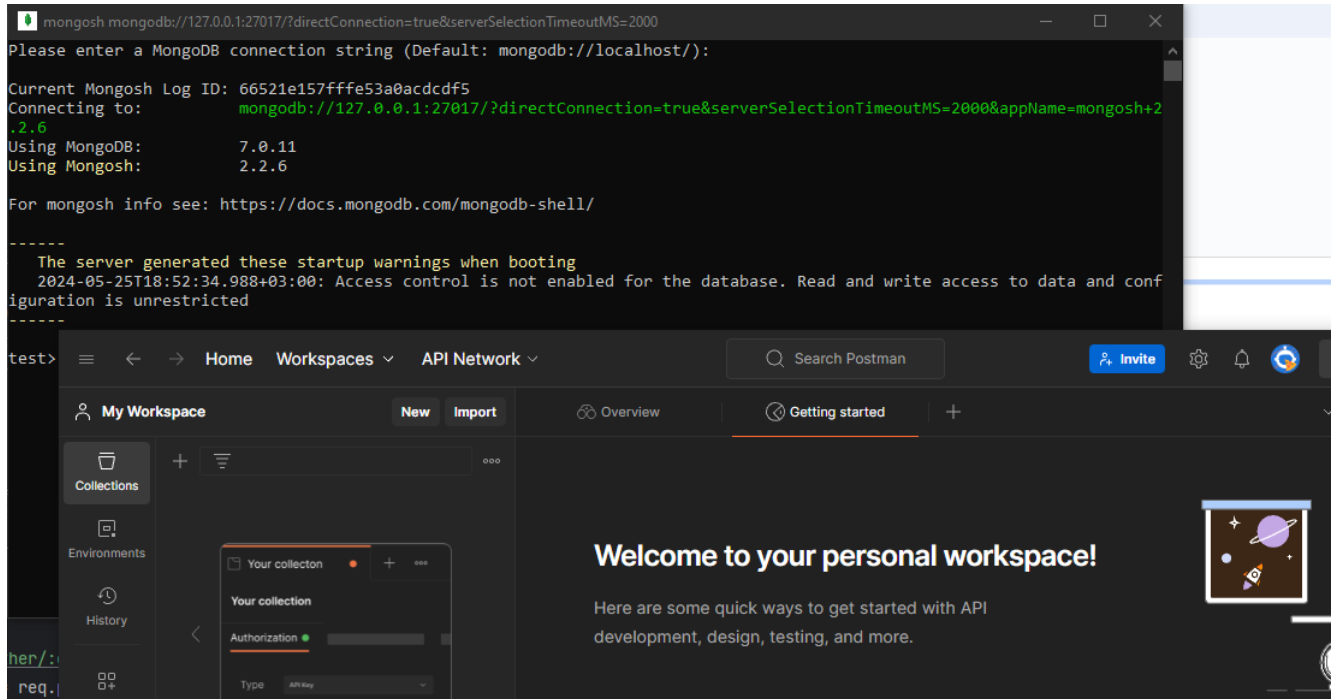


# Лабораторна робота №3

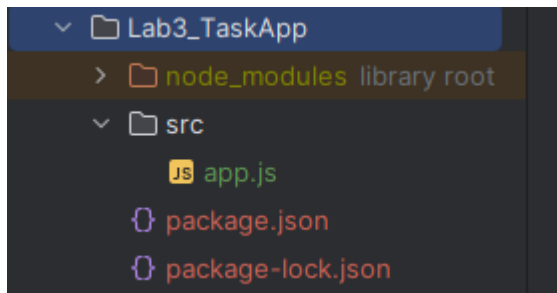
## Mongoose Rest API

Хід роботи :

### 1. Встановити MongoDB + Postman



### 2. Створіть проект TaskApp



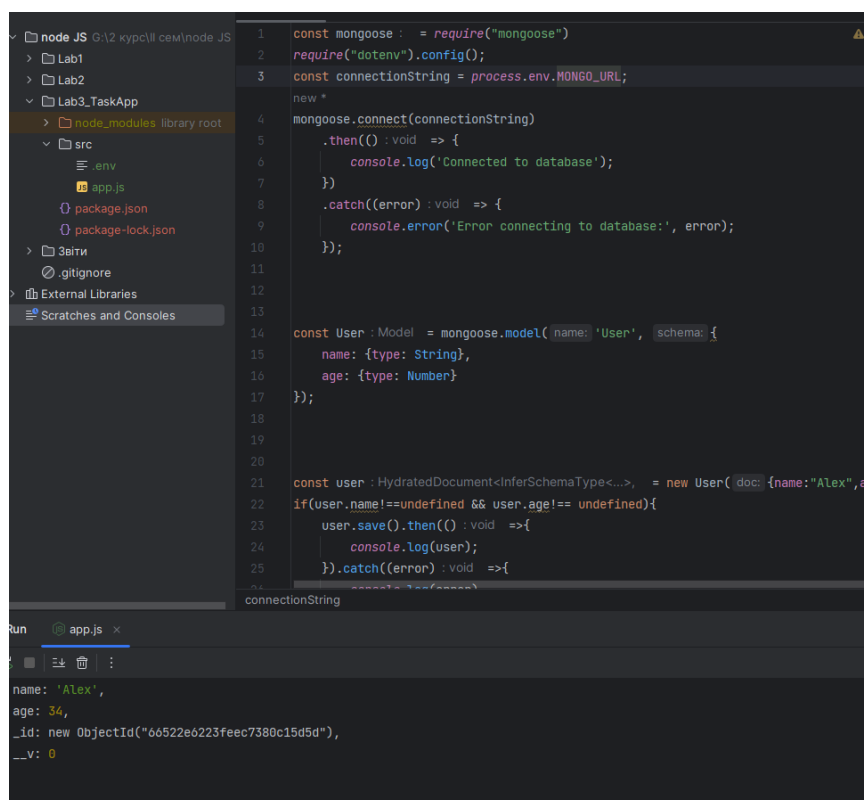
					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.01.000 – Лр.3			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Алієв О.Є			Звіт з лабораторної роботи №3		Лім.	Арк.
Перевір.		Сидорчук В.О						1
Реценз.							ФІКТ, гр.ІПЗ-22-1	
Н. Контр.								
Зав.каф.								

### 3. Підключимось до бази даних та створимо модель User

```
import mongoose from "mongoose";

const User : Model = mongoose.model( name: 'User', schema: {
  name: {type: String},
  age:{type: Number}
});
```

### 4. Створимо екземпляр моделі User



```
1 const mongoose = require("mongoose")
2 require("dotenv").config();
3 const connectionString = process.env.MONGO_URL;
4
5 mongoose.connect(connectionString)
6   .then(() => {
7     console.log("Connected to database");
8   })
9   .catch((error) => {
10    console.error("Error connecting to database:", error);
11  });
12
13
14 const User : Model = mongoose.model( name: 'User', schema: {
15   name: {type: String},
16   age: {type: Number}
17 });
18
19
20
21 const user : HydratedDocument<InferSchemaType<...>, > = new User( doc: {name:"Alex", age:34}, {connectionString});
22 if(user.name!==undefined && user.age!== undefined){
23   user.save().then(() =>{
24     console.log(user);
25   }).catch((error) =>{
26     console.error("Error saving user:", error);
27   });
28 }
```

Run app.js x

```
name: 'Alex',
age: 34,
_id: new ObjectId("60522e6223feec7380c15d5d"),
__v: 0
```



config + trash

local

nodeJSLab3

users ...

```
_id: ObjectId('66522dde2b8e33e7801f468f')
name: "Alex"
age: 34
__v: 0
```

### 5. Валідація даних

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.20.121.01.000 – Лр.3	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

The screenshot shows a VS Code editor with a file explorer on the left containing files like `src`, `.env`, `app.js`, `package.json`, `package-lock.json`, `Звіт`, `.gitignore`, `External Libraries`, and `Scratches and Consoles`. The main editor displays the following JavaScript code:

```

12
13
14 const User : Model = mongoose.model( name
15     name: {
16         type: String,
17         required: true
18     },
19     age: {type: Number}
20 );
21
22
23
24 const user : HydratedDocument<InferSchemaType<...>, new *
25 user.save().then() : void =>{
26     console.log(user);
27 }.catch((error) : void =>{
28     console.log(error)
29 });
30
31
32

```

Below the editor, the `Run` console shows the output of `app.js`:

```

[Symbol(mongoose:validatorError)]: true
}
},
_message: 'User validation failed'
}
Connected to database

```

The screenshot shows a VS Code editor with a file explorer on the left containing files like `src`, `.env`, `app.js`, `package.json`, `package-lock.json`, `Звіт`, `.gitignore`, `External Libraries`, and `Scratches and Consoles`. The main editor displays the following JavaScript code:

```

21 validate(value) : void {
22     if (value < 0){
23         throw new Error("Age must be a positive number")
24     }
25 }
26
27
28
29
30
31 : HydratedDocument<InferSchemaType<...>, = new User( doc: {name:"Alex",age:-1})
32 ).then() : void =>{
33     e.log(user);
34     error) : void =>{
35         e.log(error)
36     }
37 }
38
39
40
41

```

Below the editor, the `Run` console shows the output of `app.js`:

```

properties: [Object],
kind: 'user defined',
path: 'age',
value: -1,
reason: Error: Age must be a positive number

```

```

14
15
16 > const User : Model = mongoose.model( name: 'User', schema: {
17 >   name: {type: String...},
21 >   age: {type: Number...},
29 >   email: {
30     type: String,
31     required: true,
32     trim: true,
33     validate: {
34       validator: (value) => {
35         return validator.isEmail(value);
36       }
37     }
38   }
39 });
40
41
42
43 const user : HydratedDocument<InferSchemaType<...>, = new User( doc: {name
44   new *
45   new save() then() - void => {
User > email

```

```

[Symbol(mongoose:validatorError)]: true
}
,
message: 'User validation failed'
nnected to database

```

## 6. Фільтрація даних (Data Sanitization)

```

name: {
  type: String,
  required: true,
  trim: true
},
age: {
  type: Number,
  default: 0,
  validate(value) : void {
    if (value < 0) {
      throw new Error('Age must be a positive number')
    }
  }
},
email: {
  type: String,
  required: true,
  trim: true,
  lowercase: true,
  unique: true,
  validate: {
    validator: (value) => {
      return validator.isEmail(value);
    },
    message: 'Error: Email is invalid'
  }
}

```

## 7. Завдання для моделі User

```
password: {
  type: String,
  minLength: 7,
  required: true,
  trim: true,
  validate: {
    validator: (value) => {
      return !value.toLowerCase().includes('password');
    },
    message: 'Password can\'t have value "password" or contain it'
  }
}
```

The screenshot shows a VS Code editor with two files open: `app.js` and `user.js`. The `user.js` file contains a Mongoose schema for a `User` model. The `app.js` file shows the application logic, including database connection and saving a new user.

```
1 const mongoose = require("mongoose")
2 const User = require("../models/user")
3
4
5 require("dotenv").config({ path: "../.env" });
6 const connectionString = process.env.MONGO_URL;
7
8 mongoose.connect(connectionString)
9   .then(() => {
10     console.log('Connected to database');
11   })
12   .catch((error) => {
13     console.error('Error connecting to database:', error);
14   });
15
16 const user : HydratedDocument<InferSchemaType = new User( doc: {name: 'Omar', age: 19, email: 'omaraaliiev7729@gmail.com',
17   new *
18   user.save().then(() => {
19     console.log(user);
20   }).catch((error) => {
21     console.log(error)
22   });
23
24
25
```

The Run console shows the output of the application:

```
connected to database
name: 'Omar',
age: 19,
email: 'omaraaliiev7729@gmail.com',
password: '112233445566',
_id: new ObjectId("6652382fc5b077ca02de8c43"),
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.20.121.01.000 – Лр.3	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

_id: ObjectId('6652382fc5b077ca02de8c43')
name : "Omar"
age : 19
email : "omaraliev7729@gmail.com"
password : "112233445566"
__v : 0

```

## 8. Завдання для моделі Task

```

JS app.js JS task.js x JS user.js
1  const { default: mongoose : } = require("mongoose");
2  const validator : {__esModule?: any, default?: any} = require('validator');
3  //Create model
4  const Task : Schema<any, Model<...>, {...}, {...}, {...}, {...}, DefaultSchemaOptions, = new mongoose.Schema({
5      title: {
6          type:String,
7          required:true,
8          trim: true
9      },
10     description: {
11         type:String,
12         required:true,
13         trim: true
14     },
15     completed: {
16         type: Boolean,
17         default: false
18     }
19 })
20 const someTask : Model = mongoose.model('Task', Task);
21 module.exports = someTask;

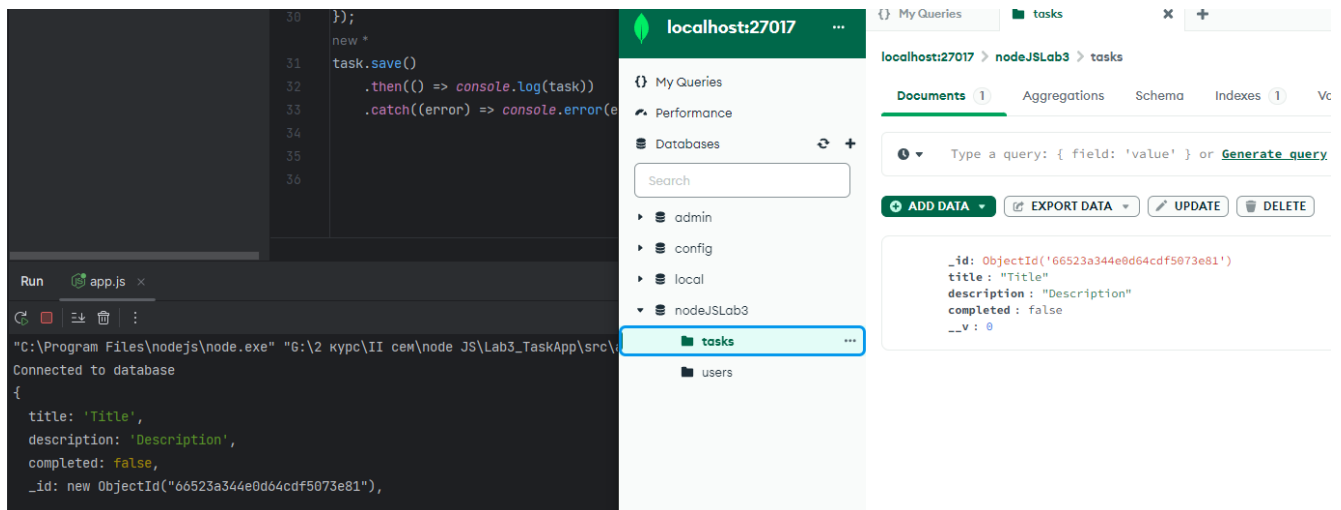
```

```

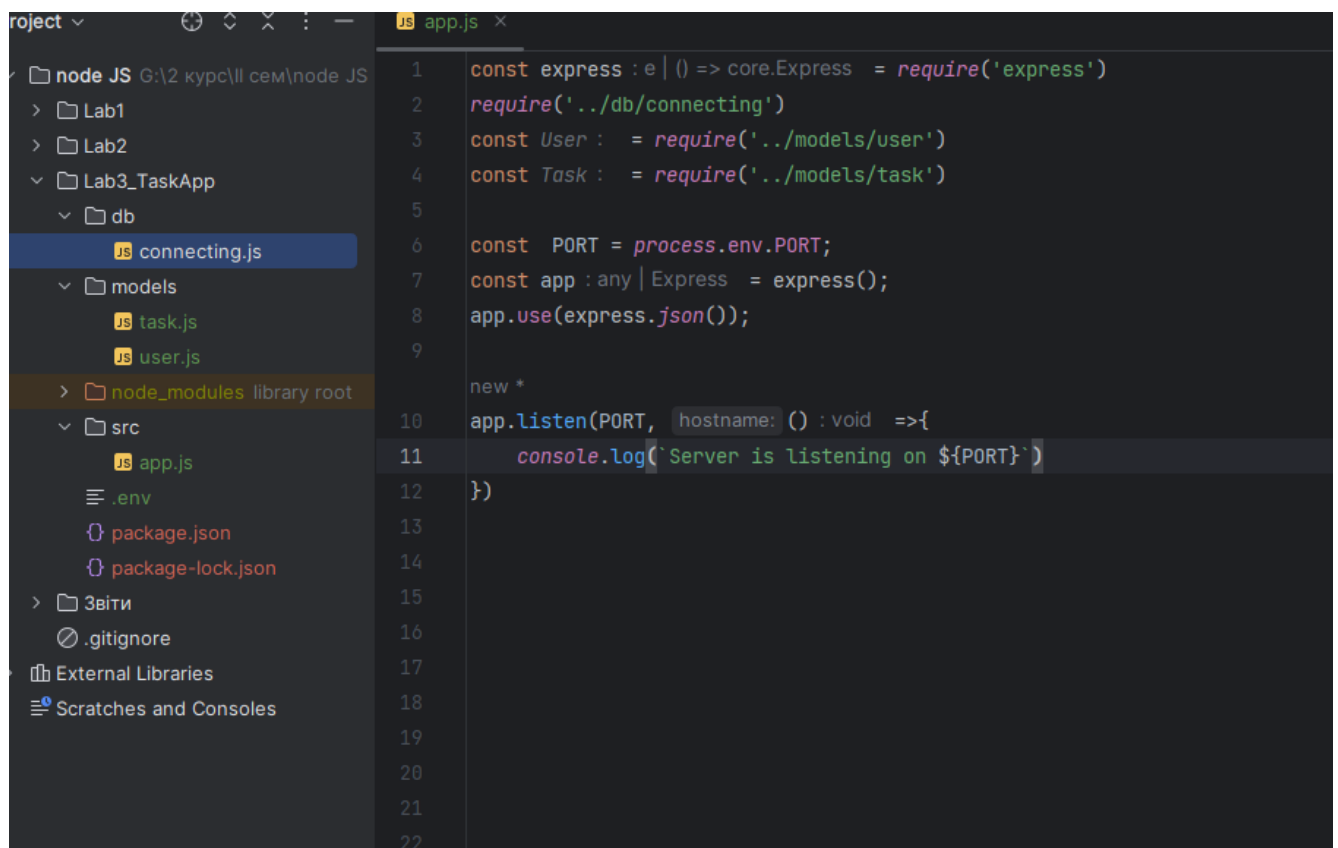
node_modules library root 23     .then(() => console.log(user))
src 24     .catch((error) => console.error(error));*/
env 25
package.json 26     const task : HydratedDocument<InferSchemaType = new Task( doc: {
package-lock.json 27         title:'Title',
28         description: 'Description',
29         completed: false
30     });
31     new *
32     task.save()
33         .then(() => console.log(task))
34         .catch((error) => console.error(error));
35
36

```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.20.121.01.000 – Лр.3	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6



## 9. Структуруйте проект



## 10. Обробка GET-запиту

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.20.121.01.000 – Лр.3	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

```

1  const express = e | () => core.Express = require('express')
2  require('../db/connecting')
3  const User = require('../models/user')
4  const Task = require('../models/task')
5
6  const PORT = process.env.PORT;
7  const app = any | Express = express();
8  app.use(express.json());
9
10 new *
11 app.listen(PORT, {hostname: () : void =>{
12   console.log('Server is listening on ${PORT}')
13 }}
14
15 new *
16 app.get('/users', (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> ,res : Response<ResBody, LocalsObj> )
17   User.find({}).then( onfulfilled: (users : (HydratedDocument<...>[] ) : void =>{
18     res.status( code: 200).send(users);
19   }).catch((error) : void =>{
20     res.status( code: 500).send;
21   });
22 });

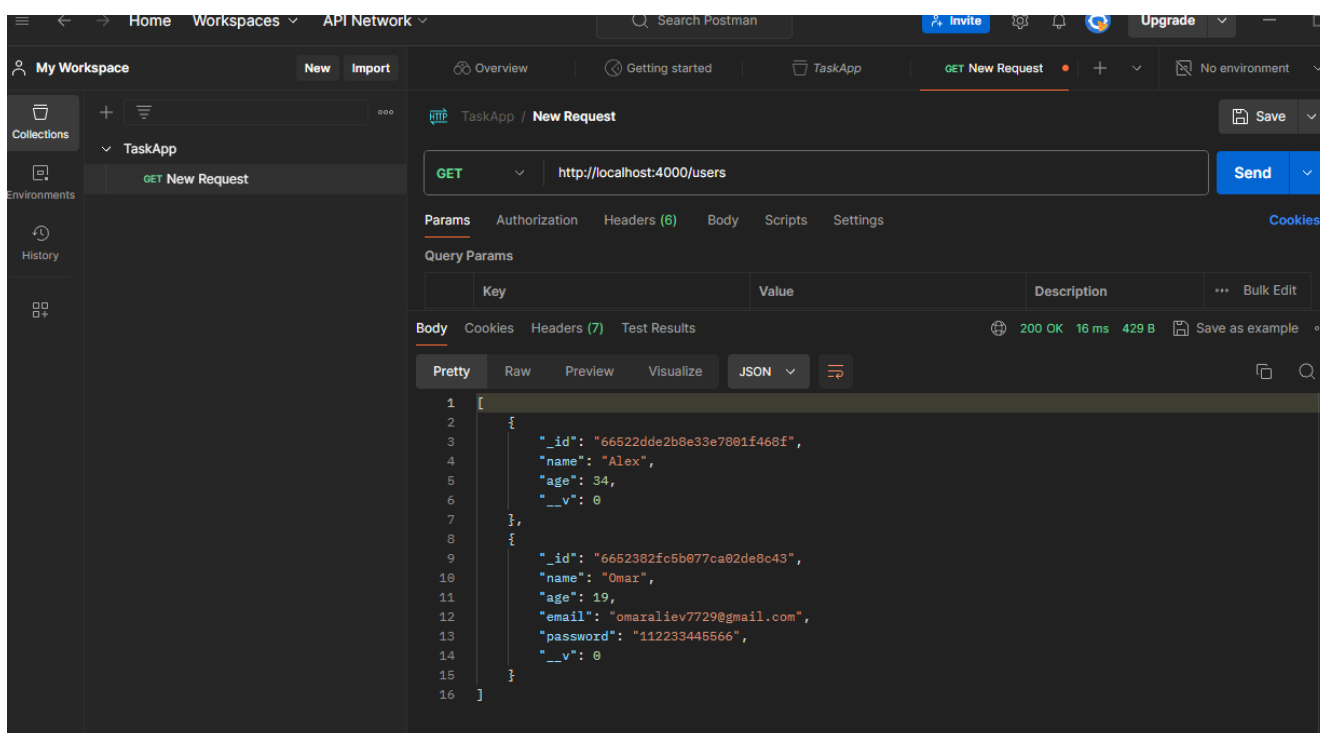
```

```

[
  {
    "_id": "66522dde2b8e33e7801f468f",
    "name": "Alex",
    "age": 34,
    "__v": 0
  },
  {
    "_id": "6652382fc5b077ca02de8c43",
    "name": "Omar",
    "age": 19,
    "email": "omaraliev7729@gmail.com",
    "password": "112233445566",
    "__v": 0
  }
]

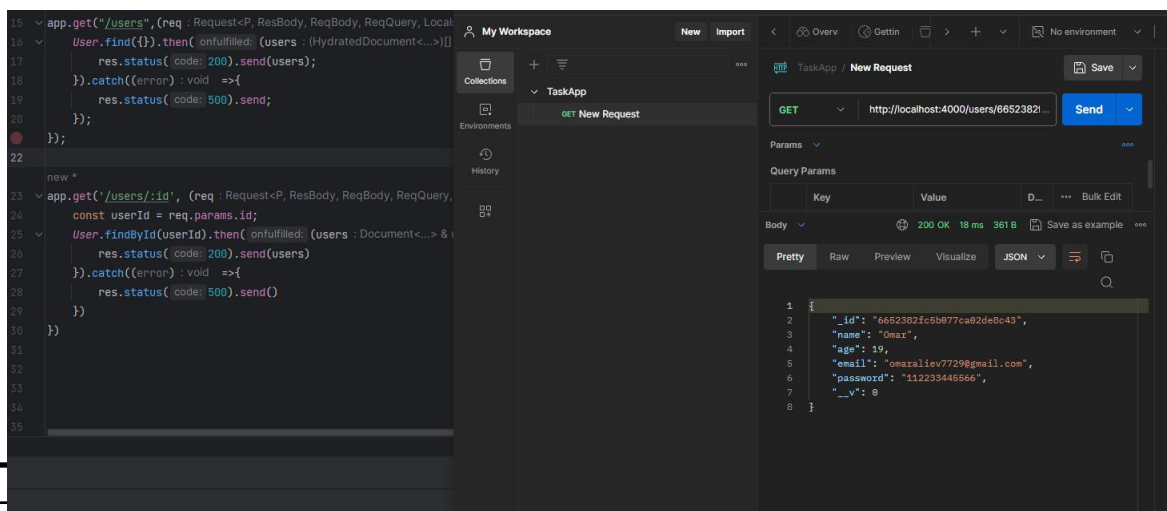
```

11. З допомогою програми POSTMAN протестуйте виконання даного запиту

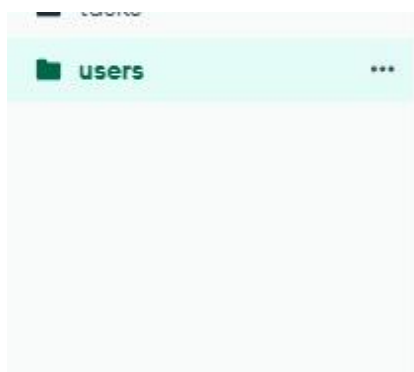
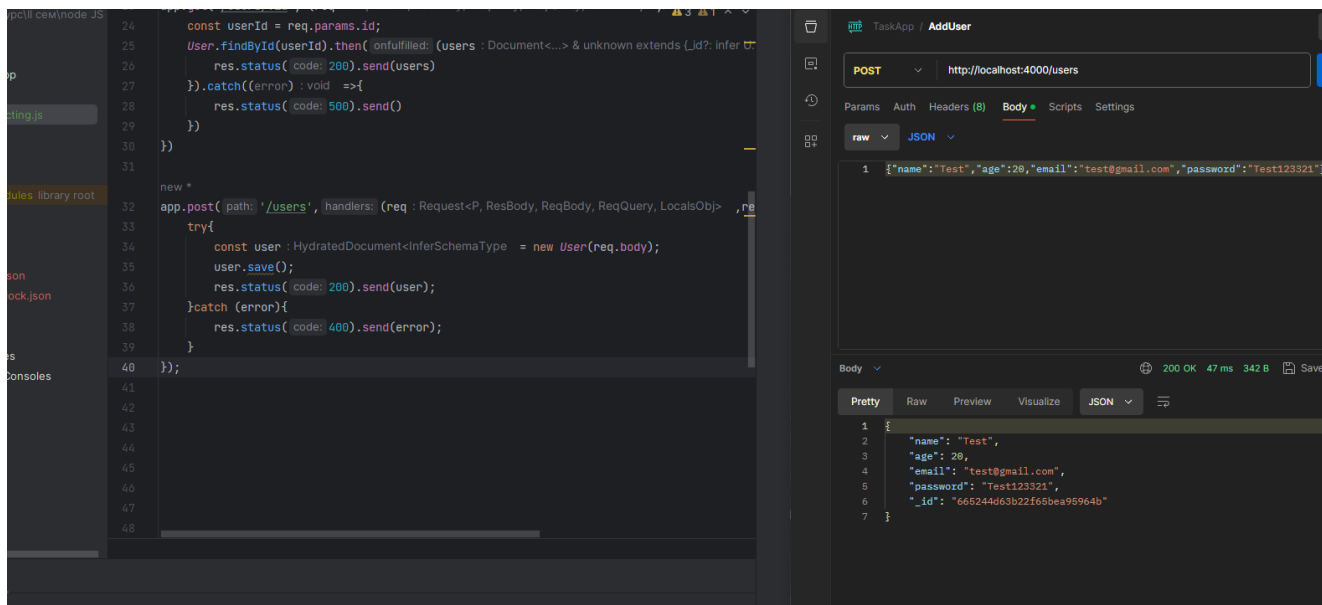


12. Створіть обробку запитів

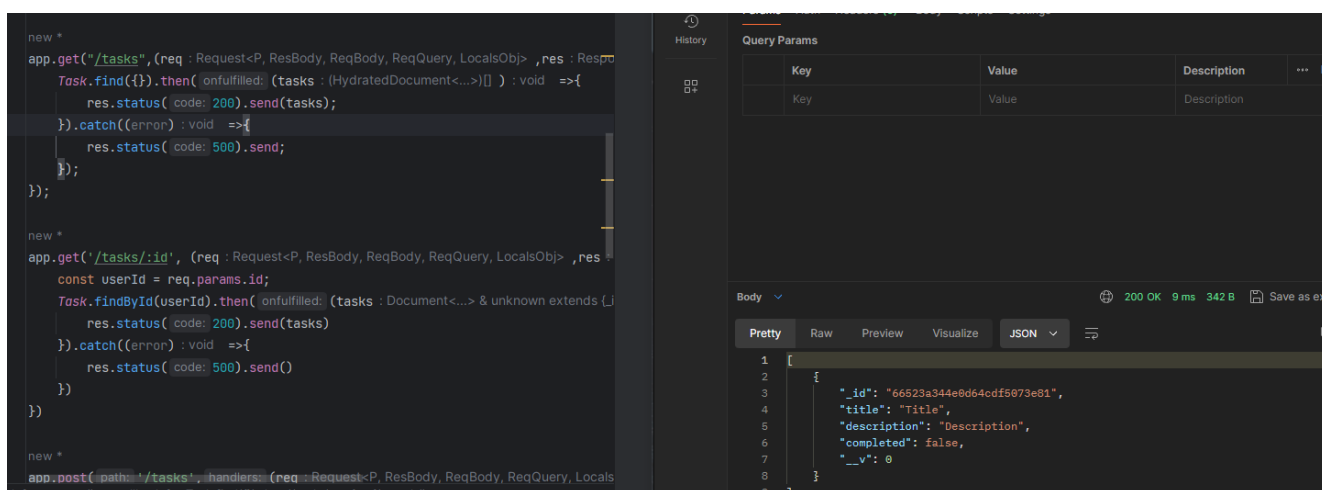
User



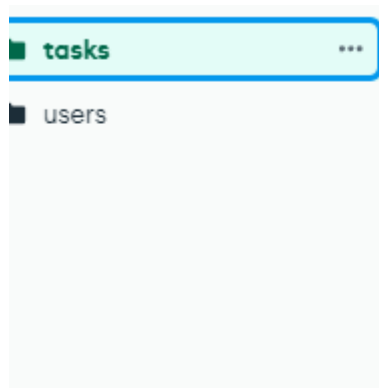




## Task







\_\_v : 0

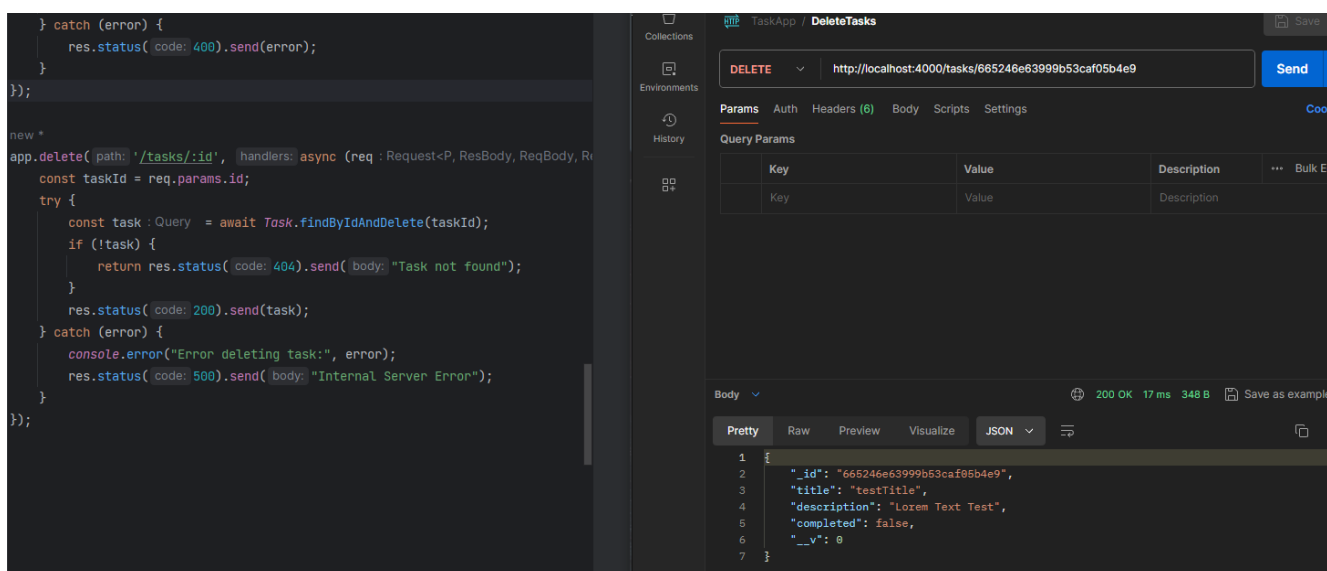
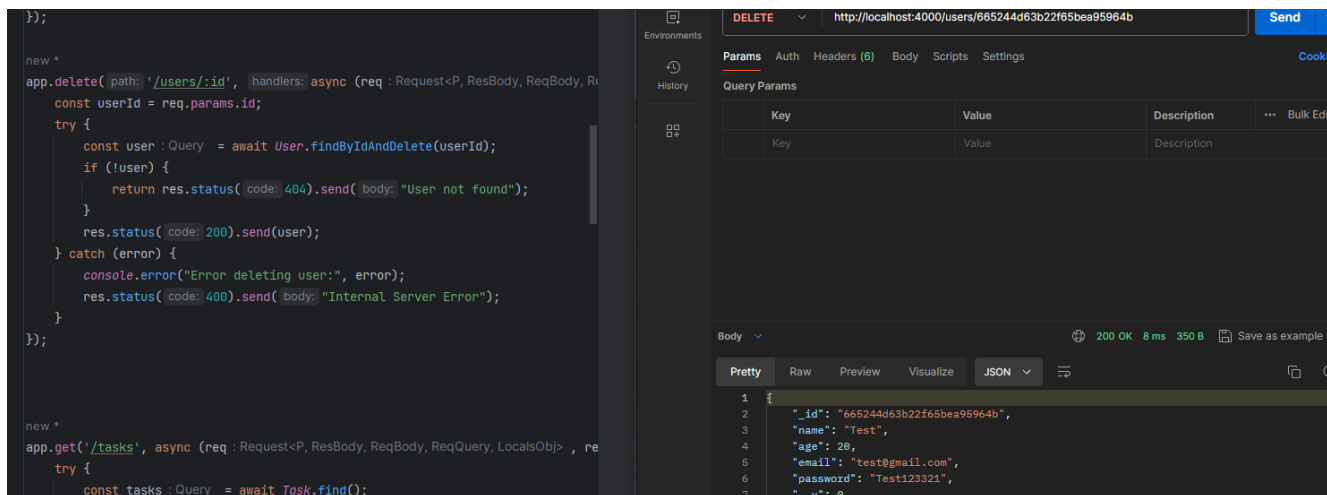
```
_id: ObjectId('665246e63999b53caf05b4e9')
title: "testTitle"
description: "Lorem Text Test"
completed: false
__v: 0
```

13. Здійсніть рефакторинг коду в додатку Task Application з використанням *async/await*

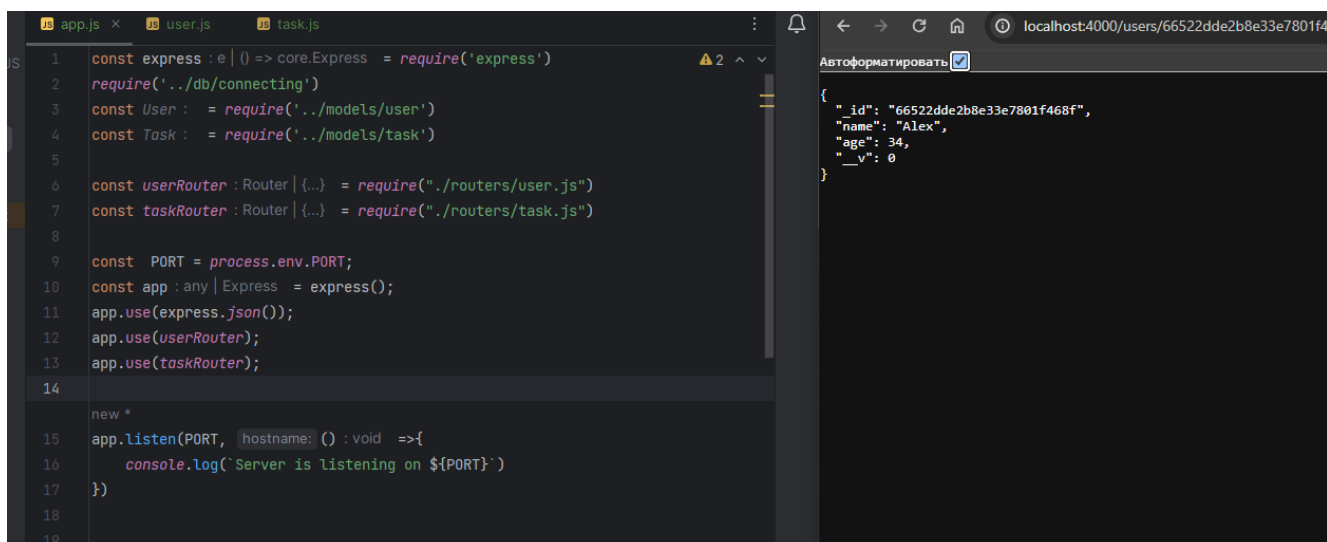
```
✓ app.post( path: '/users', handlers: async (req : Request<P, ResBody, ReqBody, ReqQue
✓   try {
      const user : HydratedDocument<InferSchemaType> = new User(req.body);
      await user.save();
      res.status( code: 200 ).send(user);
    } catch (error) {
      res.status( code: 400 ).send(error);
    }
  });

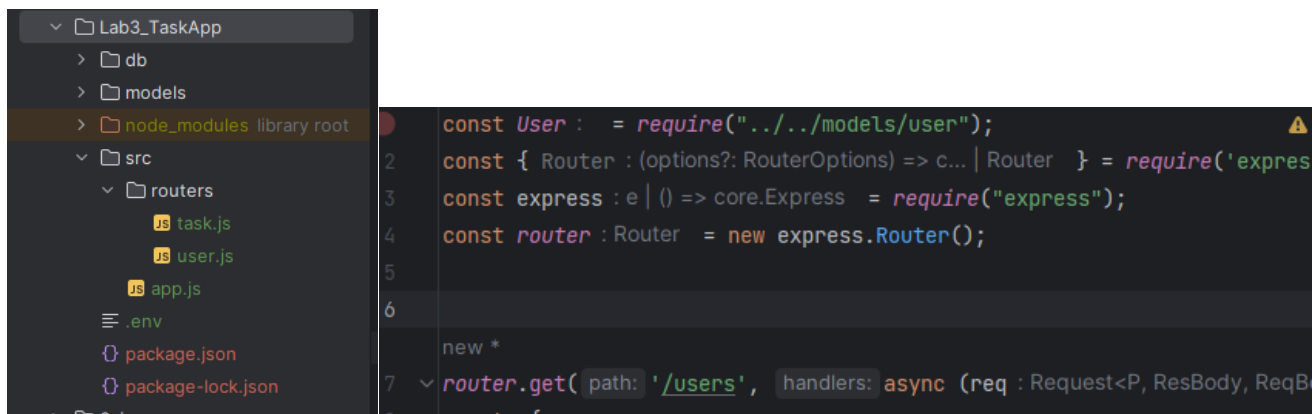
new *
✓ app.get('/tasks', async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , r
✓   try {
      const tasks : Query = await Task.find();
      res.send(tasks);
    } catch (error) {
      res.status( code: 500 ).send(error);
    }
  });
```

14. Створіть обробку запиту видалення по id для моделей User і Task



## 15. Separate Route Files (маршрутизатори)





Висновок : на лабораторному занятті ми ознайомились з Mongoose та Rest API

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.20.121.01.000 – Лр.3	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		