

Лабораторна робота №5.

Зв'язування користувачів і задач

Мета: навчитись зв'язувати дані. Завершення виконання TaskApp

Завдання

- Створити відношення між користувачами та задачами
- Забезпечити роботу користувача лише з власними задачами (GET /tasks, GET /tasks/:id, ...)
- Приховати захищені дані (password, tokens)

Завдання 1. Зв'язок між користувачами та задачами

- Реалізуйте можливість доступу та обробки лише задач поточного користувача

Крок 1. В моделі Task створіть поле для зберігання id користувача

```
    default: false
  },
  owner: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true
  }
}
```

					ДУ «Житомирська політехніка».24.121.13.000 - Лр5			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Клосович І.А.			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Сидорчук В.О.						Аркушів
Керівник							1	18
Н. контр.							ФІКТ Гр. ІІЗ-22-1[1]	
Зав. каф.								

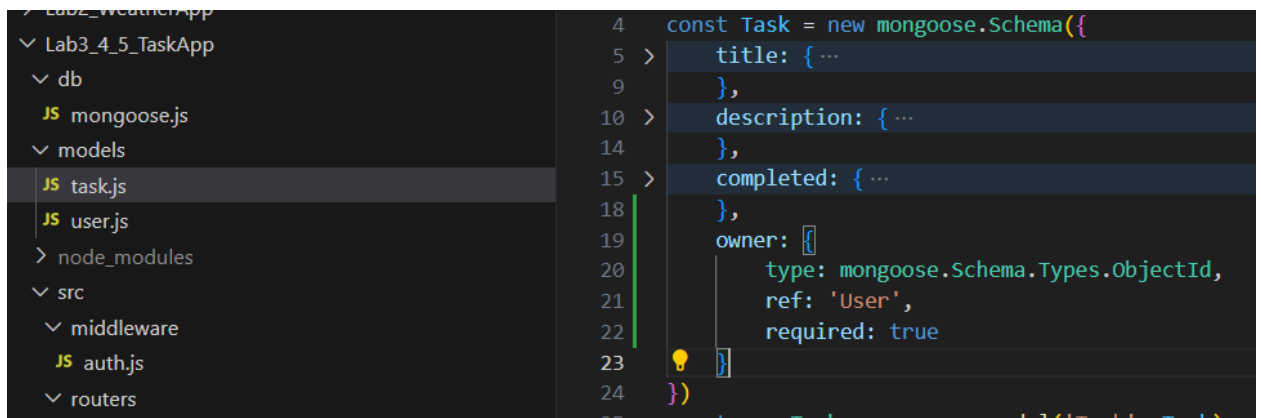


Рис. 1. Результат

Крок 2. В маршрутизаторі підключіть auth та модифікуйте обробник додавання нового запису

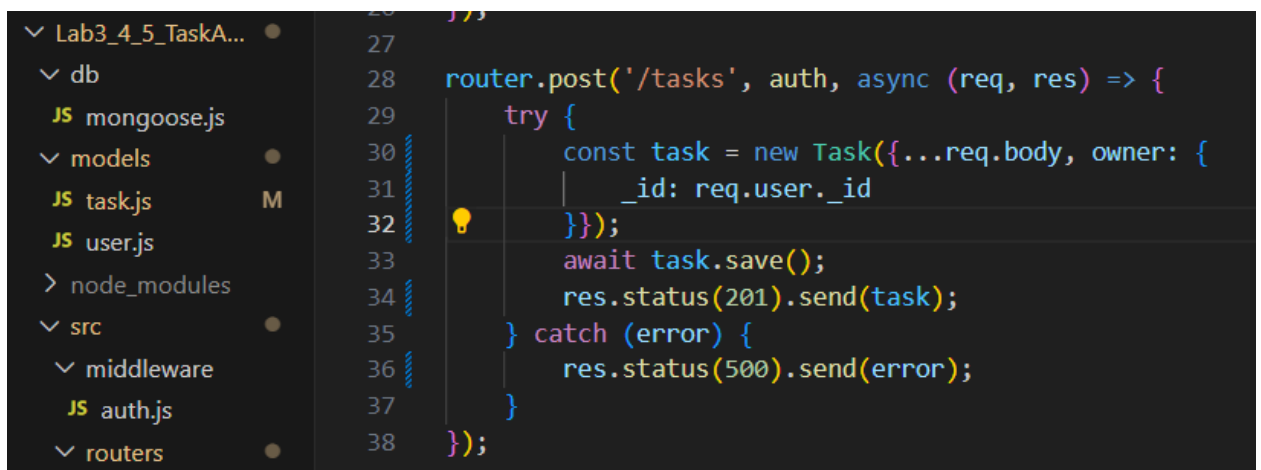


Рис. 2. Результат

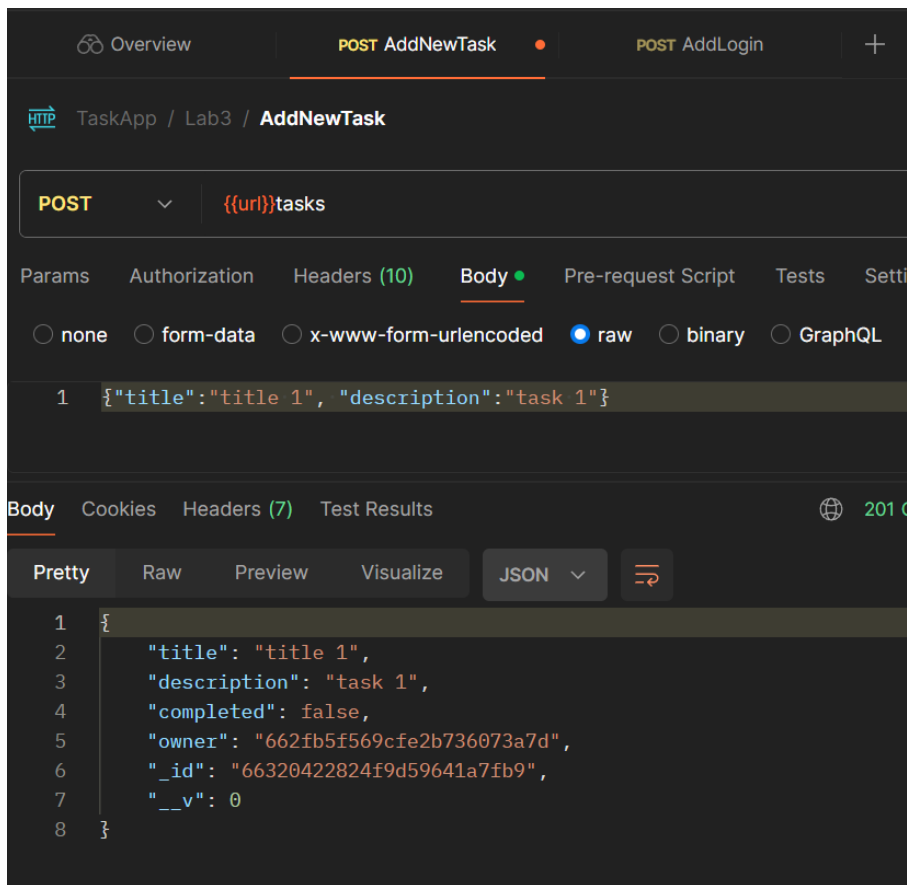


Рис. 3. Результат

Крок 3. Для отримання запису користувача, якому належить дана задача:

● Приклад коду:

```
const task = await Task.findById('5cd2975ff1194728dcc41774');
await task.populate('owner').execPopulate();
```

В нових версіях mongoose: `await task.populate('owner');`

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

Lab2_WeatherApp
Lab3_4_5_TaskA...
db
JS mongoose.js
models
JS task.js
JS user.js
node_modules
src
middleware
JS auth.js

17
18
19
20
21
22
23
24
25
26
27
28

```
router.get('/tasks/:id', auth, async (req, res) => {  
  try {  
    const task = await Task.findById(req.params.id)  
    await task.populate('owner')  
    res.status(200).send(task)  
  }  
  catch (error) {  
    res.status(500).send(error)  
  }  
});
```

Рис. 4. Результат

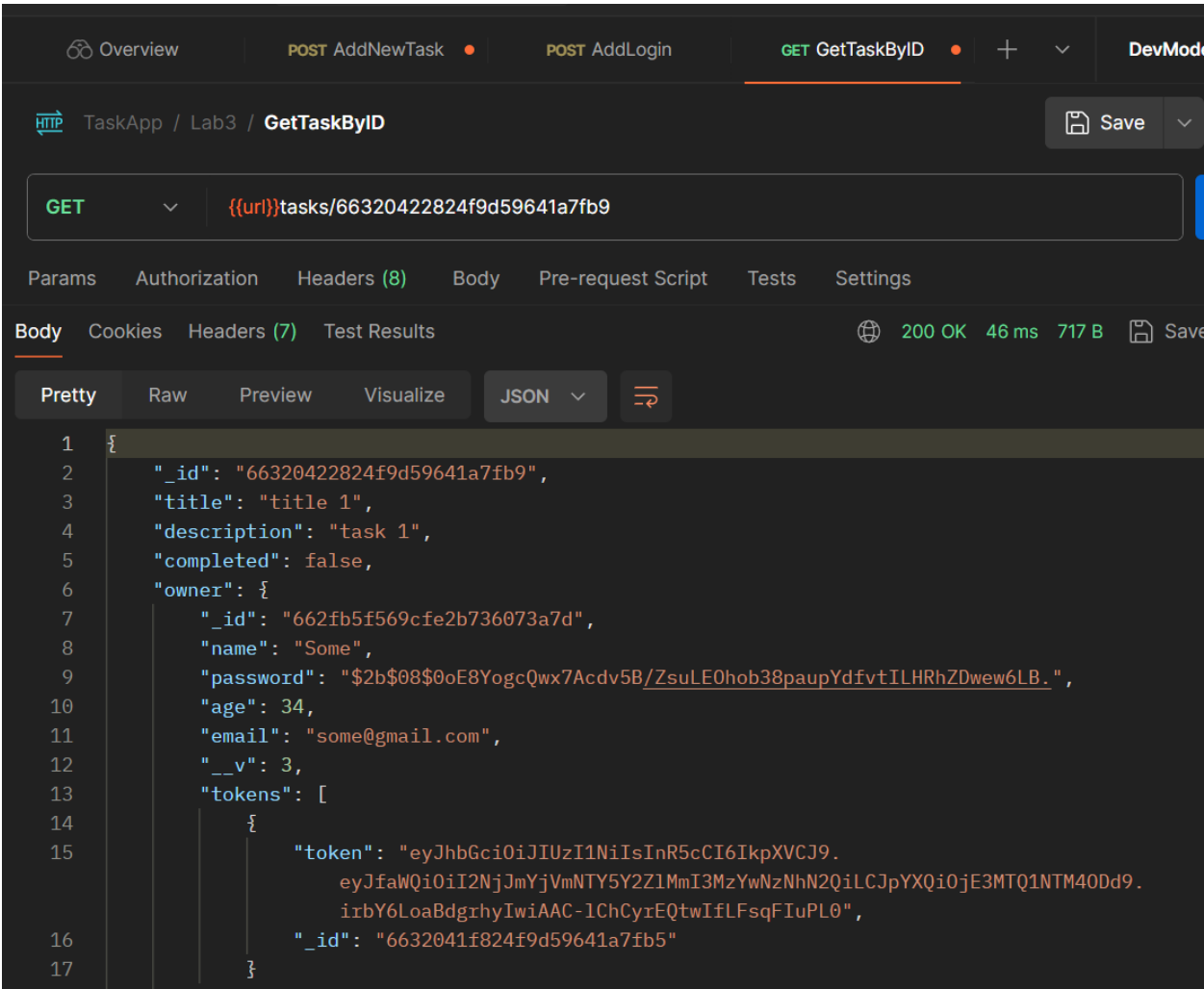


Рис. 5. Результат

Крок 4. Для отримання задач, які належать даному користувачу, створіть віртуальне поле в моделі User. Відобразіть дані поля tasks в профілі користувача (запит /users/me)

```
userSchema.virtual('tasks', {
  ref: "Task",
  localField: '_id',
  foreignField: 'owner'
})
```

```
const userSchema = new mongoose.Schema({
  firstName: {type: String...},
  lastName: {type: String...},
  age: {type: Number...},
  email: {type: String...},
  password: {type: String...},
  tokens: [{...}]
}, {toJSON: {virtuals: true}})
```

- Приклад коду для виконання:

```
const user = await User.findById('5cd29752f1194728dcc41772');
await user.populate("tasks").execPopulate();
```

В нових версіях mongoose: `await user.populate('tasks');`

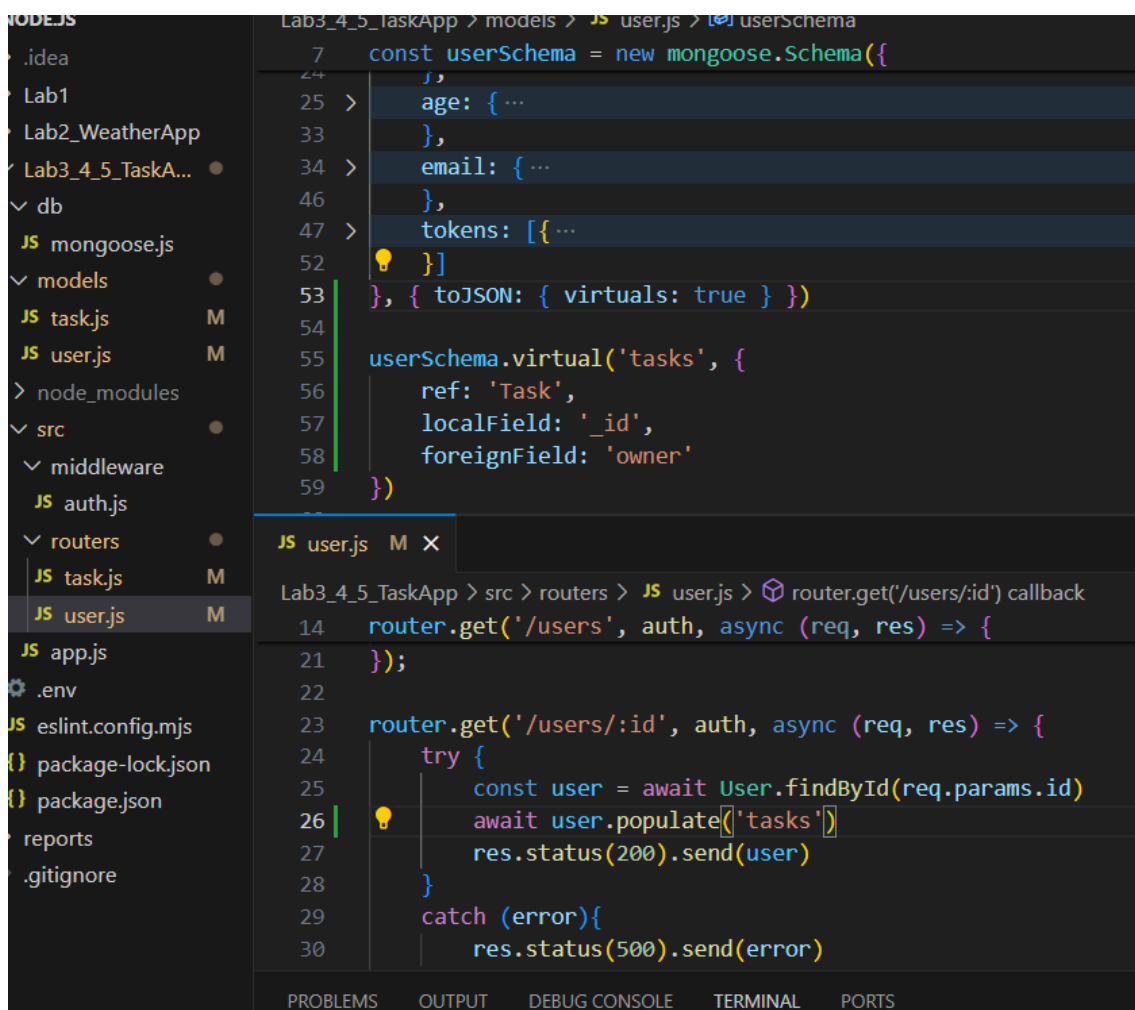


Рис. 6. Результат

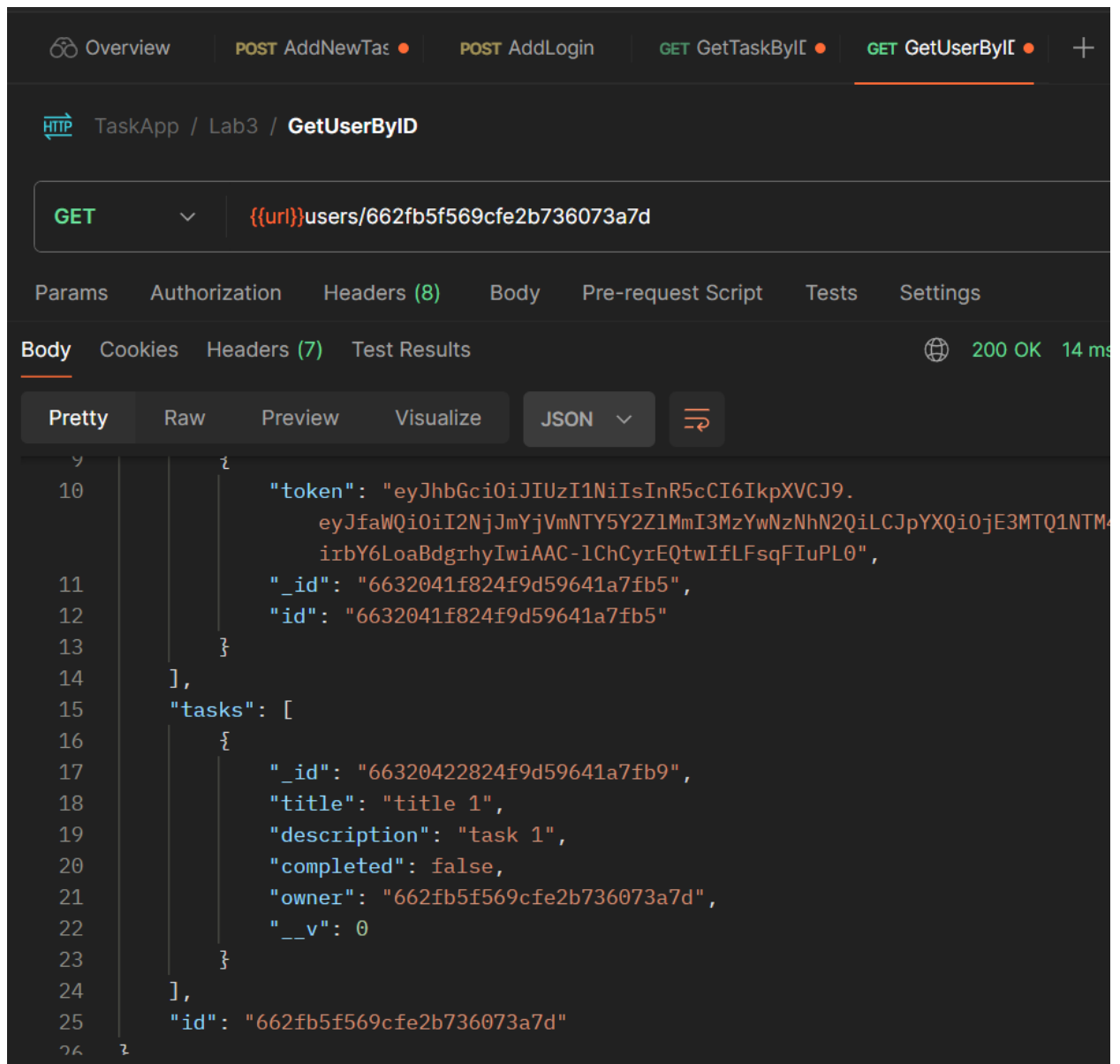


Рис. 7. Результат

Завдання 2. Реалізувати можливість роботи користувачам лише з власними завданнями

- Реалізуйте отримання задачі по її id лише для користувача, якому належить дана задача. Для авторизованого користувача, якому не належить дана задача, надіслати помилку 404.
- Модифікуйте метод отримання списку задач, лише таких, що належать даному користувачу
- Створіть методи редагування/видалення задачі лише тої, що належить даному користувачу. Надіслати 404, якщо задачі не існує або вона не належить поточному користувачеві

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

6
7 router.get('/tasks', auth, async (req, res) => {
8   try {
9     const tasks = await Task.find({ owner: {
10       _id: req.user._id
11     }});
12     res.send(tasks);
13   } catch (error) {
14     res.status(500).send(error);
15   }
16 });
17
18 router.get('/tasks/:id', auth, async (req, res) => {
19   try {
20     const task = await Task.findById(req.params.id)
21     if (!task) {
22       throw new Error("Not existing")
23     }
24     const belongs = task.owner._id.toString() === req.user._id.toString()
25     if (!belongs) {
26       return res.status(404).send({
27         error: "IdError",
28         message: "This task is not yours"
29       })
30     }
31     await task.populate('owner')
32     res.status(200).send(task)
33   }
34   catch (error) {
35     res.status(500).send(error)
36   }

```

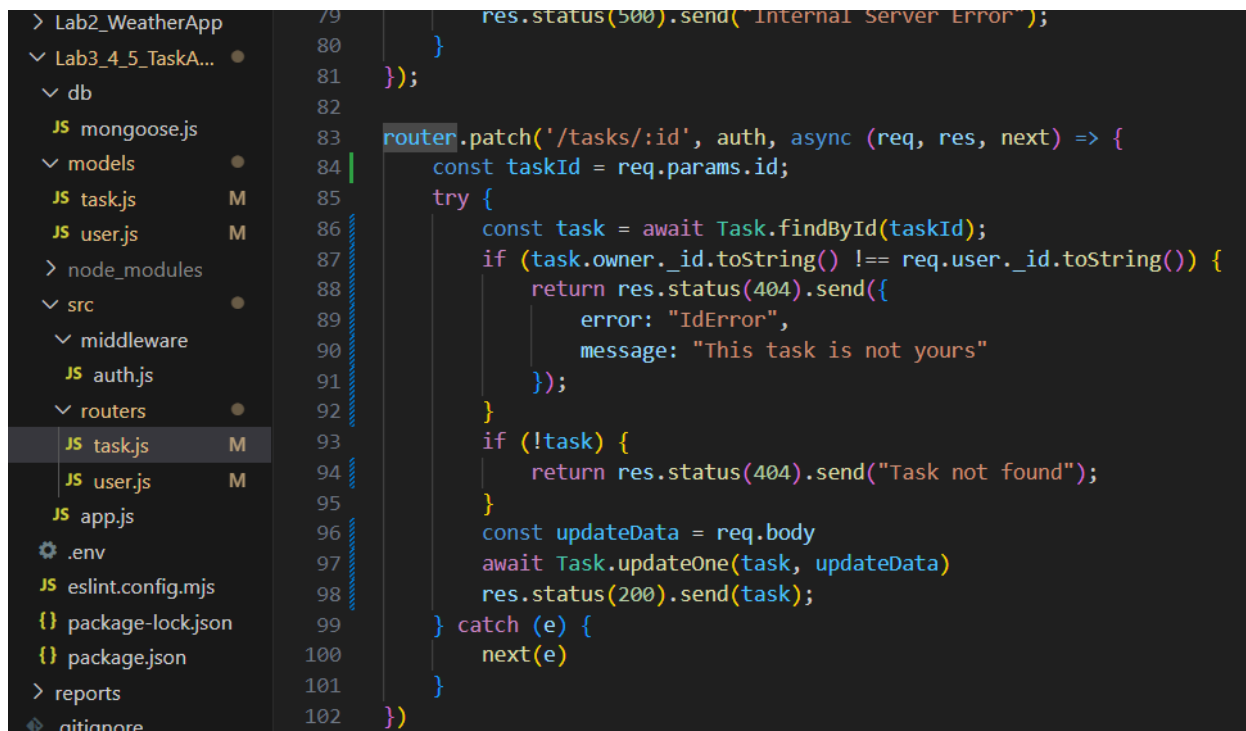
Рис. 8. Результат

```

51
52 router.delete('/tasks/:id', auth, async (req, res) => {
53   const taskId = req.params.id;
54   try {
55     const task = await Task.findById(taskId);
56     if (task.owner._id.toString() !== req.user._id.toString()) {
57       return res.status(404).send({
58         error: "IdError",
59         message: "This task is not yours"
60       });
61     }
62     if (!task) {
63       return res.status(404).send("Task not found");
64     }
65     await Task.deleteOne(task)
66     res.status(200).send(task);
67   } catch (error) {
68     console.error("Error deleting task:", error);
69     res.status(500).send("Internal Server Error");
70   }
71 });
72

```

Рис. 9. Результат



```

79     res.status(500).send("Internal Server Error");
80   }
81   });
82
83   router.patch('/tasks/:id', auth, async (req, res, next) => {
84     const taskId = req.params.id;
85     try {
86       const task = await Task.findById(taskId);
87       if (task.owner._id.toString() !== req.user._id.toString()) {
88         return res.status(404).send({
89           error: "IdError",
90           message: "This task is not yours"
91         });
92       }
93       if (!task) {
94         return res.status(404).send("Task not found");
95       }
96       const updateData = req.body
97       await Task.updateOne(task, updateData)
98       res.status(200).send(task);
99     } catch (e) {
100       next(e)
101     }
102   })

```

Рис. 10. Результат

Послідовність тестів

1. Реєструємо два користувача user1, user2
2. Список задач. Повинна бути помилка 403 (не авторизовано)
3. Логін user1.
4. Створюємо дві задачі і переглядаємо їх. Записуємо їх ідентифікатори.
5. Здійснюємо вихід user1.
6. Заходимо під user2.
7. Створюємо дві задачі. Переглядаємо їх, редагуємо, видаляємо.
8. Спробуємо переглянути або змінити задачі user1. Повинна бути помилка 404.
9. Видаляємо всі задачі і здійснюємо вихід.
10. Здійснюємо вхід під user1. Виводимо задачі user1. Вони повинні бути на місці.

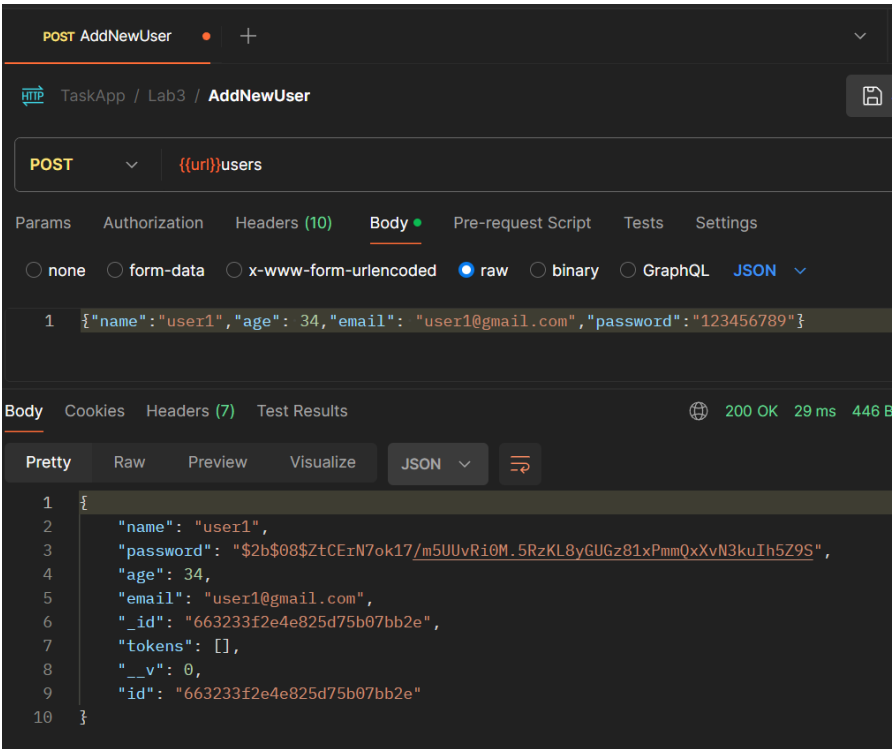


Рис. 11. Результат реєстрації user1(аналогічно створюється user2)

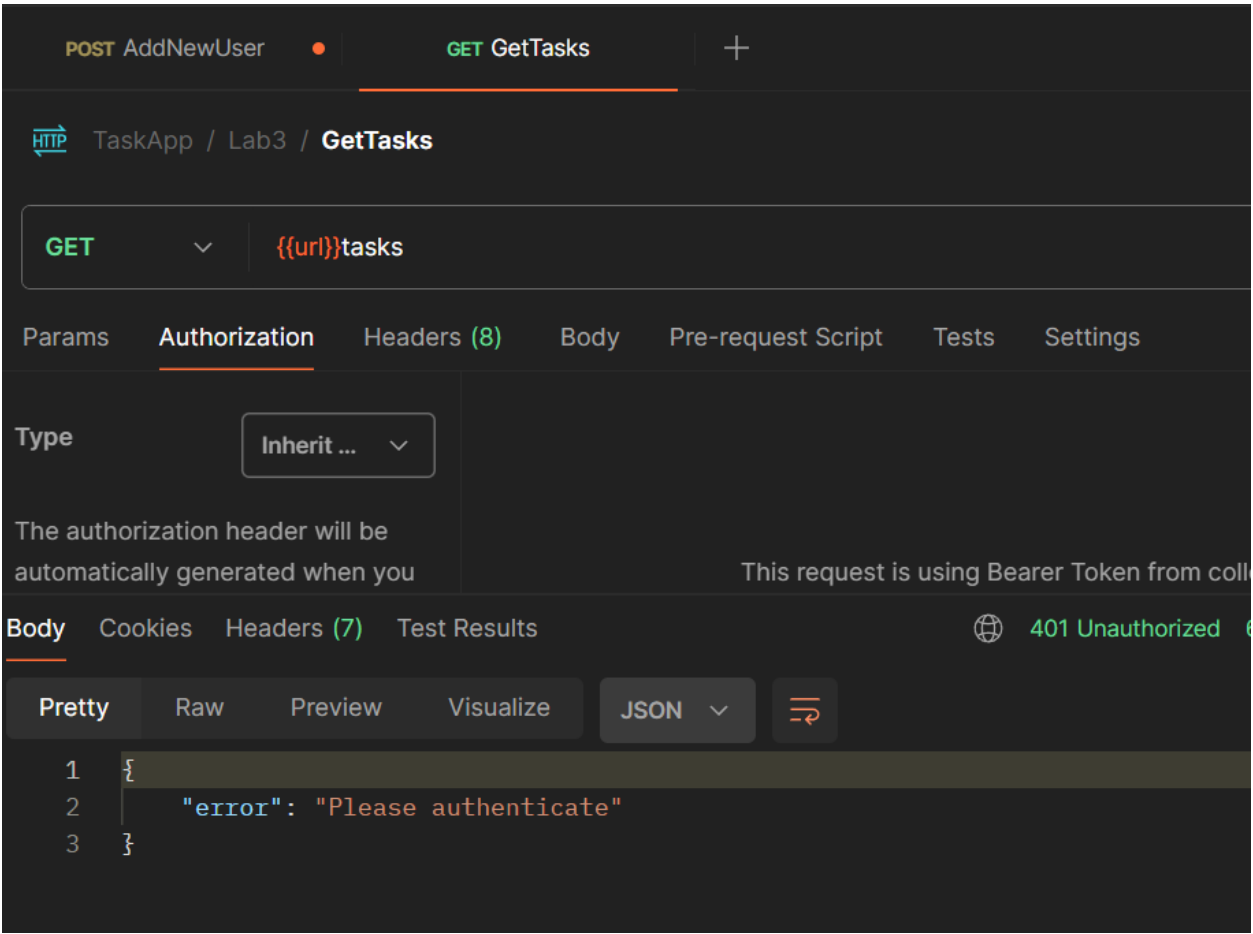


Рис. 12. Список задач

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

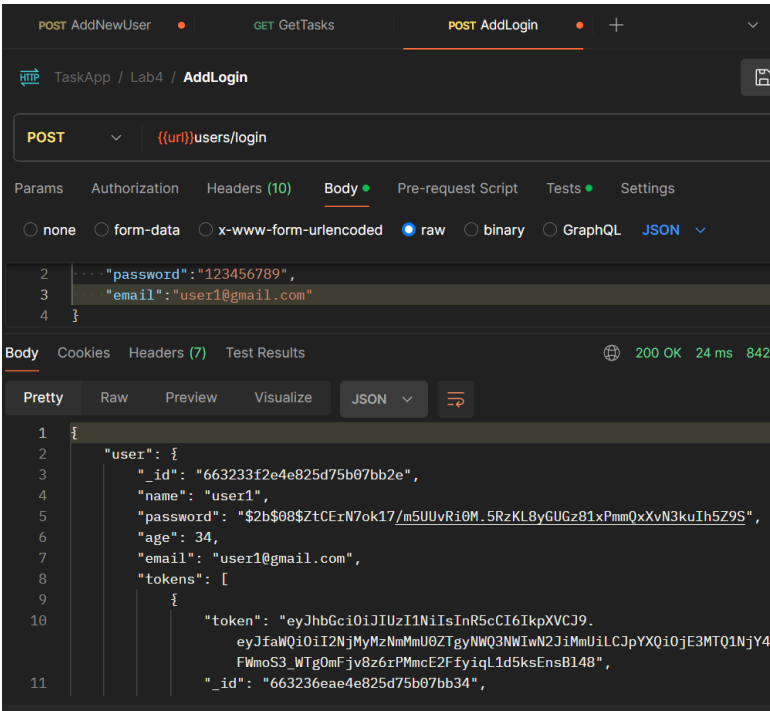


Рис. 13. Логін user1

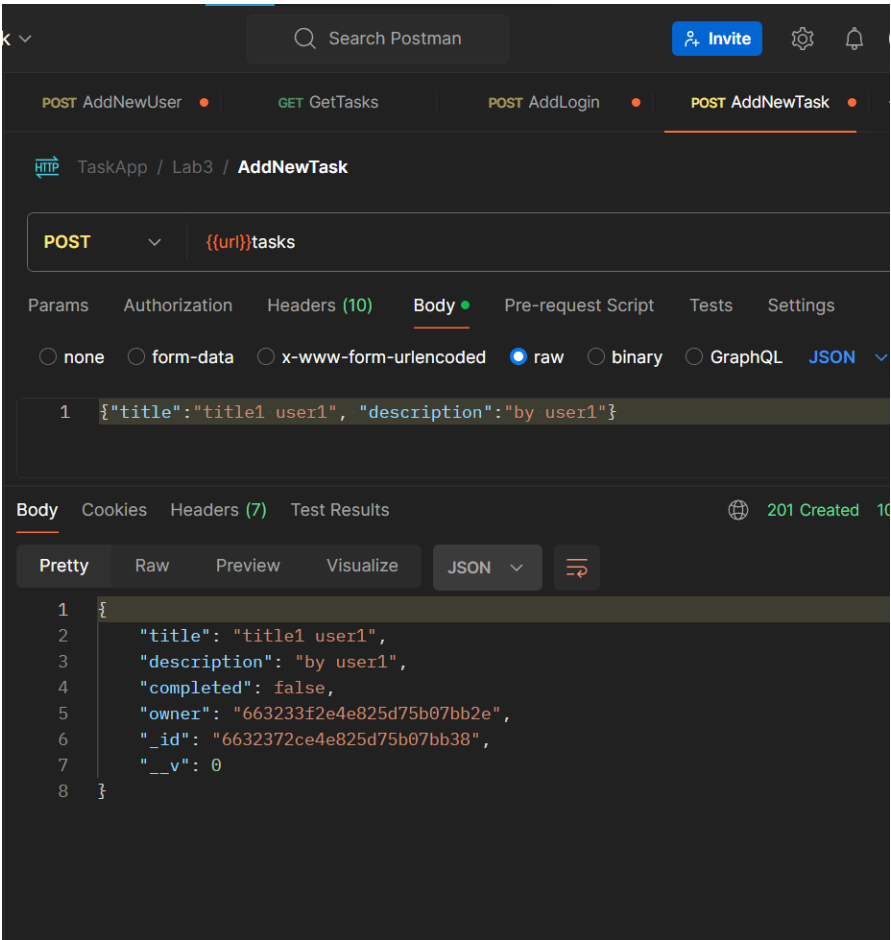


Рис. 14. Створення задачі user1(і ще одної)

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

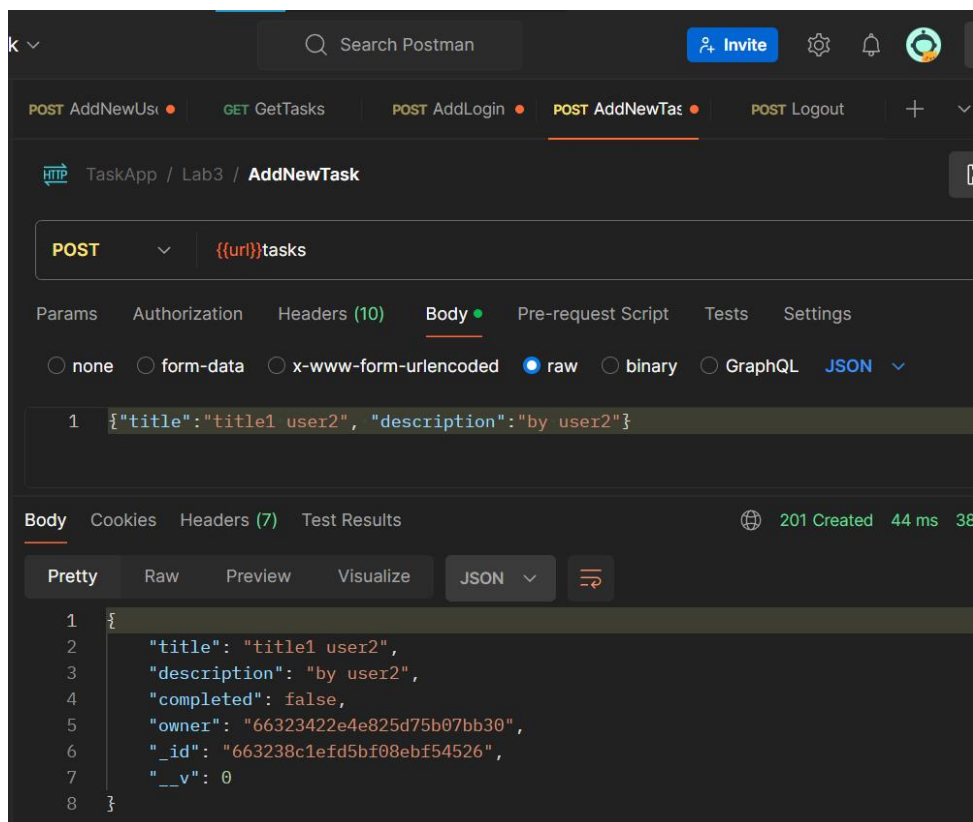


Рис. 17. Створення задачі для user2(+ще одну окремо)

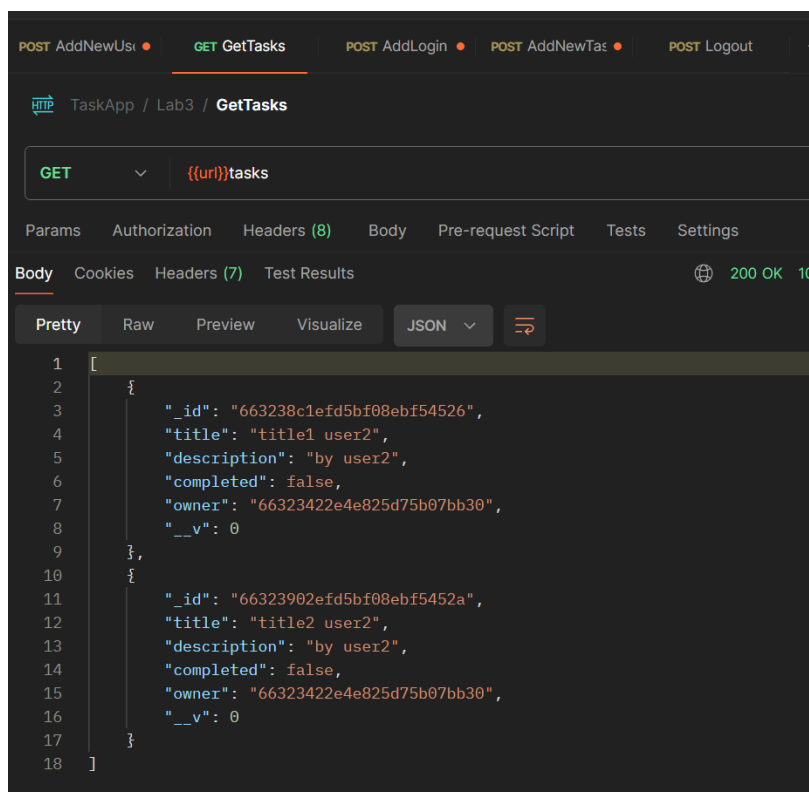


Рис. 18. Перегляд задач тільки user2

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

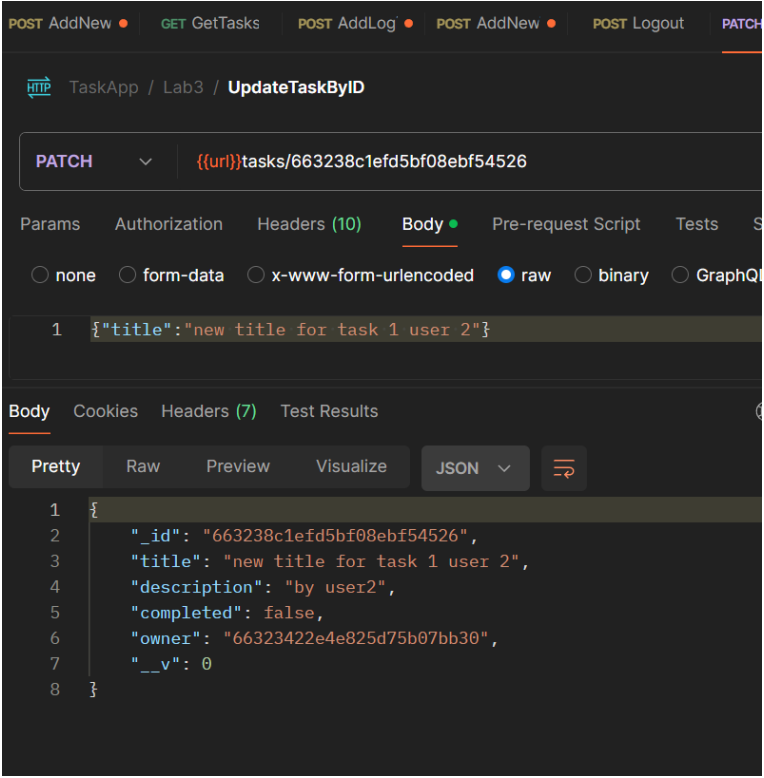


Рис. 19. Редагування першої задачі user2

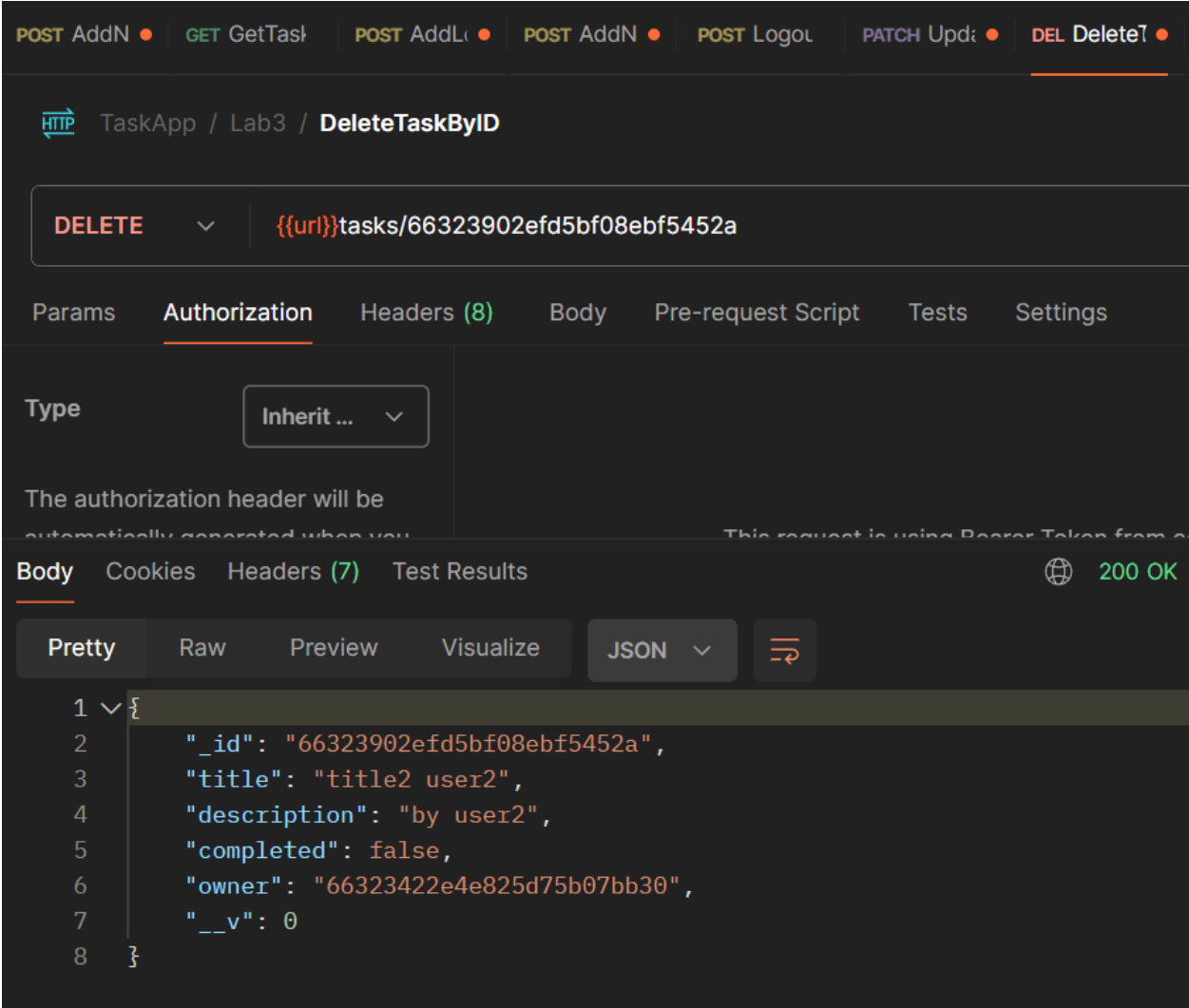


Рис. 20. Результат видалення task2 user2

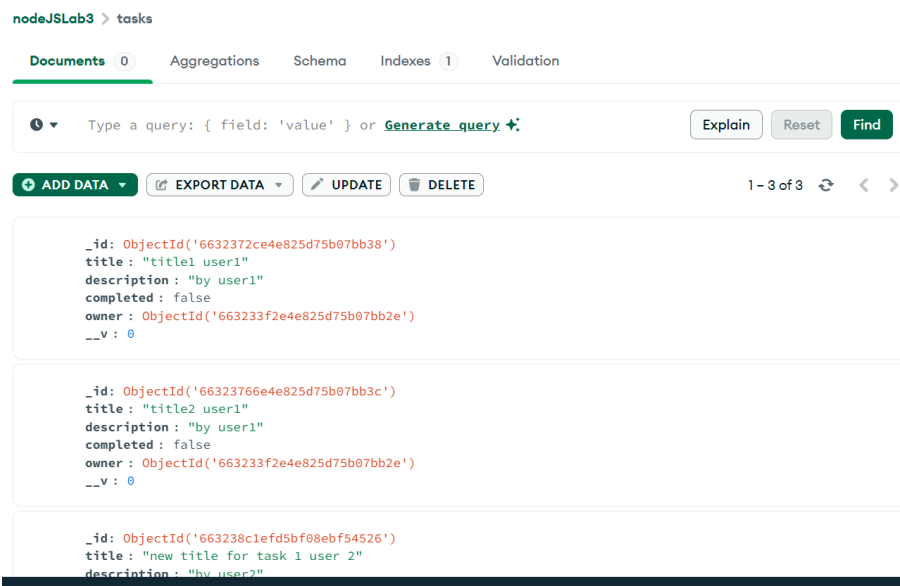


Рис. 21. Результат

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

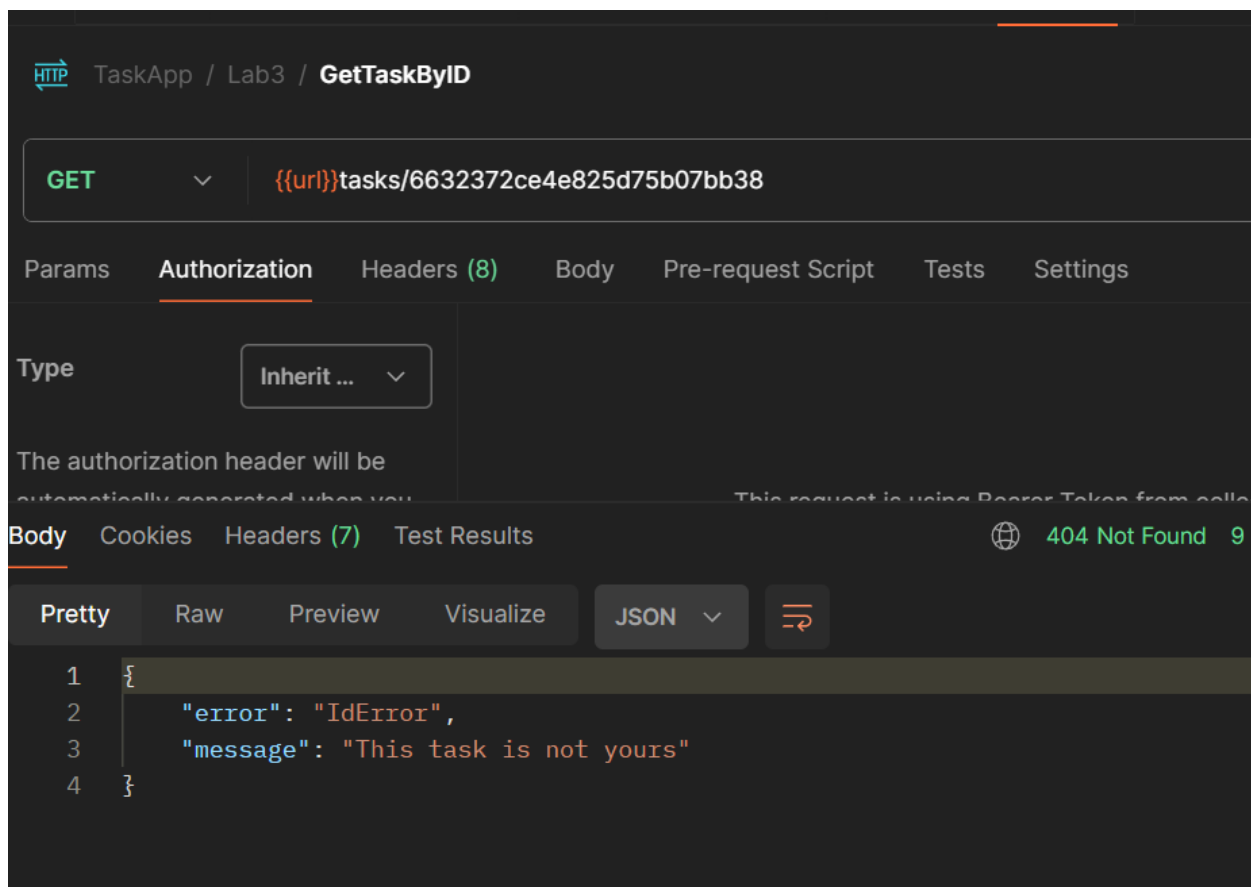


Рис. 22. Перегляд задачі, що належить user1

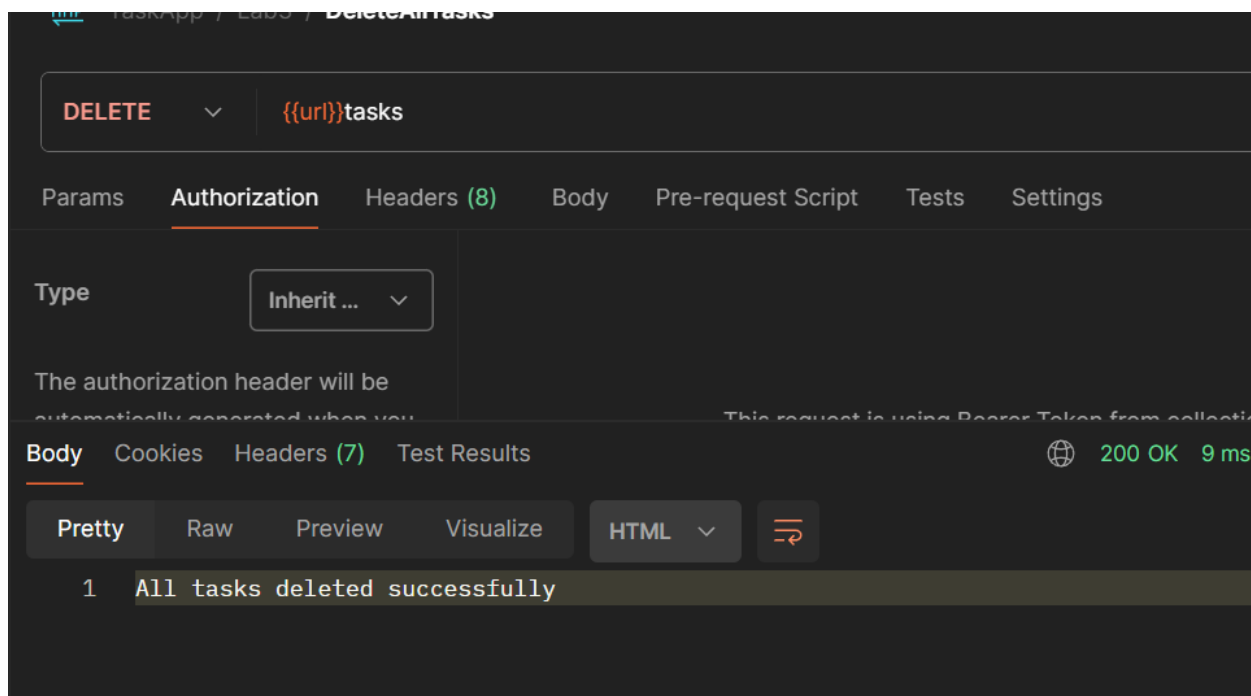


Рис. 23. Видалення всіх задач

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

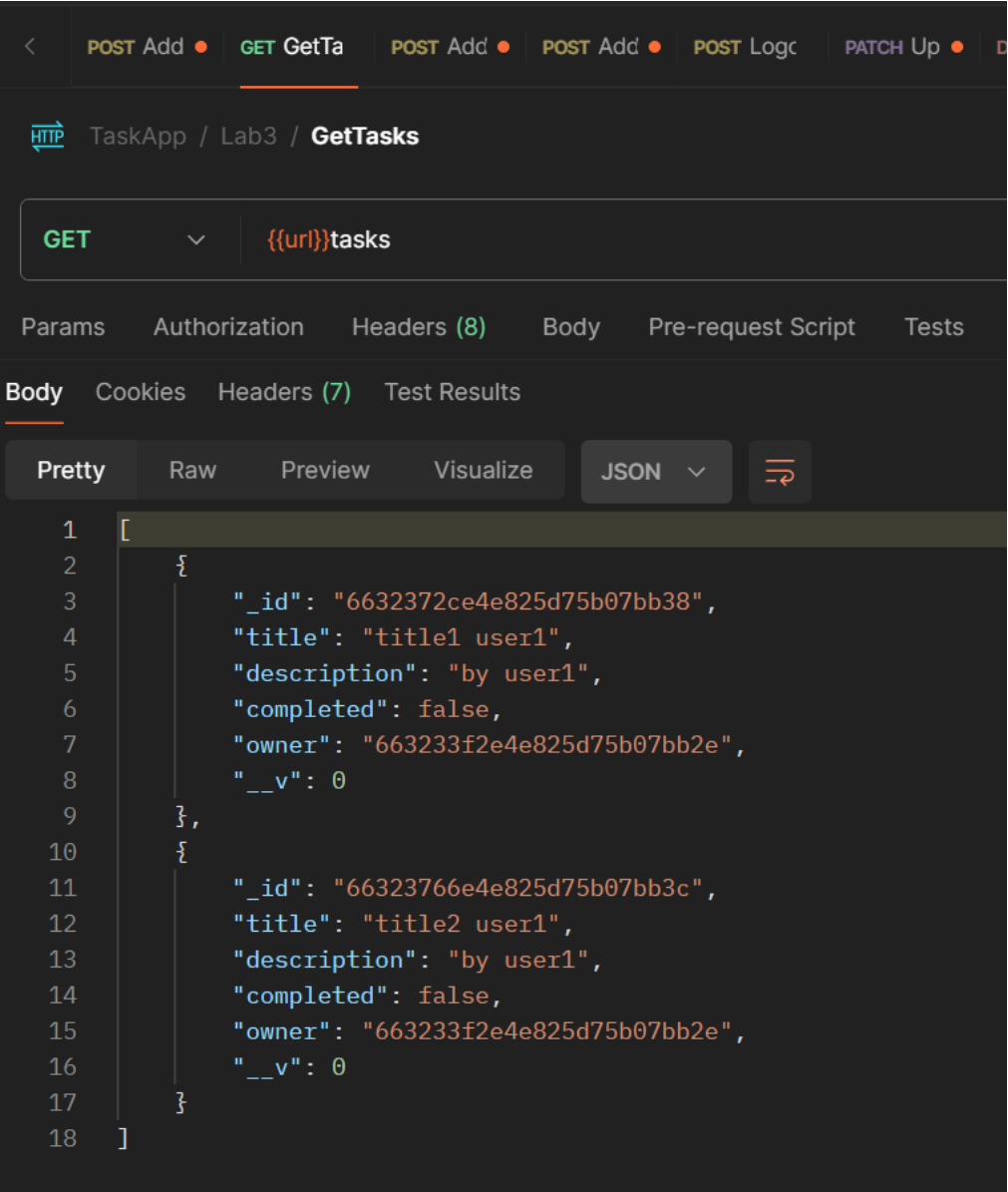


Рис. 24. Вивід задач user1

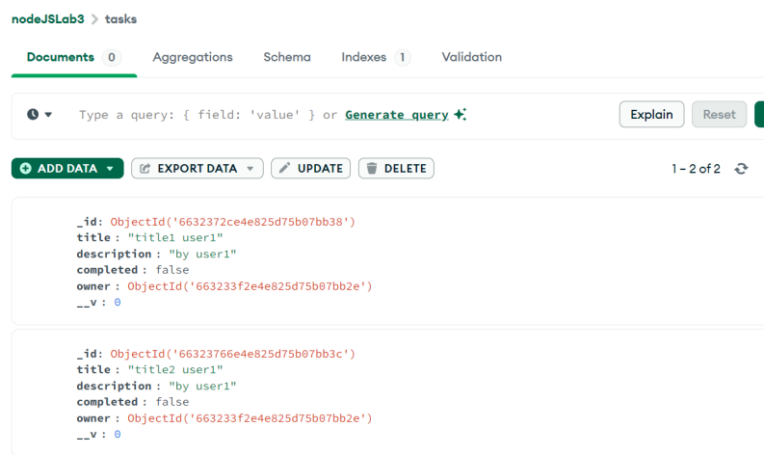


Рис. 25. Результат

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 3. Забороніть відправку захищених даних на клієнт:

```
UserSchema.methods.toJSON = function() {  
  const user = this;  
  const userObject = user.toObject();  
  delete userObject.password;  
  delete userObject.tokens;  
  return userObject;  
}
```

```
const User = mongoose.model("User", UserSchema);
```

Для дозволу передавати віртуальні поля (tasks) в метод toObject, в схемі даних додайте опцію:

```
password: {type: String...},  
tokens: [{...}]  
, {toJSON: {virtuals: true}, toObject: {virtuals: true}}
```

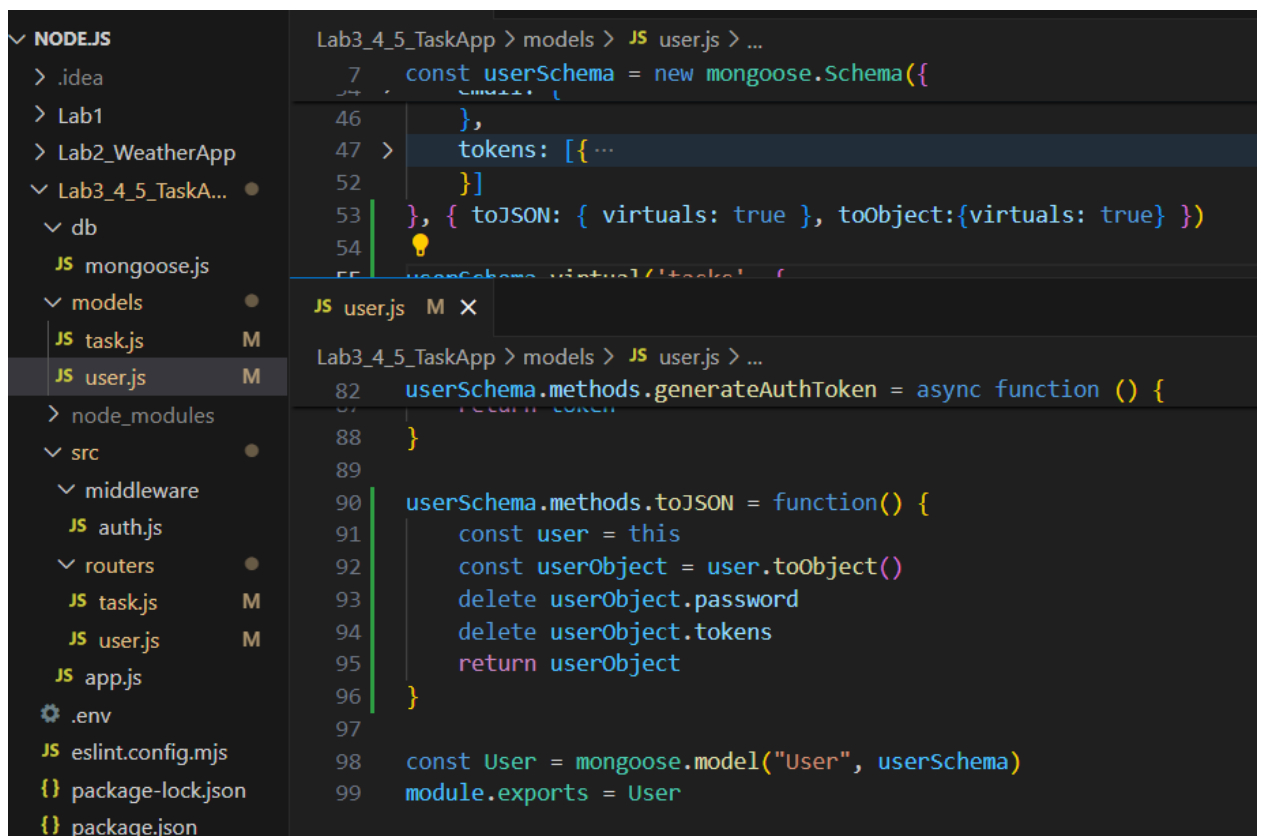


Рис. 26. Результат

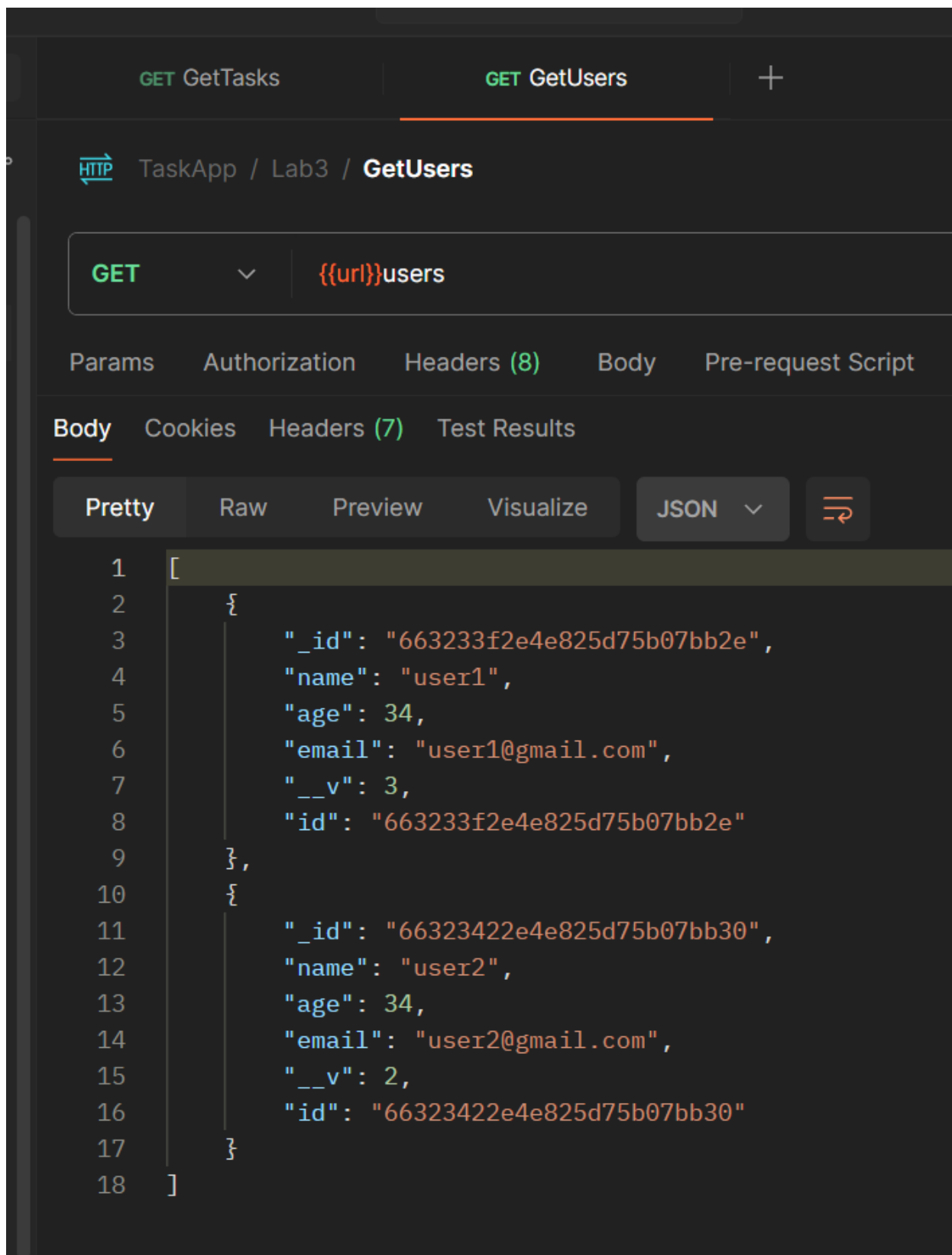


Рис. 27. Результат

Висновок: на лабораторному занятті ми навчились зв'язувати дані.

Завершили виконання TaskApp

		Клосович І.А.			ДУ «Житомирська політехніка».24.121.13.000 - Лр5	Арк.
		Сидорчук В.О.				18
Змн.	Арк.	№ докум.	Підпис	Дата		