

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО

Отчет по домашней работе №4 по курсу “Бэкенд разработка”

Выполнили:

Бахарева М. А., К3342

Привалов К.А., К3342

Проверил:

Добряков Д.И.

Санкт-Петербург

2025 г.

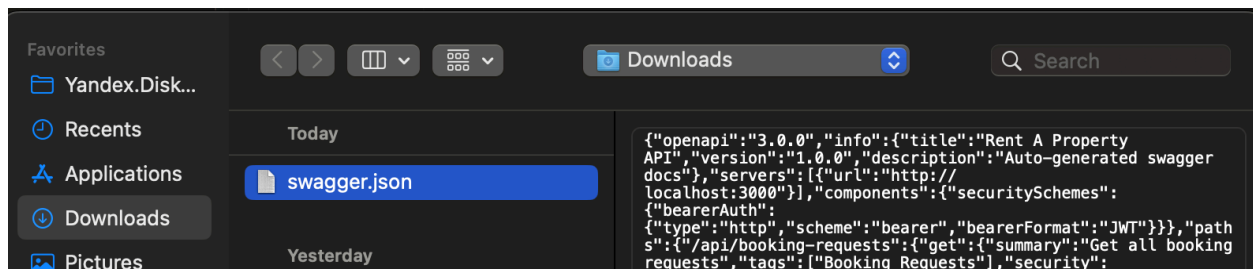
1. Задание:

- реализовать тестирование API средствами Postman;
- написать тесты внутри Postman.

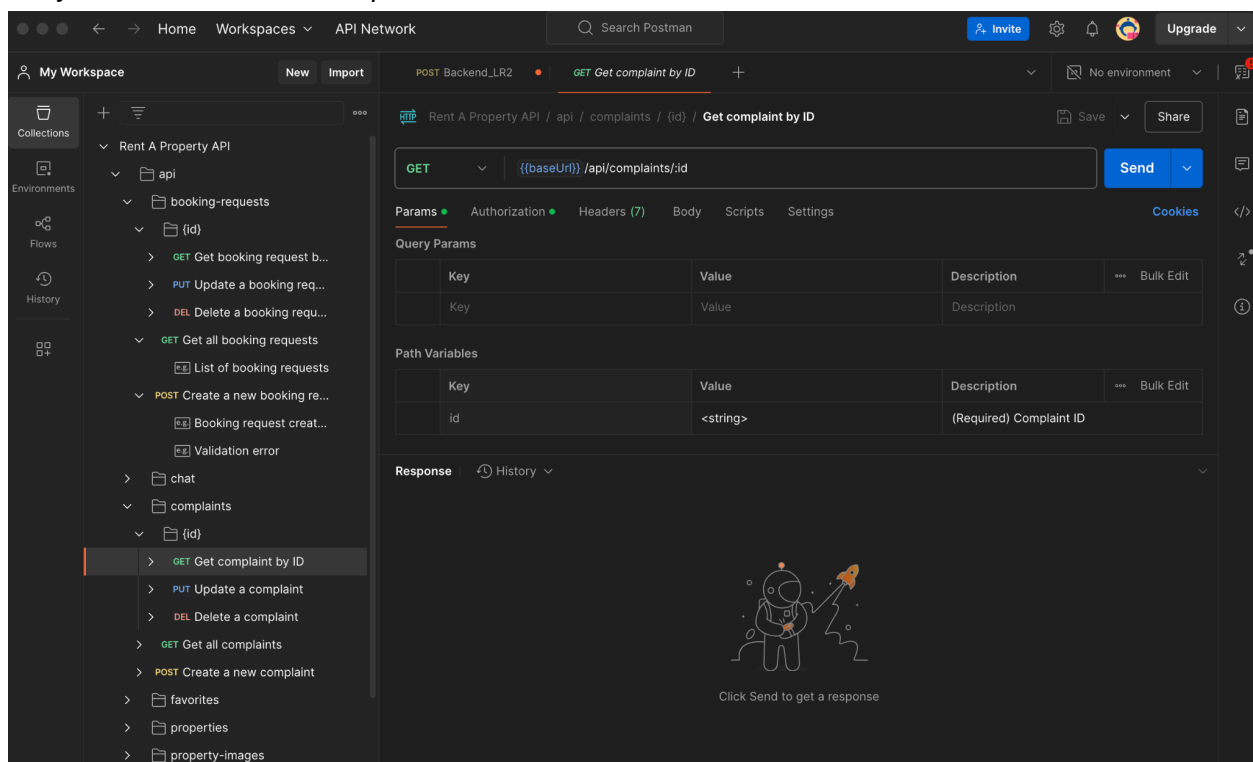
2. Ход работы

В предыдущем домашнем задании мы задокументировали наше приложение и энд-поинты в Postman. Вспомним, как это было и что мы получили: (фрагменты из прошлого ДЗ выделены курсивом)

Создаем новую коллекцию в Postman и импортируем в нее этот файл



Получаем Collection с набором энд-пойнтов



Настроим окружение. Добавим дефолтный URL, а также пропишем токен конкретного юзера (созданного с помощью Swagger выше), чтобы проверить работоспособность

запросов.

The screenshot shows the 'Local' environment configuration in Postman. At the top, there are tabs for 'POST Backend_LR2' and 'GET Get complaint by ID'. Below the tabs, there are buttons for 'Save', 'Fork', and 'Share'. A search bar labeled 'Filter variables' is present. The main table lists variables:

	Variable	Type	Initial value	Current value	...
<input checked="" type="checkbox"/>	baseUrl	default	http://localhost:3000	http://localhost:3000	
<input checked="" type="checkbox"/>	token	default	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	

Проверим работоспособность. Запрос прошел с корректным респонсом в виде запрета видеть всех юзеров системы, так как пользователь не обладает полным набором прав (у него роль - *tenant*)

The screenshot shows the REST client interface in Postman. The URL is `{{baseUrl}}/api/users/` and the method is 'GET'. The response is a '403 Forbidden' status with a message: `{ "message": "Forbidden" }`. The response is displayed in JSON format.

Используем окружения, которые мы создали ранее. Немного модифицируем их. Так как у нас две роли: *landlord/tenant*, создадим два окружения с соответствующими токенами:

The screenshot shows the 'Landlord' environment configuration in Postman. It includes a search bar 'Filter variables' and a table of variables:

	Variable	Type	Initial value	Current value	...
<input checked="" type="checkbox"/>	baseUrl	default	http://localhost:3000	http://localhost:3000	
<input checked="" type="checkbox"/>	token	default	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	

Below the table is an 'Add new variable' button.

HTTP Rent A Property API / api / properties / {id} / Get a property by ID Save Share

GET {{baseUrl}} /api/properties/23 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Pre-request

Post-response

```
1 pm.test("Status code is 200 or 404 or 403", () => {
2   pm.expect(pm.response.code).to.be.oneOf([200, 403, 404]);
3 });
4
5 if (pm.response.code === 200) {
6   const jsonData = pm.response.json();
7   pm.test("Property has required fields", () => {
8     pm.expect(jsonData).to.have.property("id");
9     pm.expect(jsonData).to.have.property("title");
10    pm.expect(jsonData).to.have.property("owner");
11  });
12 }
13
```

Packages </> Snippets Save Response

Body Cookies Headers (18) Test Results (1/1) 404 Not Found 39 ms 912 B Save Response

Filter Results

PASSED Status code is 200 or 404 or 403

Создадим тестово нашему арендодателю объект недвижимости. Ему оно доступно. Тесты также прошли успешно.

HTTP Rent A Property API / api / properties / {id} / Get a property by ID Save Share

GET {{baseUrl}} /api/properties/4 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Pre-request

```
1 pm.test("Status code is 200 or 404 or 403", () => {
2   pm.expect(pm.response.code).to.be.oneOf([200, 403, 404]);
3 });
4
```

Post-response

```
5 if (pm.response.code === 200) {
6   const jsonData = pm.response.json();
7   pm.test("Property has required fields", () => {
8     pm.expect(jsonData).to.have.property("id");
9     pm.expect(jsonData).to.have.property("title");
10    pm.expect(jsonData).to.have.property("owner");
11  });
12 }
```

Body Cookies Headers (18) Test Results (2/2) 200 OK 41 ms 1.26 KB Save Response

JSON Preview Visualize

```
1 {
2   "id": 4,
3   "title": "Cozy Studio Apartment",
4   "description": "A modern and fully furnished studio in the city center.",
5   "price": "1200.50",
6   "location": "123 Main St, New York, NY",
7   "propertyType": "studio",
8   "createdAt": "2025-05-25T19:57:11.168Z",
9   "owner": {
10     "id": 4,
11     "firstName": "maria",
12     "lastName": "mariavna",
13     "birthDate": "2000-05-25",
14     "phone": "89891",
15     "email": "mariyka@mail.ru",
```

Body Cookies Headers (18) Test Results (2/2) 200 OK 41 ms 1.26 KB Save Response

Filter Results

PASSED Status code is 200 or 404 or 403

PASSED Property has required fields

Арендатору также доступна данная недвижимость.

The screenshot shows a REST client interface with the following details:

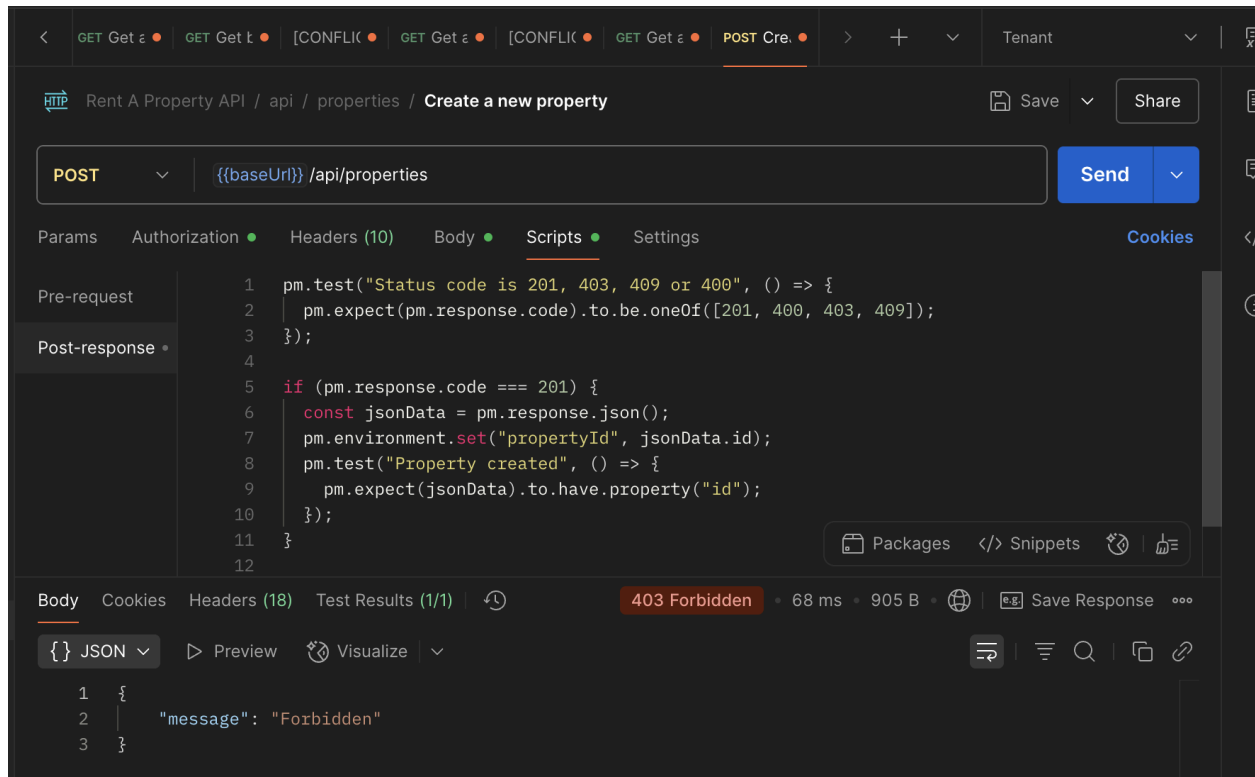
- URL:** `{{baseUrl}}/api/properties/4`
- Method:** GET
- Scripts Tab:**

```
1 pm.test("Status code is 200 or 404 or 403", () => {
2   pm.expect(pm.response.code).to.be.oneOf([200, 403, 404]);
3 });
4
5 if (pm.response.code === 200) {
6   const jsonData = pm.response.json();
7   pm.test("Property has required fields", () => {
8     pm.expect(jsonData).to.have.property("id");
9     pm.expect(jsonData).to.have.property("title");
10    pm.expect(jsonData).to.have.property("owner");
11  });
12 }
```
- Test Results (2/2):**
 - PASSED** Status code is 200 or 404 or 403
 - PASSED** Property has required fields

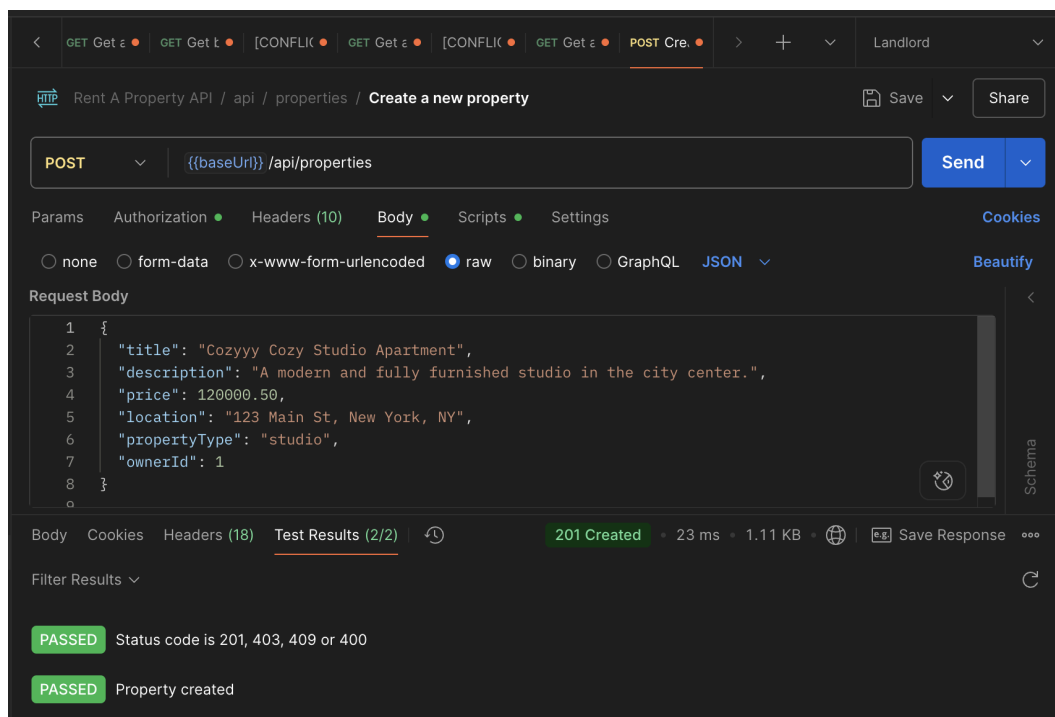
Теперь арендатор попробует создать недвижимость. Передадим в body следующий json

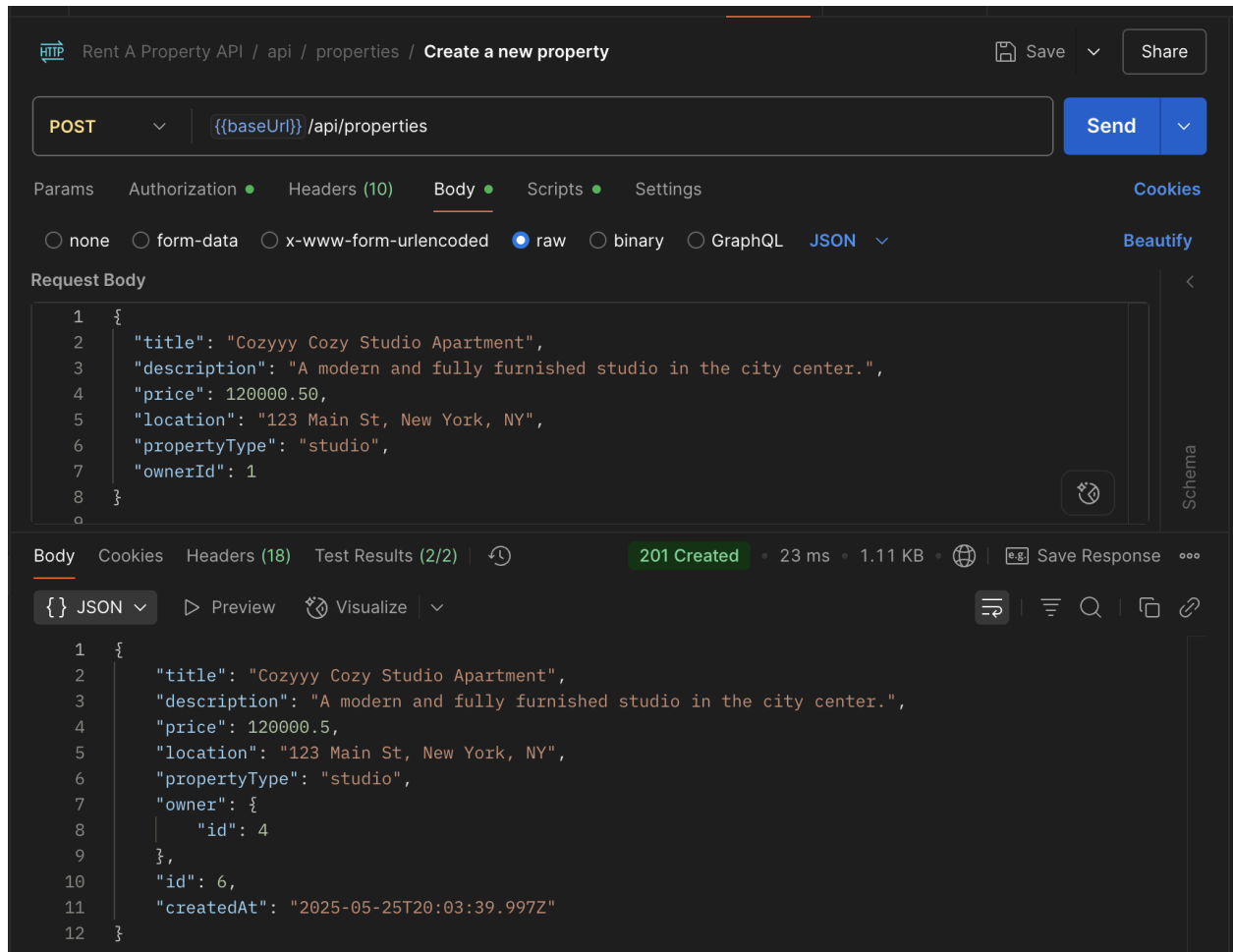
```
1  {
2    "title": "Test Apartment",
3    "description": "Bright and clean",
4    "address": "123 Test St",
5    "price": 1500,
6    "type": "apartment",
7    "ownerId": 1
8  }
```

Получаем Forbidden. Так как у арендатора не может быть прав на создание недвижимости.



Пройдем аналогичный тест, но на окружении владельца недвижимости.





В итоге можно выделить основные принципы построения тест-кейсов в Postman:

1. Разделение по типам запросов

Каждому HTTP-методу соответствует свой набор тестов:

- GET: проверка получения данных, проверка статуса 200, соответствие структуры.
- POST: создание ресурса, проверка 201 Created, валидация тела.
- PUT: обновление ресурса, проверка 200, контроль доступа.
- DELETE: удаление, проверка 204, повторная попытка — 404.

2. Проверка ролей и доступа

Проверяется поведение API при запросе с разными токенами (landlord, tenant, неавторизованный пользователь). В тестах симулируются попытки доступа к чужим данным → ожидается 403 Forbidden.

3. Вывод

В ходе выполнения лабораторной работы были в качестве примера реализованы и протестированы API-эндпоинты для управления объектами недвижимости (Properties) . Были учтены особенности ролевой модели (LANDLORD, TENANT), а также реализованы проверки прав доступа, коды ошибок и поведение API при разных сценариях.