NodeJS. TaskApp

Лабораторна робота №5.
Зв'язування користувачів і задач

Завдання

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

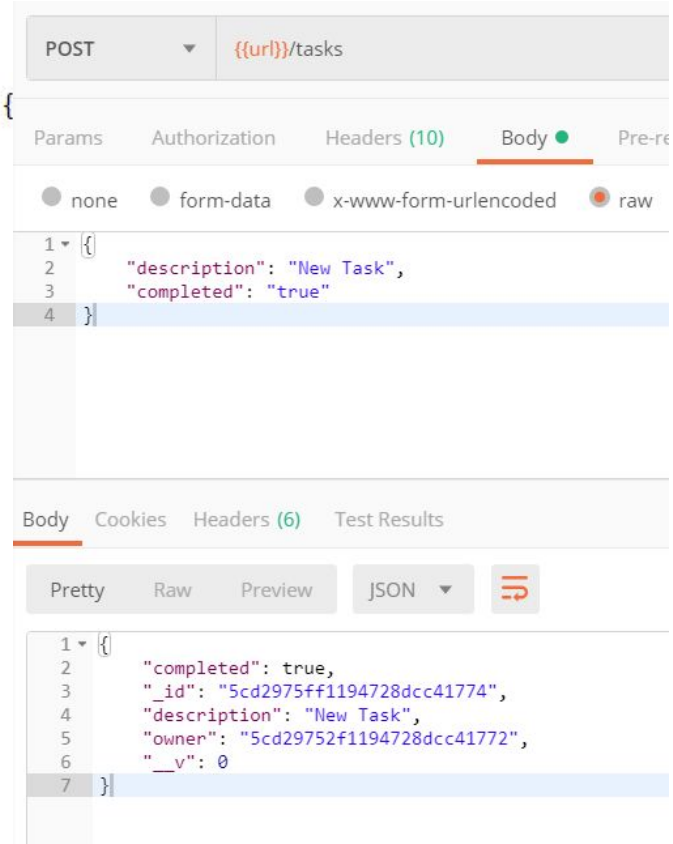
Завдання 1. Зв’язок між користувачами та задачами

- Реалізуйте можливість доступу та обробки лише задач поточного користувача

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

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

```
router.post("/tasks", auth, async (req, res) => {
    //let task = new Task(req.body);
    const task = new Task({
        ...req.body,
        owner: req.user.id
    });
    try {
        await task.save();
        res.status(201).send(task);
    } catch (e) {
        res.status(500).send(e);
    }
});
```



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’);

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’);

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

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.

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}})
```