LOGIN DAN REGISTER MENGGUNAKAN JWT : Node JS, Express, React JS, dan MYSQL (Full-Stack)

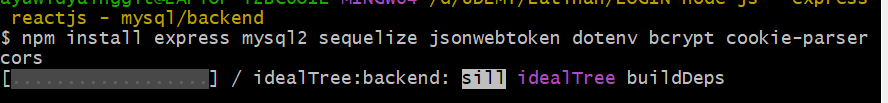
# BACKEND

## Buat package.json

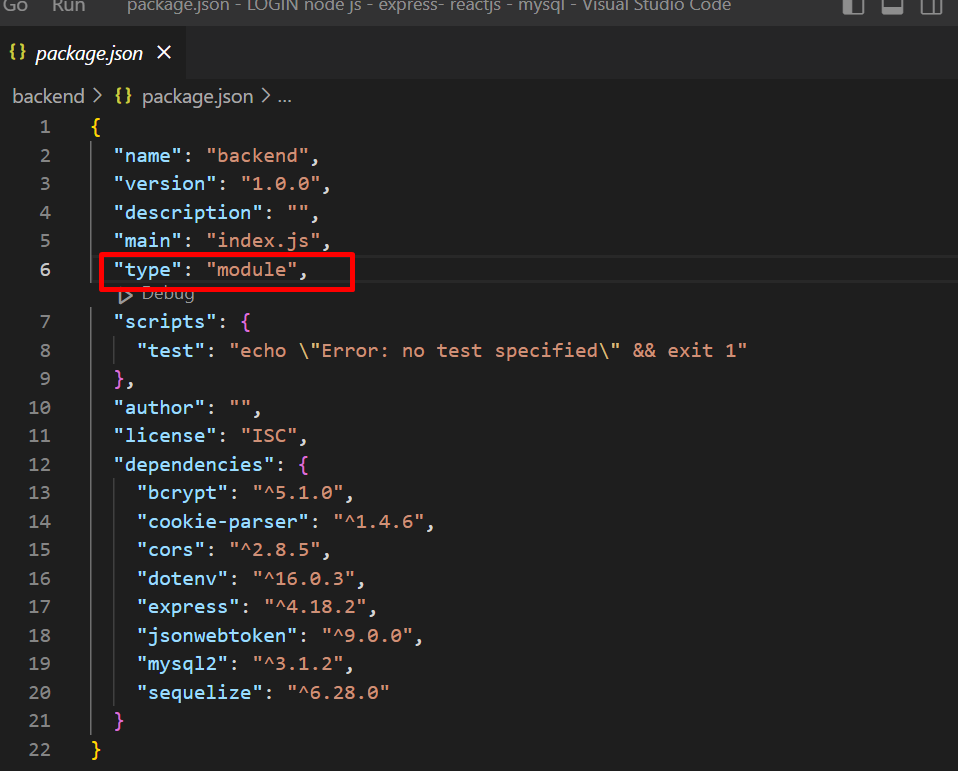
Npm init

## Install dependensi

* Express
* mysql2
* sequelize
* jsonwebtoken
* dotenv 🡪 membuat env variable
* bcrypt 🡪 untuk enkripsi pass
* cookie-parcer 🡪 parsing cookie
* cors 🡪 agar API dapat di gunakan di luar domain



Tambahkan type module



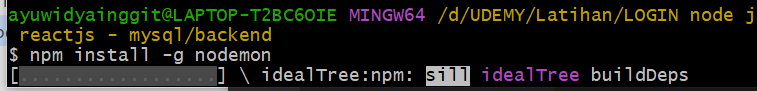
## buat file index.js

import express from 'express';

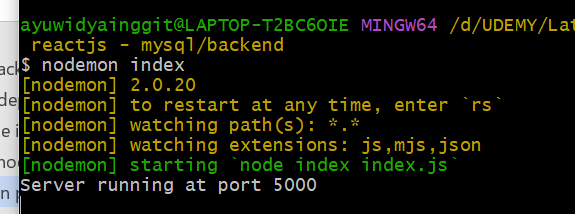
const app = express();

app.listen(5000, () => console.log('Server running at port 5000'));

## install nodemon



## jalankan projek



## buat database

nama database : my\_app

## Buat Koneksi database

### buat folder config / Database.js

import { Sequelize } from 'sequelize';

const db = new Sequelize('my\_app', 'root', '', {

  host: 'localhost',

  dialect: 'mysql',

});

export default db;

### import Database.js ke Index.js

import express from 'express';

import db from './config/Database.js';

const app = express();

// untuk memastika fungsi database berjalan dengan baik, gunakan try and catch

try {

  await db.authenticate();

  console.log('database connected...');

} catch (error) {

  console.log(error);

}

app.listen(5000, () => console.log('Server running at port 5000'));

## Generate Table Users

### Buat folder models/UserModel.js

import { Sequelize } from 'sequelize';

import db from '../config/Database.js';

const { DataTypes } = Sequelize;

//DataTypes adalah fungsi dari sequelize

const Users = db.define(

  'users',

  {

    name: DataTypes.STRING,

    email: DataTypes.STRING,

    password: DataTypes.STRING,

    refresh\_token: DataTypes.TEXT,

  },

  {

    freezeTableName: true,

  }

);

export default Users;

// membuat function (asyncronus) untuk generate table user jika user tdk terdapat di db

(async () => {

  await db.sync();

})();

### Kembali ke Index.js

import express from 'express';

import db from './config/Database.js';

import Users from './models/UserModel.js';

const app = express();

// untuk memastika fungsi database berjalan dengan baik, gunakan try and catch

try {

  await db.authenticate();

  await Users.sync(); //berfungsi jika tdk terdapat tabel maka sequelize akan mengenerate database

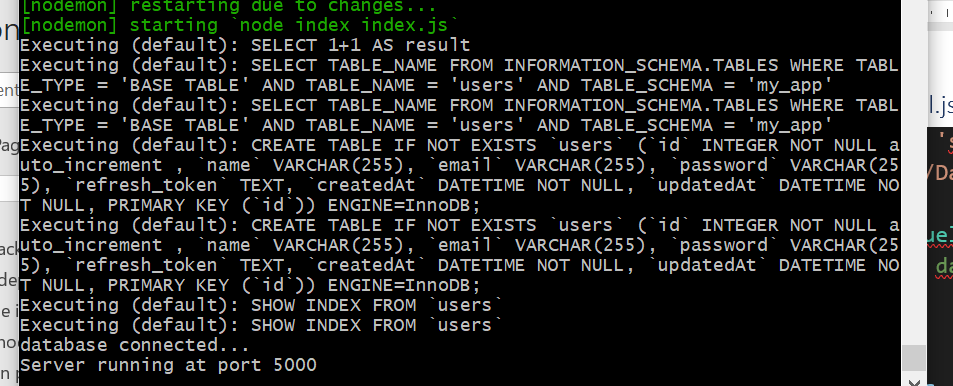
  console.log('database connected...');

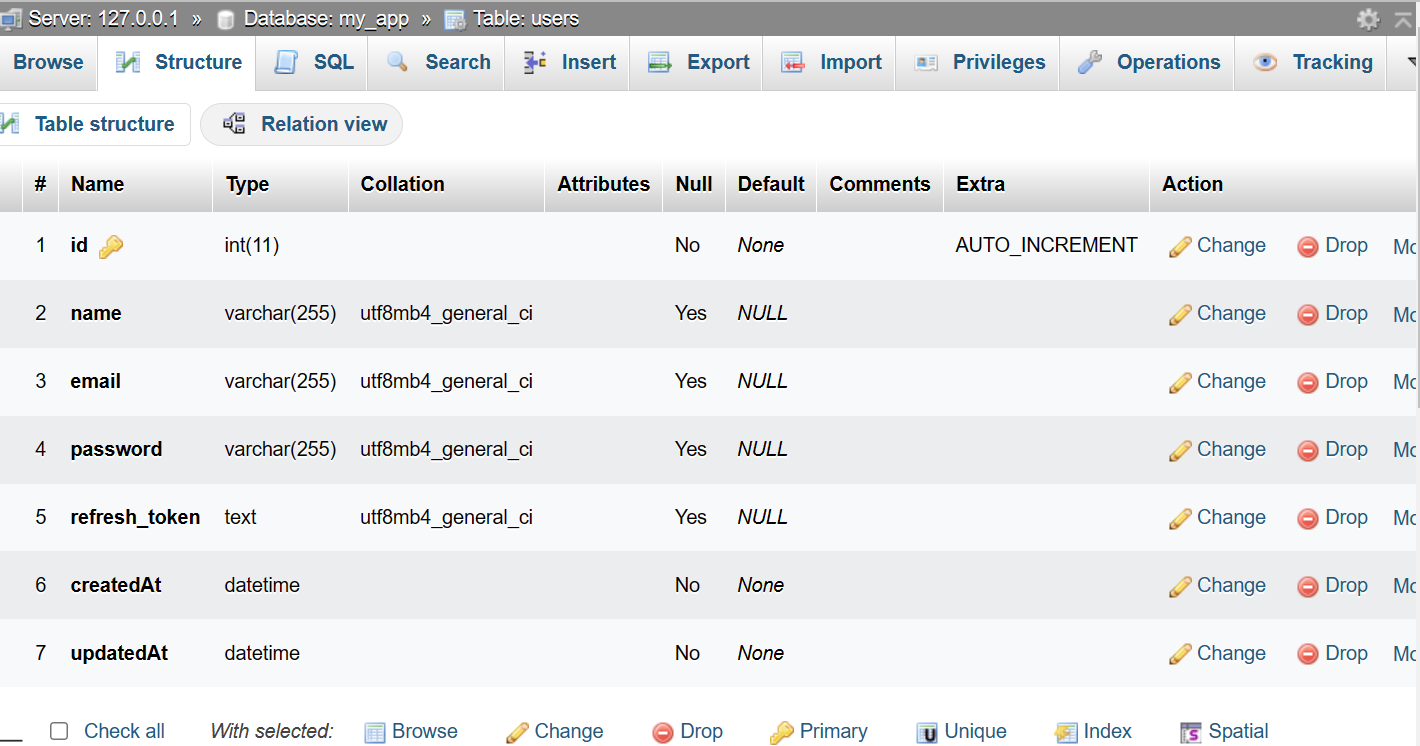
} catch (error) {

  console.log(error);

}

app.listen(5000, () => console.log('Server running at port 5000'));





Id, created\_at dan updateAt di generate otomatis oleh sequelize

## Membuat endpoint

### Buat folder controller/ UserController.js

import Users from '../models/UserModel';

export const getUsers = async (req, res) => {

  try {

    const users = await Users.findAll();

    res.json(users);

  } catch (error) {

    console.log(error);

  }

};

### Buat folder routes/index.js

import express from 'express';

import { getUsers } from '../controller/UserController';

const router = express.Router();

router.get('/users', getUsers);

export default router;

### import routes di index.js

import express from 'express';

import db from './config/Database.js';

import Users from './models/UserModel.js';

import router from './routes/index.js';

const app = express();

// untuk memastika fungsi database berjalan dengan baik, gunakan try and catch

try {

  await db.authenticate();

  await Users.sync(); //berfungsi jika tdk terdapat tabel maka sequelize akan mengenerate database

  console.log('database connected...');

} catch (error) {

  console.log(error);

}

app.use(express.json()); //berfungsi agar kita bisa menerima data dalam bentuk json

app.use(router);

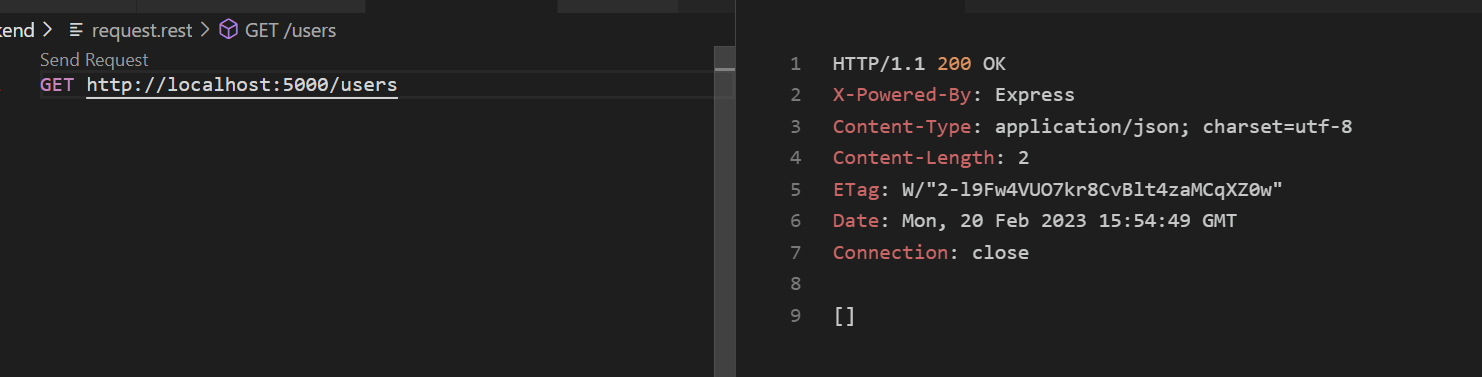
app.listen(5000, () => console.log('Server running at port 5000'));

## Melakukan pengujian API

### Gunakan ekstension REST Client

### Buat file request.rest di dalam folder backend

* Berfungsi untuk melakukan pengujian API yang sudah kita buat



### Buat function untuk regiter (UserController.js)

import Users from '../models/UserModel.js';

import bcrypt from 'bcrypt';

export const getUsers = async (req, res) => {

  try {

    const users = await Users.findAll();

    res.json(users);

  } catch (error) {

    console.log(error);

  }

};

export const Register = async (req, res) => {

  const { name, email, password, confPassword } = req.body;

  if (password !== confPassword) return res.status(400).json({ msg: 'Password and Confirm Password is not match' });

  const salt = await bcrypt.genSalt();

  const hashPassword = await bcrypt.hash(password, salt);

  try {

    await Users.create({

      name: name,

      email: email,

      password: hashPassword,

    });

    res.json({ msg: 'Register Successfully' });

  } catch (error) {

    console.log(error);

  }

};

Routes/index.js

import express from 'express';

import { getUsers, Register } from '../controller/UserController.js';

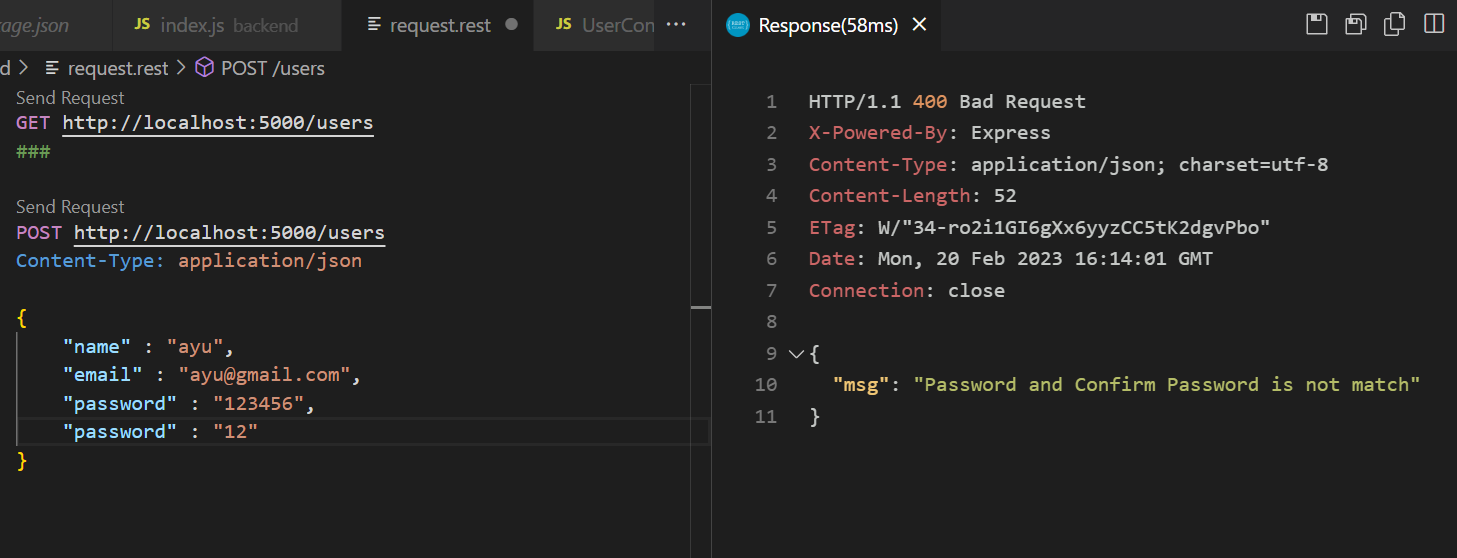
const router = express.Router();

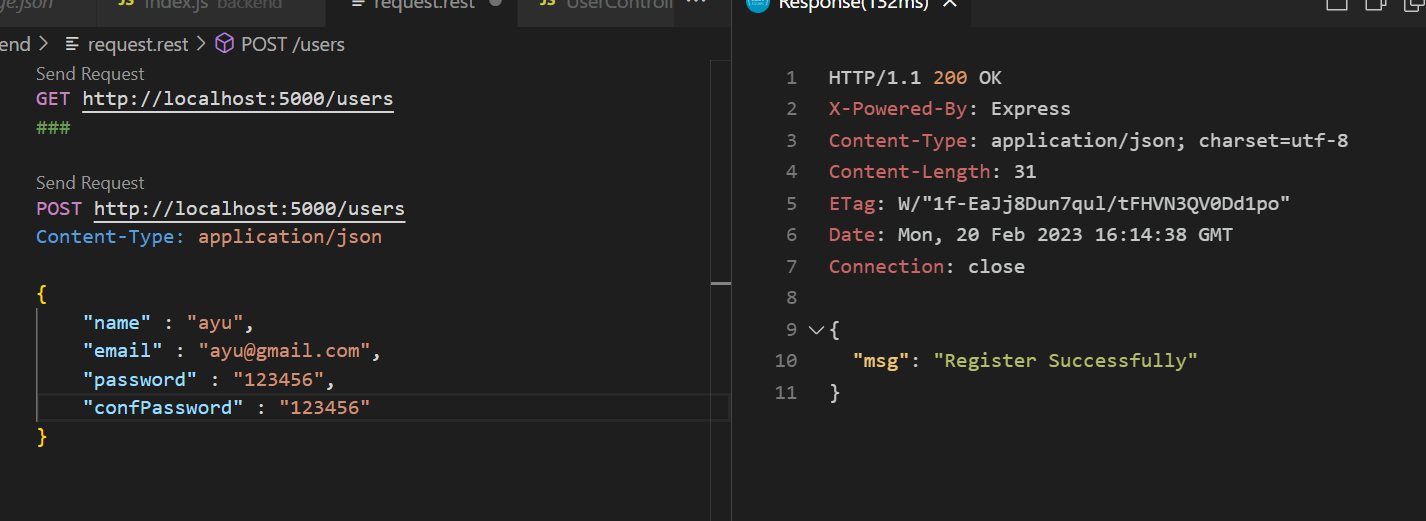
router.get('/users', getUsers);

router.post('/users', Register);

export default router;

### lakukan pengujian endpoint register





## Buat function untuk login

### Buat file .env

ACCESS\_TOKEN\_SECRET = sdhdjhskjdskdhuewdsjxzjsznhksdsnhdjshjfcbdx

REFRESH\_TOKEN\_SECRET = skdjkderudkjscjdlkfldksldsnxmdckxlkcxkcljhsyf

### Import .env di index.js

import express from 'express';

import dotenv from 'dotenv';

import db from './config/Database.js';

// import Users from './models/UserModel.js';

import router from './routes/index.js';

dotenv.config();

const app = express();

// untuk memastika fungsi database berjalan dengan baik, gunakan try and catch

try {

  await db.authenticate();

  //   await Users.sync(); //berfungsi jika tdk terdapat tabel maka sequelize akan mengenerate database

  console.log('database connected...');

} catch (error) {

  console.log(error);

}

app.use(express.json()); //berfungsi agar kita bisa menerima data dalam bentuk json

app.use(router);

app.listen(5000, () => console.log('Server running at port 5000'));

### ubah UserController.js

import Users from '../models/UserModel.js';

import bcrypt from 'bcrypt';

import jwt from 'jsonwebtoken';

export const getUsers = async (req, res) => {

  try {

    const users = await Users.findAll();

    res.json(users);

  } catch (error) {

    console.log(error);

  }

};

export const Register = async (req, res) => {

  const { name, email, password, confPassword } = req.body;

  if (password !== confPassword) {

    return res.status(400).json({ msg: 'Password and Confirm Password is not match' });

  } else {

    const salt = await bcrypt.genSalt();

    const hashPassword = await bcrypt.hash(password, salt);

    try {

      await Users.create({

        name: name,

        email: email,

        password: hashPassword,

      });

      res.json({ msg: 'Register Successfully' });

    } catch (error) {

      console.log(error);

    }

  }

};

export const Login = async (req, res) => {

  try {

    const user = await Users.findAll({

      where: {

        email: req.body.email,

        //ini digunakan untuk mencari user menggunakan email karena kita login menggunakan email

      },

    });

    // jika user ditemukan, maka kita akan membandingkan pass yang di kirim client dengan pass yang ada di DB

    const match = await bcrypt.compare(req.body.password, user[0].password); //ini digunakan untuk mengkompare pass yang di kirim client dengan yang ada di db

    if (!match) return res.status(400).json({ msg: 'Wrong password' }); //jika pass tidak cocok maka akan mengirimkan message ini

    //jika pass cocok maka akan construct satu persatu userId, name dan email nya, kita pake index ke  0 karene single data

    const userId = user[0].id;

    const name = user[0].name;

    const email = user[0].email;

    //buat access token , kasih payload , parameter 1 dlm bentuk object , parameter ke 2 ambil dr environtmentnya

    const accessToken = jwt.sign({ userId, name, email }, process.env.ACCESS\_TOKEN\_SECRET, {

      expiresIn: '20s', //dibuat exp dlm waktu 20 detik

    });

    const refreshToken = jwt.sign({ userId, name, email }, process.env.REFRESH\_TOKEN\_SECRET, {

      expiresIn: '1d', //dibuat exp dlm 1 hari

    });

    //simpan value dr variabel refreshToken  ke database, update berdasarkan id

    await Users.update(

      { refresh\_token: refreshToken },

      {

        where: {

          id: userId,

        },

      }

    );

    // setelah refreshToken disimpan ke db, kita buat httpOnly cookie yang akan dikirimkan ke client

    res.cookie('refreshToken', refreshToken, {

      httpOnly: true,

      maxAge: 24 \* 60 \* 60 \* 1000, //membuat cookie exp 1 hari

      // jika menggunakan https, tambahkan secure: true , (tidak perlu karena masih di server local )

    });

    res.json({ accessToken }); //kirim access token

  } catch (error) {

    res.status(404).json({ msg: 'Email Tidak ditemukan' }); //ini jika email tidak di temukan

  }

};

Import di routes/index.js

import express from 'express';

import { getUsers, Login, Register } from '../controller/UserController.js';

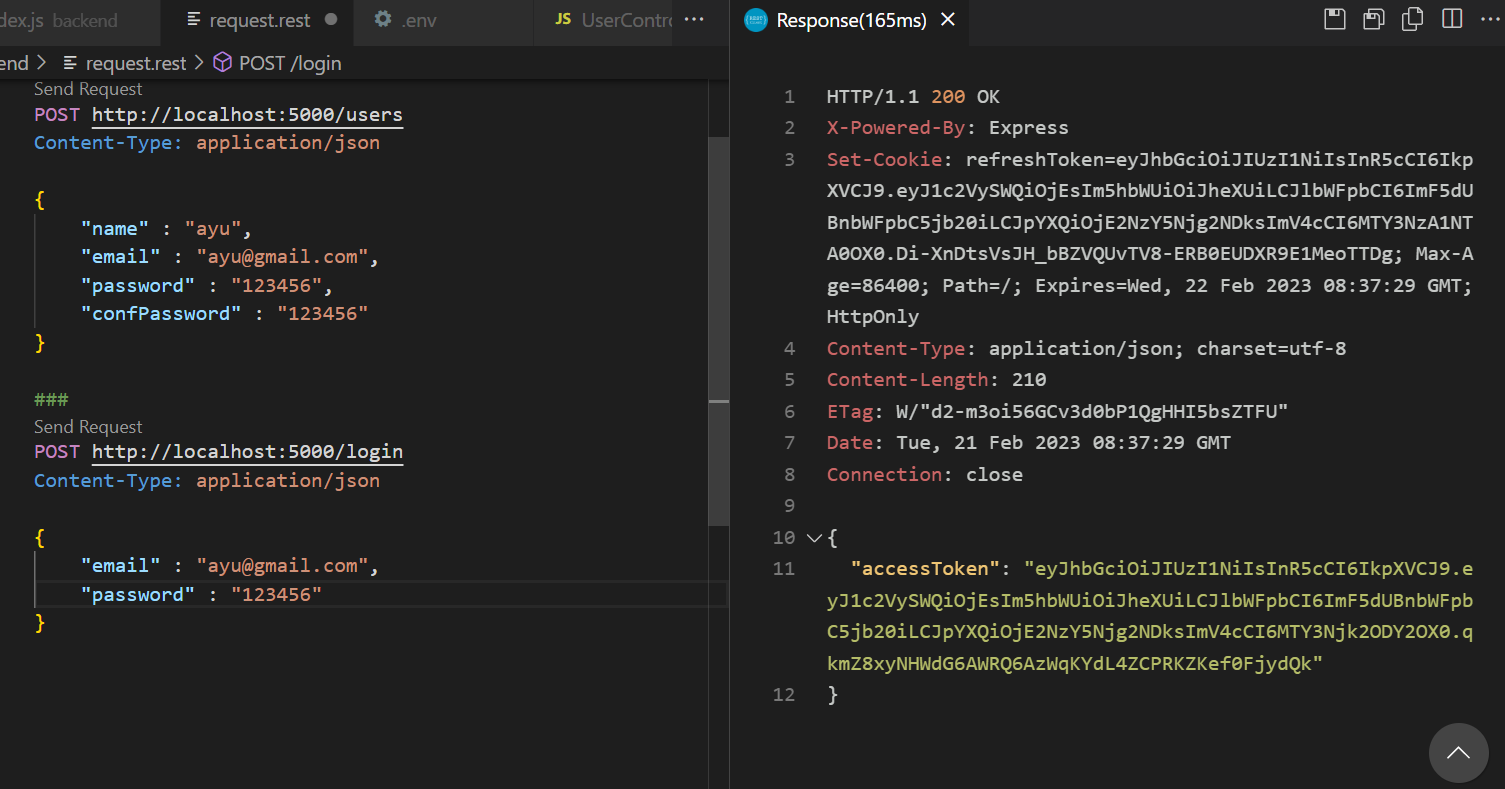
const router = express.Router();

router.get('/users', getUsers);

router.post('/users', Register);

router.post('/login', Login);

export default router;



Di cookie terdapat HTTP only , artinya cookie tidak dapat di akses oleh client

## Buat Middleware untuk verifikasi token

### Buat folder middleware, dan file VerifyToken.js

### Import middleware di dlam route/index.js

import express from 'express';

import { getUsers, Login, Register } from '../controller/UserController.js';

import { verifyToken } from '../middleware/VerifyToken.js'; // untuk memferivikasi endpoint yang tdk dapat di akses jika user tdk login

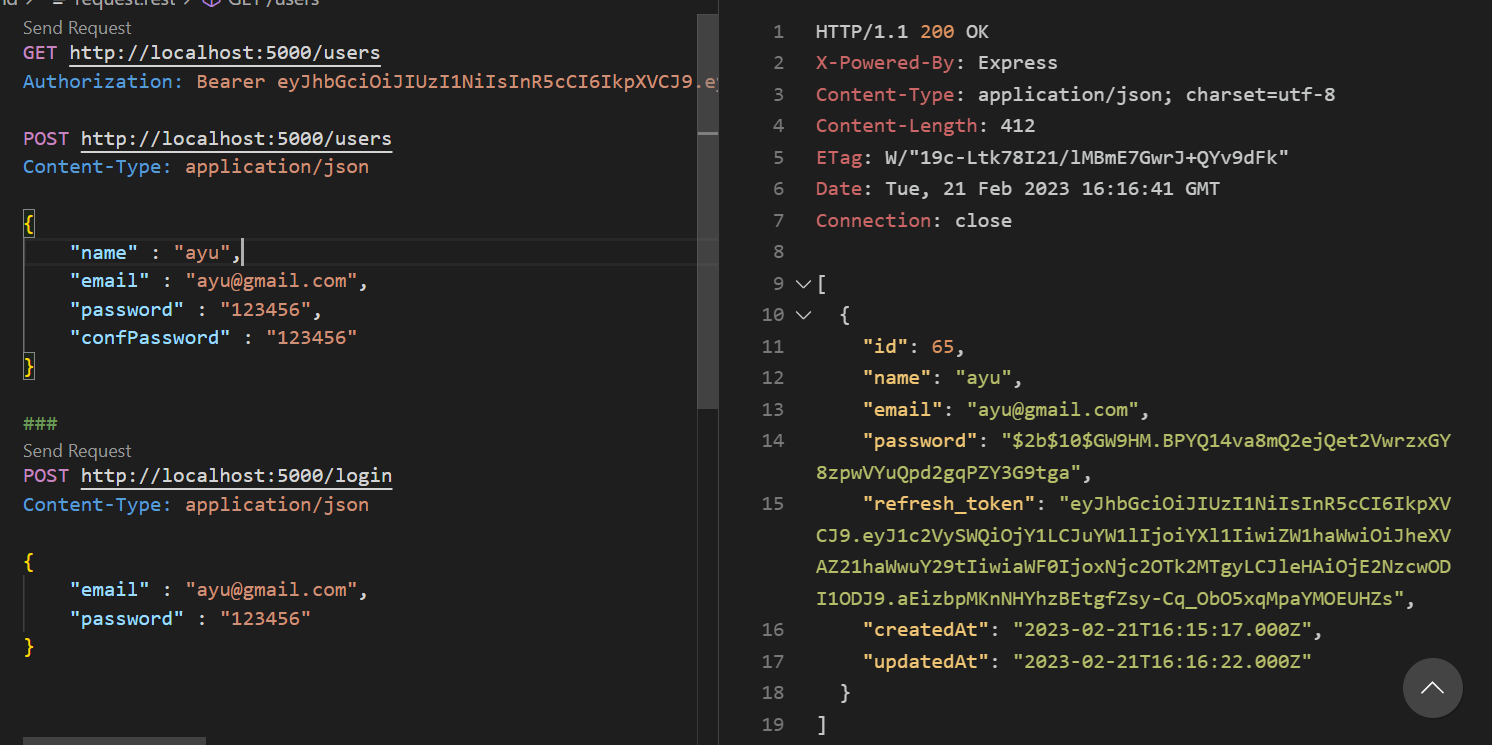
const router = express.Router();

router.get('/users', verifyToken, getUsers);

router.post('/users', Register);

router.post('/login', Login);

export default router;



### Kita akan hilanngkan respon pass dan refresh token (UserController.js)

import Users from '../models/UserModel.js';

import bcrypt from 'bcrypt';

import jwt from 'jsonwebtoken';

export const getUsers = async (req, res) => {

  try {

    const users = await Users.findAll({

      attributes: ['id', 'name', 'email'],

    });

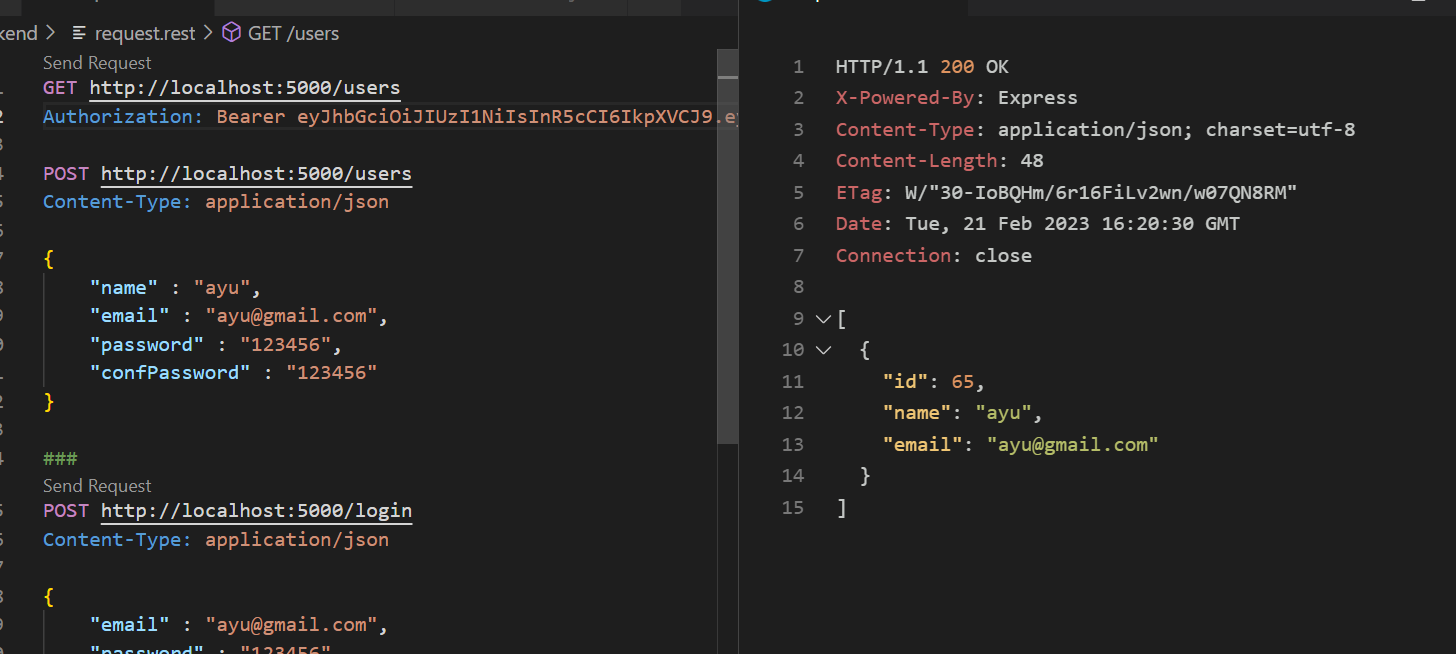
    res.json(users);

  } catch (error) {

    console.log(error);

  }

};



### Buat refreshToken jika exp tidak perlu login lagi (index.js)

import express from 'express';

import dotenv from 'dotenv';

import db from './config/Database.js';

// import Users from './models/UserModel.js';

import router from './routes/index.js';

import cookieParser from 'cookie-parser'; //untuk parsing cookie dr refreshToken

dotenv.config();

const app = express();

// untuk memastika fungsi database berjalan dengan baik, gunakan try and catch

try {

  await db.authenticate();

  //   await Users.sync(); //berfungsi jika tdk terdapat tabel maka sequelize akan mengenerate database

  console.log('database connected...');

} catch (error) {

  console.log(error);

}

app.use(cookieParser()); //gunakan dlm bentuk middleware

app.use(express.json()); //berfungsi agar kita bisa menerima data dalam bentuk json

app.use(router);

app.listen(5000, () => console.log('Server running at port 5000'));

sekarang kita dapat memanggil value cookienya

### buat controller baru RefreshToken.js

import Users from '../models/UserModel';

import jwt from 'jsonwebtoken';

// buat func untuk refresh token

export const refreshToken = async (req, res) => {

  try {

    const refreshToken = req.cookies.refreshToken;

    if (!refreshToken) return res.sendStatus(401);

    const user = await Users.findAll({

      where: {

        refresh\_token: refreshToken,

      },

    });

    if (!user[0]) return res.sendStatus(403);

    jwt.verify(refreshToken, process.env.REFRESH\_TOKEN\_SECRET, (err, decode) => {

      if (err) return res.sendStatus(403);

      const userId = user[0].id;

      const name = user[0].name;

      const email = user[0].email;

      const accessToken = jwt.sign({ userId, name, email }, process.env.ACCESS\_TOKEN\_SECRET, {

        expiresIn: '15s',

      });

      res.json({ accessToken });

    });

  } catch (error) {

    console.log(error);

  }

};

### Panggil di route/index.js

import express from 'express';

import { refreshToken } from '../controller/RefreshToken.js';

import { getUsers, Login, Register } from '../controller/UserController.js';

import { verifyToken } from '../middleware/VerifyToken.js'; // untuk memferivikasi endpoint yang tdk dapat di akses jika user tdk login

const router = express.Router();

router.get('/users', verifyToken, getUsers);

router.post('/users', Register);

router.post('/login', Login);

router.get('/token', refreshToken);

export default router;

## Logout

### Tambahkan func di UserController.js

export const Logout = async (req, res) => {

  const refreshToken = req.cookies.refreshToken;

  if (!refreshToken) return res.sendStatus(204);

  const user = await Users.findAll({

    where: {

      refresh\_token: refreshToken,

    },

  });

  if (!user[0]) return res.sendStatus(204);

  const userId = user[0].id;

  await Users.update(

    { refresh\_token: null },

    {

      where: {

        id: userId,

      },

    }

  );

  res.clearCookie('refreshToken');

  return res.sendStatus(200);

};

### Import di routes

import express from 'express';

import { refreshToken } from '../controller/RefreshToken.js';

import { getUsers, Login, Logout, Register } from '../controller/UserController.js';

import { verifyToken } from '../middleware/VerifyToken.js'; // untuk memferivikasi endpoint yang tdk dapat di akses jika user tdk login

const router = express.Router();

router.get('/users', verifyToken, getUsers);

router.post('/users', Register);

router.post('/login', Login);

router.get('/token', refreshToken);

router.delete('/logout', Logout);

export default router;

## Agar api dapat digunakan dr luar

Index.js

import express from 'express';

import dotenv from 'dotenv';

import db from './config/Database.js';

// import Users from './models/UserModel.js';

import router from './routes/index.js';

import cookieParser from 'cookie-parser'; //untuk parsing cookie dr refreshToken

import cors from 'cors';

dotenv.config();

const app = express();

// untuk memastika fungsi database berjalan dengan baik, gunakan try and catch

try {

  await db.authenticate();

  //   await Users.sync(); //berfungsi jika tdk terdapat tabel maka sequelize akan mengenerate database

  console.log('database connected...');

} catch (error) {

  console.log(error);

}

app.use(cors({ credentials: true, origin: 'http://localhost:3000' }));

app.use(cookieParser()); //gunakan dlm bentuk middleware

app.use(express.json()); //berfungsi agar kita bisa menerima data dalam bentuk json

app.use(router);

app.listen(5000, () => console.log('Server running at port 5000'));