**ЛАБОРАТОРНА РОБОТА № 3-5**

**NodeJS. TaskApp**

Хід роботи

Лістинг коду:

app.js:

const express = require('express');

require('../db/mongoose');

const userRouter = require('../src/routers/user');

const taskRouter = require('../src/routers/task');

const app = express();

app.use(express.json());

app.use(userRouter);

app.use(taskRouter);

const PORT = process.env.PORT || 3000;

app.listen(PORT, (err) => {

err ? console.log(err) : console.log(`Listening port ${PORT}`);

});

module.exports = app;

routers/user.js:

const express = require('express');

const auth = require('../middleware/auth');

const {

getUsers,

getUser,

addUser,

deleteUser,

updateUser,

loginUser,

authenticate,

logout,

logoutAll

} = require('../../controllers/user-controller');

const router = express.Router();

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

router.get('/users/me', auth, authenticate);

router.get('/users/:id', getUser);

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

router.delete('/users/:id', deleteUser);

router.patch('/users/:id', updateUser);

router.post('/users/login', loginUser);

router.post('/users/logout', auth, logout);

router.post('/users/logoutAll', auth, logoutAll);

module.exports = router;

routers/task.js:

const express = require('express');

const auth = require('../middleware/auth');

const {

getTasks,

getTask,

addTask,

deleteTask,

updateTask,

} = require('../../controllers/task-controller');

const router = express.Router();

router.get('/tasks', auth, getTasks);

router.get('/tasks/:id', auth, getTask);

router.post('/tasks', auth, addTask);

router.delete('/tasks/:id', auth, deleteTask);

router.patch('/tasks/:id', auth, updateTask);

module.exports = router;

middleware/auth.js:

const jwt = require('jsonwebtoken');

const User = require('../../models/user');

const auth = async (req, res, next) => {

try {

const token = req.header('Authorization').replace("Bearer ", "");

const decoded = jwt.verify(token, 'kdweueksdsjfij');

const user = await User.findOne({ \_id: decoded.\_id, 'tokens.token': token });

if(!user) {

throw new Error();

}

req.user = user;

req.token = token;

next();

} catch (error) {

res.status(403).send({ error: "Please authenticate." });

}

};

module.exports = auth;

models/user.js:

const mongoose = require('mongoose');

const validator = require("validator");

const bcrypt = require('bcrypt');

const UserSchema = new mongoose.Schema({

name: {

type: String,

required: true,

trim: true

},

password: {

type: String,

required: true,

validate(value) {

if (value.length < 7) {

throw new Error("Password must contain more than 6 characters");

} else if (value === 'password') {

throw new Error("password cannot be 'password'");

}

}

},

age: {

type: Number,

required: true,

default: 0,

validate(value) {

if (value < 0) {

throw new Error("Age must be a positive number");

}

}

},

email: {

type: String,

required: true,

unique: true,

trim: true,

lowercase: true,

validate(value) {

if (!validator.isEmail(value)) {

throw new Error('Invalid email address');

}

},

},

tokens: [{

token: {

type: String,

required: true

}

}]

}, {toJSON: {virtuals: true}, toObject: {virtuals: true}});

// Перед збереженням хешуємо пароль

UserSchema.pre('save', async function (next) {

// Отримуємо екземпляр даного користувача

const user = this;

// Якщо модифікується пароль

if (user.isModified('password')) {

// Зашифруємо його

user.password = await bcrypt.hash(user.password, 8);

}

next();

});

UserSchema.virtual('tasks', {

ref: 'Task',

localField: '\_id',

foreignField: 'owner'

});

UserSchema.methods.JSON = function() {

const user = this;

const userObject = user.toObject();

delete userObject.password;

delete userObject.tokens;

return userObject;

}

const User = mongoose.model('User', UserSchema);

module.exports = User;

models/task.js:

const mongoose = require('mongoose');

const TaskSchema = new mongoose.Schema({

title: {

type: String,

required: true,

trim: true

},

description: {

type: String,

required: true,

trim: true

},

completed: {

type: Boolean,

default: false

},

owner: {

type: mongoose.Schema.Types.ObjectId,

ref: "User",

required: true

}

});

const Task = mongoose.model('Task', TaskSchema);

module.exports = Task;

db/mongoose.js:

const mongoose = require('mongoose');

const env = require('dotenv').config();

const URL = process.env.MONGODB\_URL;

mongoose

.connect(URL)

.then(() => console.log('Connected to MongoDB'))

.catch((err) => console.log(`DB connection error: ${err}`));

controllers/user-controller.js:

const User = require('../models/user');

const bcrypt = require("bcrypt");

const jwt = require("jsonwebtoken");

const handleError = (res, error) => {

res.send(error.message);

}

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

try {

const users = await User.find({});

res.status(200).json(users);

} catch(err) {

handleError(res, err);

}

};

const getUser = async (req, res) => {

try {

const user = await User.findById(req.params.id);

res.status(200).json(user);

} catch(err) {

handleError(res, err);

}

};

const addUser = async (req, res) => {

const user = new User(req.body);

try {

await user.save();

res.status(200).json(user);

} catch(err) {

res.status(401).send();

}

};

const deleteUser = async (req, res) => {

try {

const user = await User.findByIdAndDelete(req.params.id);

if (!user) {

res.status(404).json();

} else {

res.status(200).json(user);

}

} catch(err) {

handleError(res, err);

}

};

const updateUser = async (req, res) => {

try {

const user = await User.findOne({\_id: req.params.id});

if (!user) {

res.status(404).json();

throw new Error("User not found");

}

const fields = ['firstName', 'lastName', 'age', 'password'];

fields.forEach((field) => {

if (req.body[field]) {

user[field] = req.body[field];

}

});

await user.save();

res.json(user);

} catch(err) {

handleError(res, err);

}

};

const loginUser = async (req, res) => {

try {

const user = await User.findOne({ email: req.body.email });

if (!user) {

return res.status(400).send({ error: 'Incorrect email' });

}

const isMatch = await bcrypt.compare(req.body.password, user.password);

if (!isMatch) {

return res.status(400).send({ error: 'Incorrect password' });

}

const token = jwt.sign({\_id: user.\_id.toString()}, 'kdweueksdsjfij');

user.tokens = user.tokens.concat({ token });

await user.save();

res.send({user, token});

} catch(err) {

res.status(400).send();

}

};

const authenticate = async (req, res) => {

res.send(req.user);

};

const logout = async (req, res) => {

try {

req.user.tokens = req.user.tokens.filter((token) => {

return token.token !== req.token;

});

await req.user.save();

res.send({ message: 'Logged out successfully' });

} catch (err) {

res.status(500).send(err);

}

};

const logoutAll = async (req, res) => {

try {

req.user.tokens = [];

await req.user.save();

res.send({ message: 'Logged out all successfully' });

} catch (err) {

res.status(500).send(err);

}

};

module.exports = {

getUsers,

getUser,

addUser,

deleteUser,

updateUser,

loginUser,

authenticate,

logout,

logoutAll

};

task-controller.js:

const Task = require('../models/task');

const handleError = (res, error) => {

res.send(error.message);

}

const getTasks = async (req, res) => {

try {

const tasks = await Task.find({ owner: req.user.\_id });

res.status(200).json(tasks);

} catch(err) {

handleError(res, err);

}

};

const getTask = async (req, res) => {

try {

const task = await Task.findById(req.params.id);

if (task.owner.toString() !== req.user.\_id.toString()) {

res.status(404).json('The task does not belong to an authorised user');

} else {

res.status(200).json(task);

}

} catch(err) {

handleError(res, err);

}

};

const addTask = async (req, res) => {

const task = new Task({

...req.body,

owner: req.user.id

});

try {

await task.save();

res.status(200).json(task);

} catch(err) {

handleError(res, err);

}

};

const deleteTask = async (req, res) => {

try {

const task = await Task.findOneAndDelete({ \_id: req.params.id, owner: req.user.\_id });

if (!task) {

res.status(404).json();

} else {

res.status(200).json(task);

}

} catch(err) {

handleError(res, err);

}

};

const updateTask = async (req, res) => {

try {

const task = await Task.findOne({ \_id: req.params.id, owner: req.user.\_id });

if (!task) {

res.status(404).send({ error: "The task does not exist or does not belong to the current user." });

} else {

Object.keys(req.body).forEach(key => {

task[key] = req.body[key];

});

await task.save();

res.send(task);

}

} catch(err) {

handleError(res, err);

}

};

module.exports = {

getTasks,

getTask,

addTask,

deleteTask,

updateTask

};

Перевіряти це все доволі довго, але я завіряю вас все працює😊