AUTH.ROUTER

const { Router } = require('express');

const { generateToken, getPayload } = require('../utils/jwt');

const router = Router();

const USERS = [];

router.get('/', (*req*, *res*) => {

*res*.send(USERS);

});

router.get('/getUserfromToken', getPayload, (*req*, *res*) => {

  console.log(*req*.payload);

*res*.send({ ...*req*.payload, user: USERS[*req*.payload.index] });

});

router.post('/login', (*req*, *res*) => {

  const username = *req*.body.username;

  const index = USERS.findIndex((*user*) => *user*.username == username);

  console.log(index);

  if (index == -1) {

*res*.status(404).send({ error: 'usuario no encontrado' });

  } else {

    const token = generateToken({ index });

*res*.send({ msg: 'usuario exitosamente encontrado', token });

  }

*res*.send('login');

});

router.post('/register', (*req*, *res*) => {

  const { username, email } = *req*.body;

  const exist = USERS.findIndex((*userArray*) => *userArray*.username == username);

  if (exist == -1) {

    USERS.push({ username, email });

*res*.send({ msg: 'register' });

  } else {

*res*.status(403).send({ error: 'usuario existente' });

  }

});

*module*.*exports* = router;

UTIS/JWT

const jwt = require('jsonwebtoken');

const PRIVATE\_KEY = 'sOlo1los2MuerTos3Ven4el5Fin6de7La8Guerra9';

const generateToken = (*payload*) => {

  const token = jwt.sign({ payload }, PRIVATE\_KEY, { expiresIn: '5m' });

  return token;

};

const getPayload = (*req*, *res*, *next*) => {

  const headerAuth = *req*.headers.authorization;

  if (!headerAuth) {

    return *res*.status(401).send({ error: 'no se encontro token' });

  }

  const token = headerAuth.split(' ')[1];

  if (token) {

    jwt.verify(token, PRIVATE\_KEY, (*e*, *credential*) => {

      if (*e*) {

*res*.status(500).send({ error: 'ocurrio un error inesperado', e });

      } else {

*req*.payload = *credential*.payload;

        next();

      }

    });

  } else {

*res*.status(401).send({ error: 'no se encontro token' });

  }

};

*module*.*exports* = {

  generateToken,

  getPayload,

};