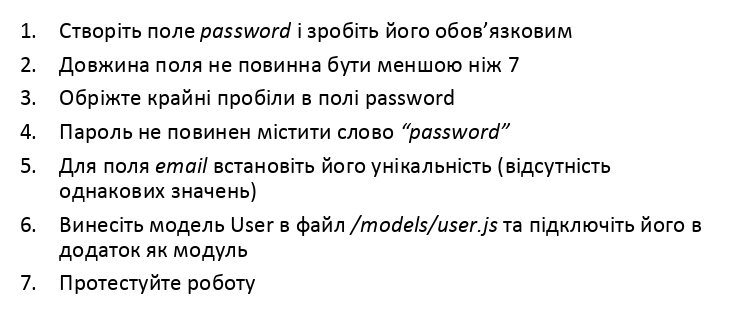
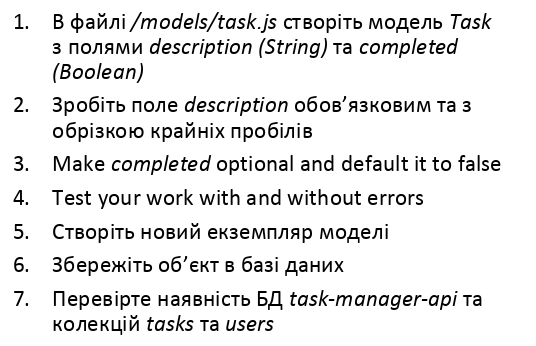
**Лабораторна робота №2**

**ЗНАЙОМСТВО З NODE.JS, MONGOOSE, REST API**



**Завдання для моделі Task.**



User.js

require('../db/mongoose')  
const ***mongoose*** = require("mongoose");  
const eValidator = require("email-validator")  
const { isValidPassword } = require("mongoose-custom-validators")  
  
let User1 = new ***mongoose***.Schema({  
 name: {type: ***String***, required: true, trim: true},  
 email: {  
 type: ***String***, unique: true, validate(value) {  
 if (eValidator.validate(value) === false) {  
 throw new ***Error***("Error: email is invalid");  
 }  
 },  
 validate: {  
 validator: async function (email) {  
 const user = await this.constructor.findOne({email});  
 if (user) {  
 if (this.id === user.id) {  
 return true;  
 }  
 return false;  
 }  
 return true;  
 },  
 message: props => 'Email is already in use.'  
 },  
 required: [true, 'User email required']  
 },  
 age: {  
 type: ***Number***,  
 default: 0,  
 validate(value) {  
 if (value < 0) {  
 throw new ***Error***("Age cannot be below zero")  
 }  
 }  
 },  
 password: {  
 type: ***String***,  
 required: true,  
 trim: true,  
 validate: {  
 validator: (value) => isValidPassword(value, {minlength: 7}),  
 message: 'Password is too short'  
 },  
 validate(value) {  
 if (value.match(/\bpassword\S\*\b/g)) {  
 throw new ***Error***("Too easy:)")  
 }  
 }  
 }  
})  
const ***User*** = ***mongoose***.model("User", User1)  
module.exports = ***User***;

task.js

require('../db/mongoose')  
const ***mongoose*** = require("mongoose");  
  
const ***Task*** = ***mongoose***.model('Task', {  
 description: {type: ***String***, required: true, trim: true},  
 completed: {type: ***Boolean***, default: false}  
})  
  
module.exports = ***Task***

mongoose.js

const ***mongoose*** = require("mongoose");  
***mongoose***.connect('mongodb://127.0.0.1:27017/task-manager-api', {  
 useNewUrlParser: true,  
 useCreateIndex: true  
})

index.js

require('./db/mongoose')  
const ***express*** = require("express")  
const app = ***express***()  
const ***User*** = require("./models/user")  
const ***Task*** = require("./models/task")  
  
const user = new ***User***({name: 'Alex', email: 'alex@gmail.com', age: 34, password: 'smth\_here'})  
  
user.save().then(() => {  
 ***console***.log(user);  
}).catch((error) => {  
 ***console***.log(error);  
});

const task = new ***Task***({description: "some task", completed: true});  
  
task.save().then(() => {  
 ***console***.log(task);  
}).catch((error) => {  
 ***console***.log(error);  
});  
  
app.get("/users", (req, res) => {  
 ***User***.find({}).then((users) => {  
 res.status(200).send(users);  
 }).catch((error) => {  
 res.status(500).send();  
 });  
});  
  
app.get("/users/:id", (req, res) => {  
 ***User***.findById(req.params.id).then((users) => {  
 res.status(200).send(users);  
 }).catch((error) => {  
 res.status(500).send();  
 });  
});  
  
app.post("/users", async (req, res) => {  
 const user = new ***User***({name: req.query.name, email: req.query.email, age: req.query.age, password: req.query.password})  
  
 await user.save().then(() => {  
 res.status(201).send(user);  
 }).catch((error) => {  
 res.status(500).send;  
 });  
});  
  
app.get("/tasks", (req, res) => {  
 ***Task***.find({}).then((tasks) => {  
 res.status(200).send(tasks);  
 }).catch((error) => {  
 res.status(500).send();  
 });  
});  
  
app.get("/tasks/:id", (req, res) => {  
 ***Task***.findById(req.params.id).then((tasks) => {  
 res.status(200).send(tasks);  
 }).catch((error) => {  
 res.status(500).send();  
 });  
});  
  
app.post("/tasks", async (req, res) => {  
 const task = new ***Task***({description: req.query.description, completed: req.query.completed})  
  
 await task.save().then(() => {  
 res.status(201).send(task);  
 }).catch((error) => {  
 res.status(500).send;  
 });  
});  
  
  
app.listen(3000, () => {  
 ***console***.log("Server is listening")  
});

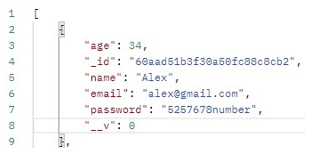


Рис. 1. GET USERS.

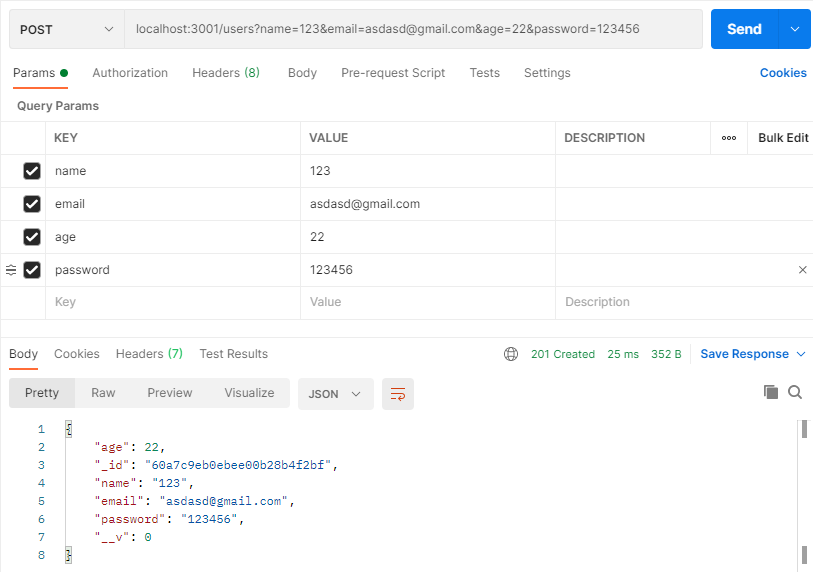


Рис. 2. ADD USER

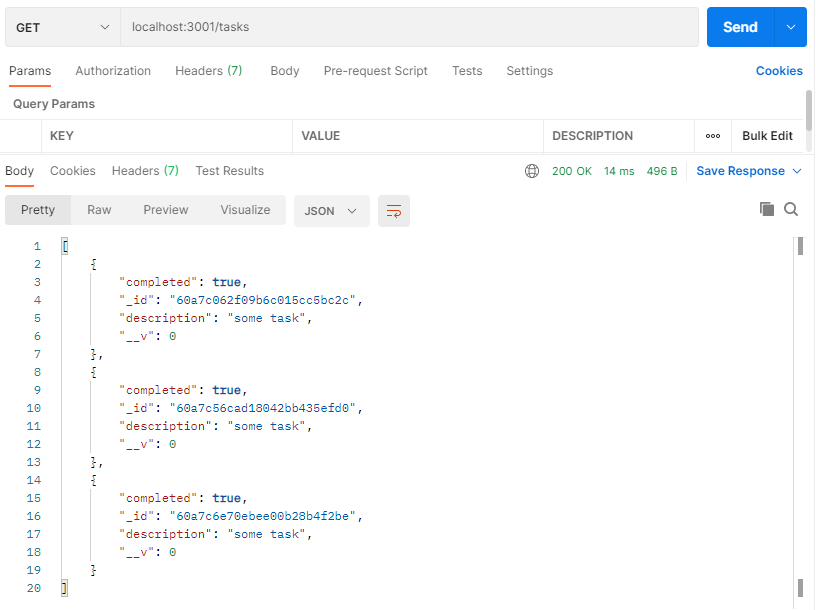


Рис. 3. GET TASK.

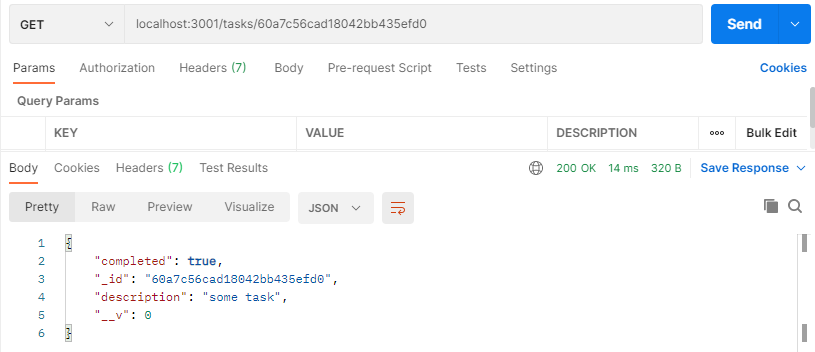


Рис. 4. GET TASK BY ID.

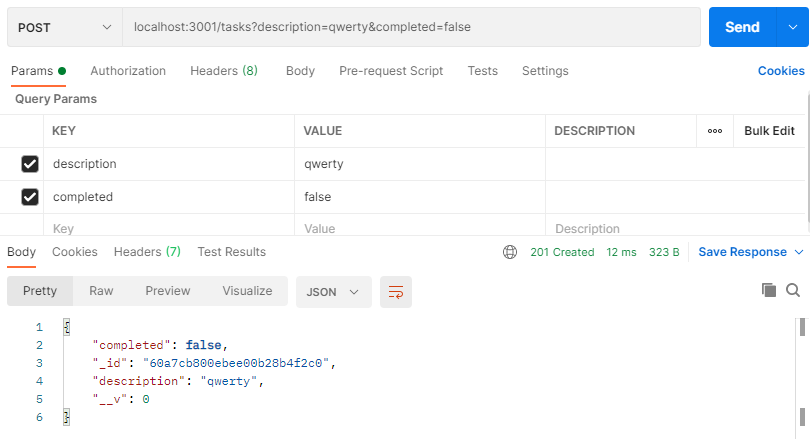


Рис. 5. ADD TASK.

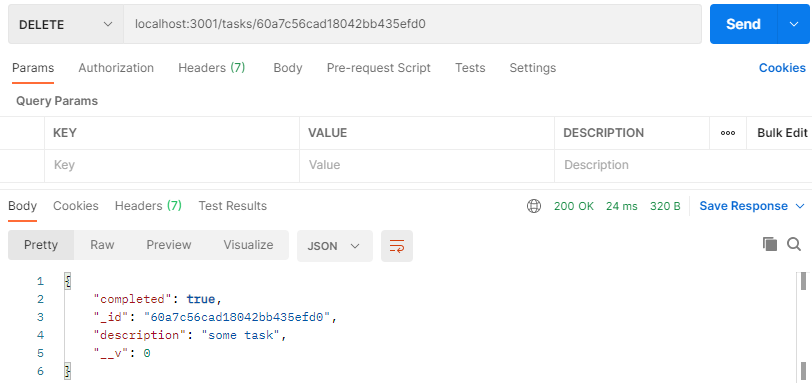


Рис. 6. DELETE TASK.

**Висновок:** В ході виконання роботи я подробніше ознайомився з Node.js, познайомився з фреймворком express, з програмою Postman, і основами mongodb