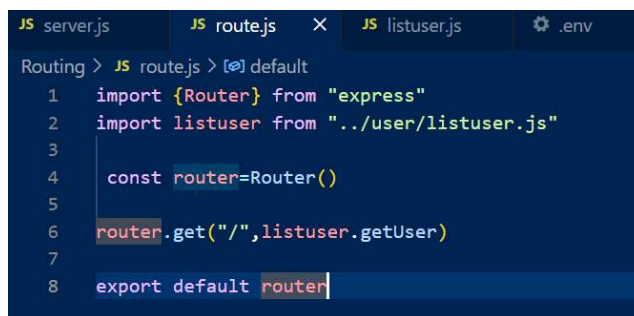
```

8. Membuat Route

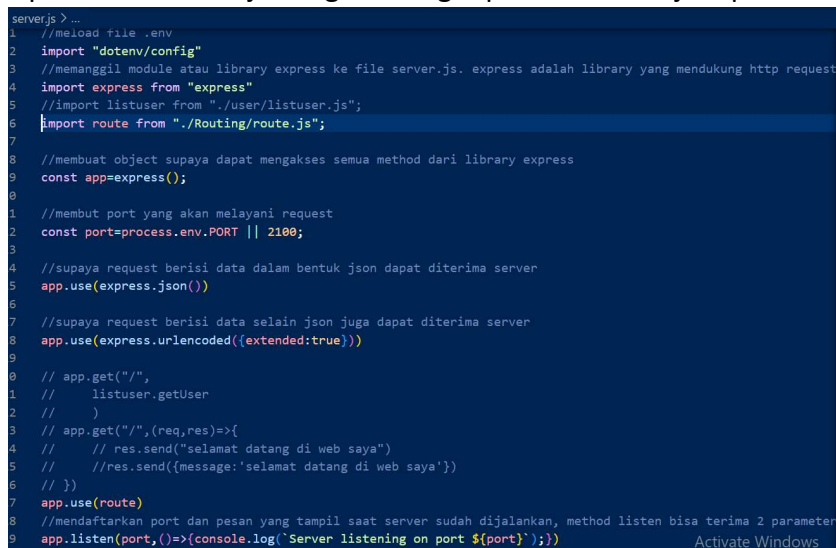
- Route yang akan dibuat berfungsi untuk memetakan setiap request dengan method yang akan memproses request. Membuat folder dan file untuk route seperti berikut:



- Import component Router dari library/module express, tambahkan code seperti berikut:



- Update file server.js dengan mengimport file route.js seperti berikut:

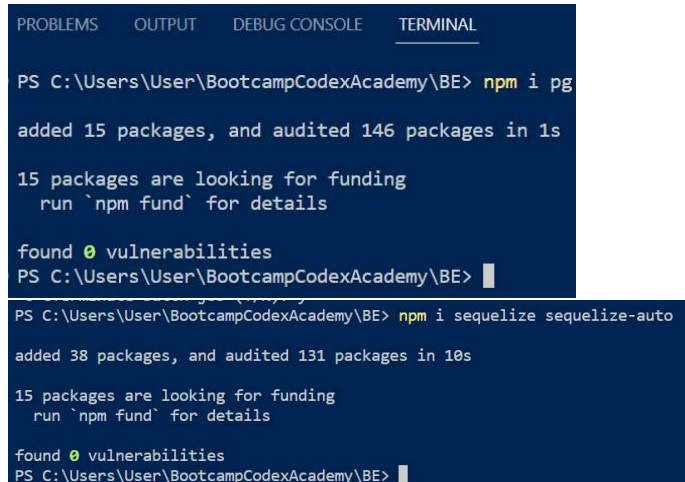


- Silahkan akses kembali melalui browser untuk memastikan request berhasil diproses server.

9. Membuat Model Database

- Model database berguna untuk memudahkan dalam membuat tipe data object sesuai field yang ada di dalam database yang akan digunakan secara otomatis.

Untuk membuat model secara otomatis menggunakan salah satu library pemodelan data di javascript yaitu: sequelize-auto. Untuk menghubungkan expressjs dengan model menggunakan library sequelize, dan untuk menghubungkan expressjs dengan database postgresql menggunakan library pg, sehingga perlu install ketiga library tersebut, seperti berikut:



```
PS C:\Users\User\BootcampCodexAcademy\BE> npm i pg
added 15 packages, and audited 146 packages in 1s

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

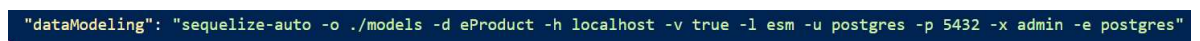
found 0 vulnerabilities
PS C:\Users\User\BootcampCodexAcademy\BE>

PS C:\Users\User\BootcampCodexAcademy\BE> npm i sequelize sequelize-auto
added 38 packages, and audited 131 packages in 10s

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

found 0 vulnerabilities
PS C:\Users\User\BootcampCodexAcademy\BE>
```

- b. Buat alias di package.json untuk menjalankan script yang generate model seperti berikut:



```
"dataModeling": "sequelize-auto -o ./models -d eProduct -h localhost -v true -l esm -u postgres -p 5432 -x admin -e postgres"
```

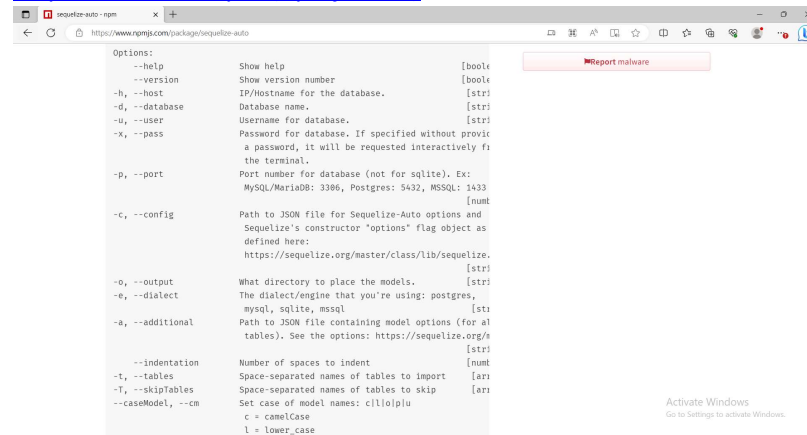
- c. Atau cara kedua langsung menjalankan script dari terminal seperti berikut:



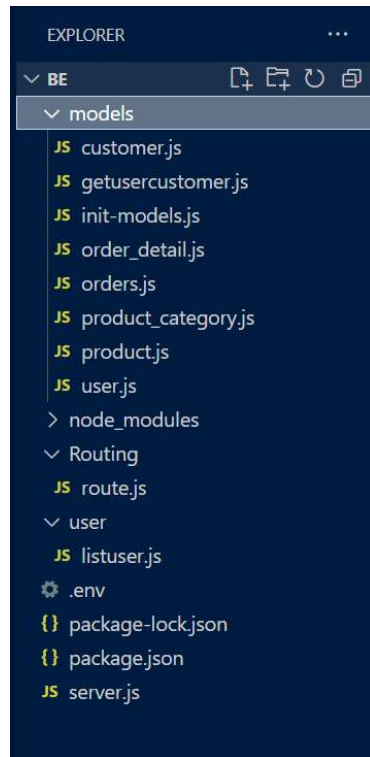
```
PS C:\Users\User\BootcampCodexAcademy\BE> npx sequelize-auto -o ./models -d eProduct -h localhost -v true -l esm -u postgres -p 5432 -x admin -e postgres
```

- d. Keterangan script bisa diakses pada url:

[sequelize-auto - npm \(npmjs.com\)](https://www.npmjs.com/package/sequelize-auto)



- e. Jika berhasil akan tergenerate folder model yang didalamnya terdapat file dengan nama sesuai nama table dalam database.



- f. Update file .env tambahkan variable untuk konfigurasi database seperti berikut:
 - g. Selanjutnya buka file init-model.js lalu lakukan perubahan seperti berikut:
 - h.
- 10. Request menggunakan Postman
 - 11. Membuat Method-method menangani CRUD
 - 12. Membuat Custom Error Handling