

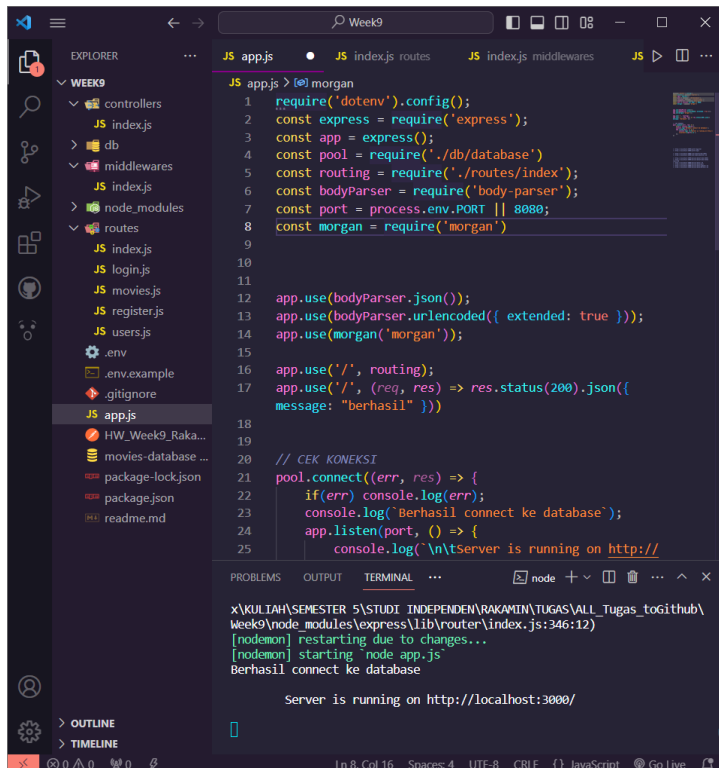
# HOMEWORK WEEK 9 - RestfullAPI & Middleware

## RestfullAPI & Middleware - Rakamin Academy

Nama : Raie Aswajjillah

Kelas : FSWD 5B

LINK LIVE Backup Hosting: [Week 9 - BACKUP](#) LINK GITHUB : [Week 9 - Suce Code](#)



```
1 require('dotenv').config();
2 const express = require('express');
3 const app = express();
4 const pool = require('./db/database');
5 const routing = require('./routes/index');
6 const bodyParser = require('body-parser');
7 const port = process.env.PORT || 8080;
8 const morgan = require('morgan');
9
10
11
12 app.use(bodyParser.json());
13 app.use(bodyParser.urlencoded({ extended: true }));
14 app.use(morgan('morgan'));
15
16 app.use('/', routing);
17 app.use('/', (req, res) => res.status(200).json({
18   message: "berhasil" }));
19
20 // CEK KONEKSI
21 pool.connect((err, res) => {
22   if(err) console.log(err);
23   console.log('Berhasil connect ke database');
24   app.listen(port, () => {
25     console.log(`\n\tServer is running on http://`);
26   });
27 });
```

PROBLEMS OUTPUT TERMINAL ...

```
x\KULIAH\SEMESTER 5\STUDI INDEPENDEN\RAKAMIN\TUGAS\ALL_Tugas_toGithub\
Week9\node_modules\express\lib\router\index.js:346:12)
[nodemon] restarting due to changes...
[nodemon] starting node app.js
Berhasil connect ke database

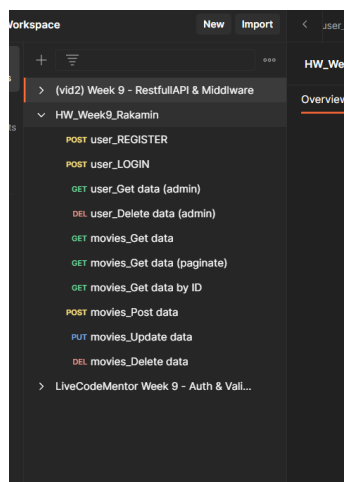
Server is running on http://localhost:3000/
```

## SOAL YANG DIBERIKAN

Ini adalah aplikasi Restfull API & Middleware pada homework pertemuan ke 9.

Pada soal 1 Saya ditugaskan untuk membuat RESTful API yang terdiri dari GET, POST, DELETE, dan PUT. Setelah itu membuat endpoint untuk register user dan login user untuk implementasi authorization dan authentication. dan yang hanya bisa mengakses API hanyalah user yang terdaftar. Lalu pada soal ke-2, saya ditugaskan untuk Lakukan Pagination pada GET users dan GET movies dengan limit 10 user. Di soal ke-3 saya ditugaskan untuk membuat dokumentasi API menggunakan swagger. dan terakhir soal ke-4 adalah mengimplementasikan Logging server pada aplikasi yang saya buat

## PENJELASAN TESTING PADA POSTMAN

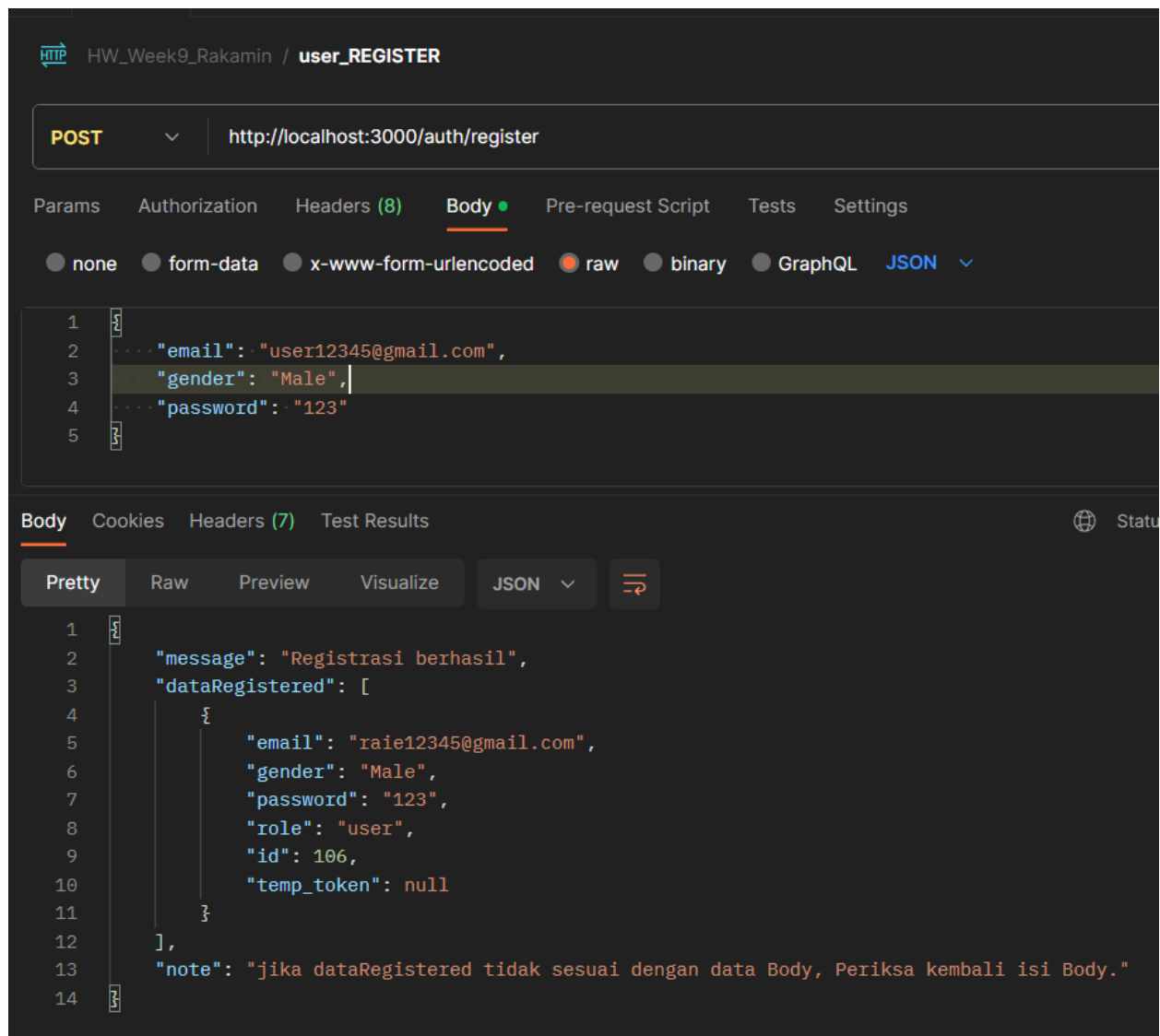


Aplikasi testing API yang saya gunakan adalah POSTMAN, dan sudah saya sediakan file backup hasil testing postman saya pada directory project yang bernama : *HW\_Week9\_Rakamin.postman\_collection.json*

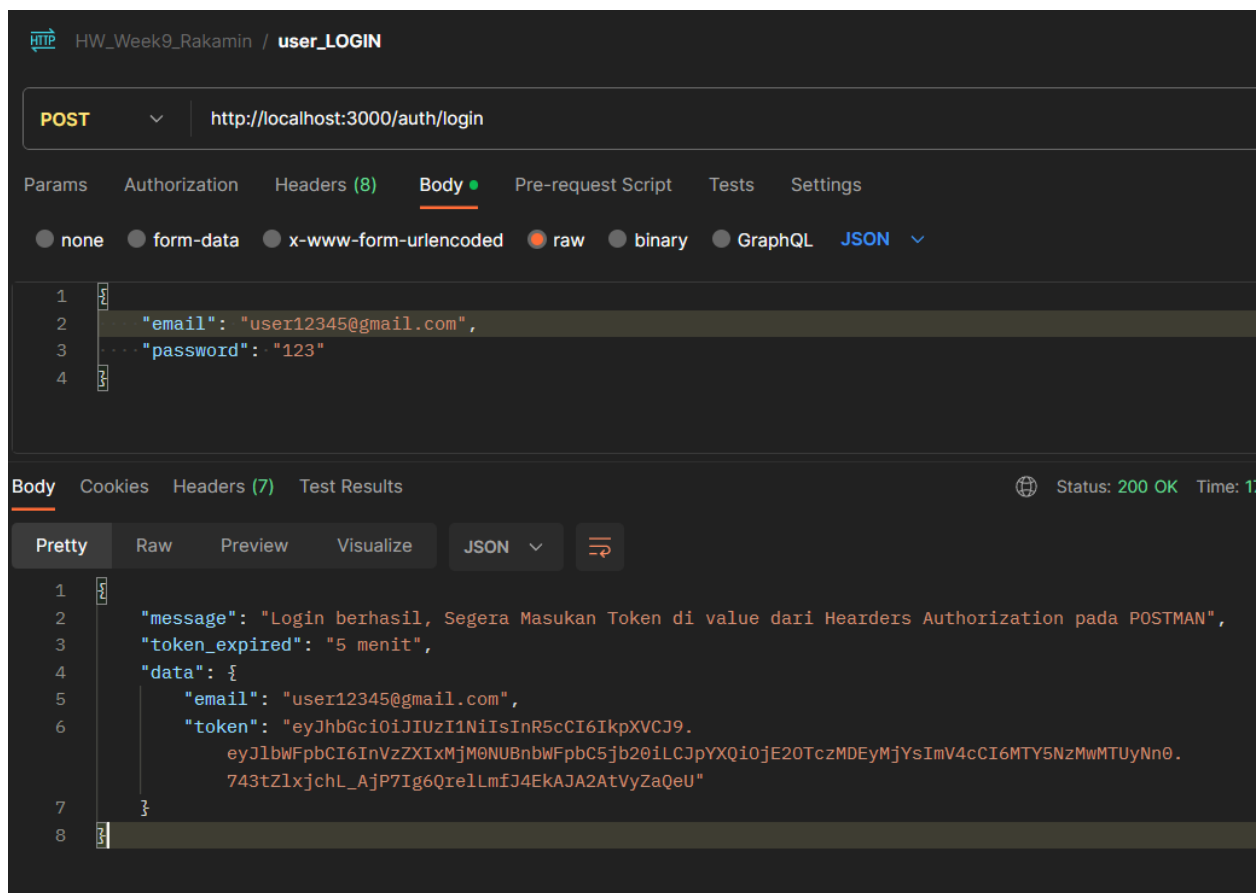
Sebelum menjalankan aplikasi import data backup SQL ke database postgresQL terlebih dahulu. data baskup SQL sudah tersedia didalam project yaitu : *movies-database(edit).sql*

Urutan testing sudah dibuat berurutan, dari atas hingga bawah seperti gambar disamping. Dimana **Pertama** kita akan melakukan register dan login.

Pada bagian register yang terjadi hanya proses POST data ke data base dengan **middleware** : jika email sudah terdaftar maka kita tidak bisa registrasi.



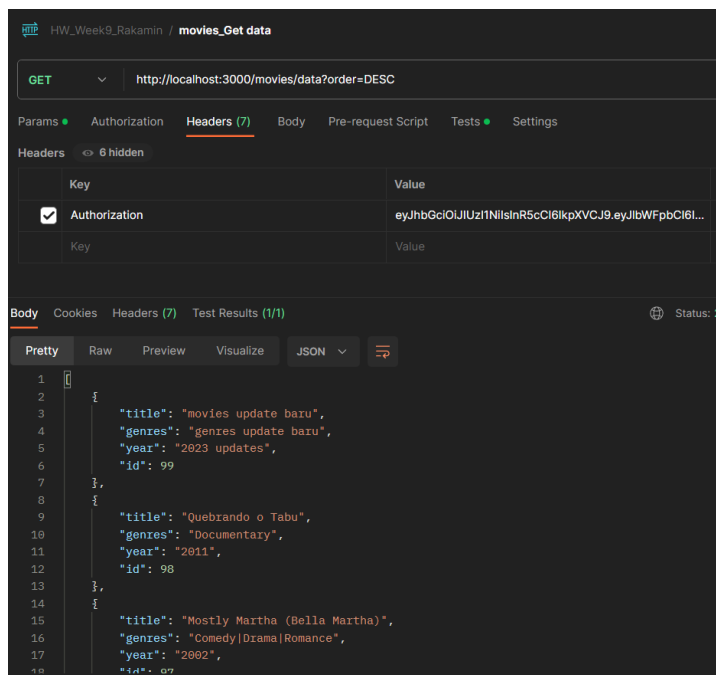
Lalu yang **kedua** adalah proses LOGIN dimana terdapat banyak fungsi diantaranya **middleware** yaitu : `checkUserExistence` dimana jika email dan password yang di simpan dibody tidak sesuai dengan database maka response body akan menyatakan bahwa Email atau password salah. ketika proses login selesai maka akan ada hasil proses dari `JSONWEBTOKEN`, dimana nantinya response dari body akan menampilkan data hasil enkripsi JWT. Ini digunakan untuk middleware pada setiap PATH lainnya, dengan menambahkan TOKEN ke bagian Headers pada value Authorization.



Berikutnya ada Contoh dari hasil login, selanjutnya kita bisa lakukan semua jenis query dari HTTP request di postman yang diantaranya adalah :

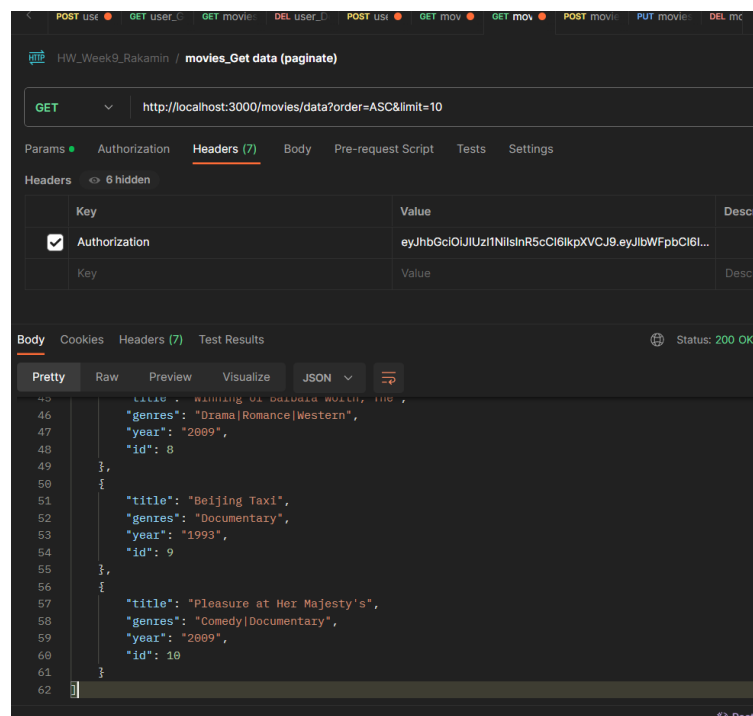
```
http://localhost:3000/auth/register
http://localhost:3000/auth/login
http://localhost:3000/user/data?order=DESC
http://localhost:3000/user/data/delete/:id
http://localhost:3000/movies/data?order=DESC
http://localhost:3000/movies/data?order=ASC&limit=10
http://localhost:3000/movies/data/:id
http://localhost:3000/movies/data/post
http://localhost:3000/movies/data/update/:id
http://localhost:3000/movies/data/delete/:id
```

Berikut adalah contoh dari proses penggunaan middleware menggunakan JWT pada proses query data, proses ini akan berjalan lancar jika kita menginput data TOKEN ke bagian Headers pada value Authorization di setiap request yang ada. Lalu perlu diperhatikan bahwa TOKEN JWT akan hangus dalam kurun waktu **5 menit**, namun kita bisa mengubahnya dengan pergi ke bagian controller.login dan mengubah isi expiresIn.



Pada gambar disamping diperlihatkan bahwa kita berhasil melakukan get semua data movies dan bisa mengurutkannya dari id yang terbesar dengan cara mengisi bagian order pada HTTP Resquest dengan isi DESC, dan jika ingin mengurutkannya dari yang terkecil bisa dengan ASC.

Berikutnya ada penggunaan **Paginate** dengan limit 10 baris hasil queries. kita bisa merubah jumlah dari paginasi menjadi berapapun dengan mengubah isi dari limit di HTTP Resquest. Dan jangan lupa juga untuk menggunakan TOKEN JWT yang telah di berikan pada saat login tadi.



Saya juga sudah membuat **dokumentasi** dari hasil Aplikasi ini dengan **swagger** pada file swagger.txt sebagai keseluruhan hasil file swager yang bisa di check pada [Swagger Editor](#), atau juga implementasinya langsung yang ada pada bagian

routes.

dan yang terakhir adalah pengimplementasian proses **logging** pada aplikasi ini dengan package bernama morgan agar setiap proses logging bisa terlihat dengan jelas dan rapih pada terminal atau console.