САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №4

Выполнил:

Кадникова Екатерина

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

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

Ход работы

1. Документация API в Postman

Экспортирован Swagger JSON (api-docs.json) из документации. Импортирована документация в Postman (см. Рисунок 1). Настроены коллекции запросов для всех endpoints. Добавлены примеры запросов и ответов. Настроена авторизация (JWT токен).

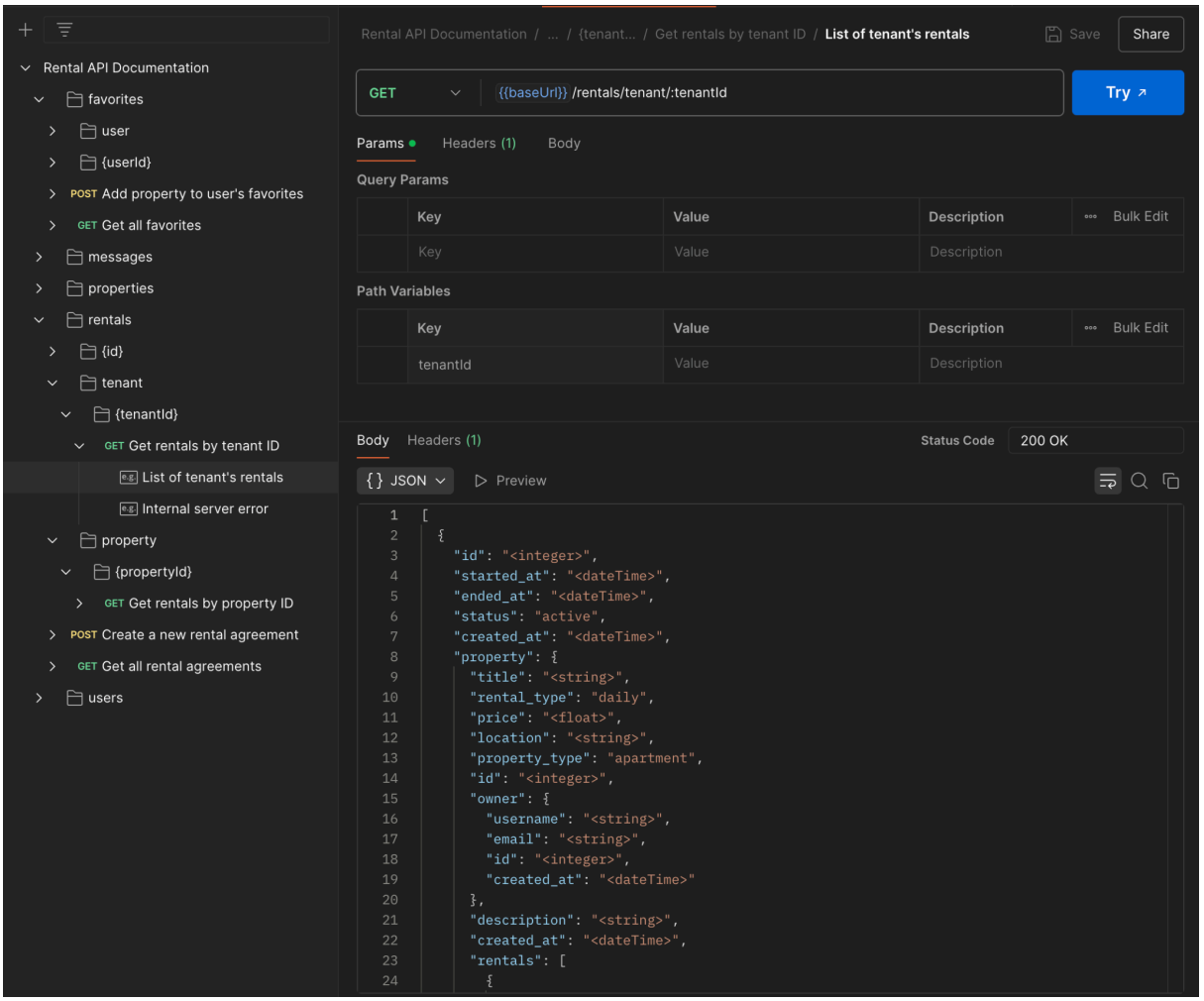

Рисунок 1 - Настроенная коллекция в Postman

2. Общие тесты в Postman

Сначала были созданы тесты на уровне коллекции (см. Листинг 1), которые будут применяться ко всем запросам. Эти тесты выполняют следующие проверки:

- Проверка статуса (2XX или ожидаемые ошибки);
- Проверка формата ответа (JSON);
- Проверка структуры ответа (как минимум одно поле, содержит ‘message’ или ‘error’в случае ошибки и т.д.);
- Проверка времени ответа (<= 2 сек.);
- Автоматическая установка токена авторизации.

Листинг 1 - Тесты на уровне коллекции

```
pm.test("Сообщение отправлено - статус 201", function() {
    pm.response.to.have.status(201);
});

const message = pm.response.json();

pm.test("Ответ содержит полную информацию о сообщении",
function() {
    pm.expect(message).to.have.property("id");
    pm.expect(message).to.have.property("content");
    pm.expect(message).to.have.property("sent_at");
    pm.expect(message).to.have.property("sender");
    pm.expect(message).to.have.property("receiver");
    pm.expect(message).to.have.property("property");
});

pm.test("Данные сообщения соответствуют отправленным",
function() {

pm.expect(message.sender.id).to.eql(pm.environment.get("te
st_user_id"));

pm.expect(message.receiver.id).to.eql(pm.environment.get("
test_receiver_id"));

pm.expect(message.property.id).to.eql(pm.environment.get("
test_property_id"));
});

pm.environment.set("test_message_id", message.id);
```

3. Тесты для запросов

Для каждого запроса были также реализованы свои локальные тесты. Рассмотрим на примере тестов для запросов по работе с пользователем.

Первым тестом в цепочке является запрос на регистрацию пользователя POST /users. Для него был добавлен JS-скрипт, выполняющийся перед отправкой запроса (см. Листинг 2), который выполняет:

- Генерацию случайного имени пользователя;
- Генерацию случайно email'а;
- Фиксированный пароль (можно также генерировать его случайным образом, но для уникальности регистрируемого пользователя достаточно логина и почты).

Эти данные сохраняются как переменные и используются в отправляемом запросе (см. Рисунок 2).

Листинг 2 - Pre-request Script для запроса на регистрацию

```
const randomUsername = `user_${Math.floor(Math.random() * 10000)}_${Date.now()}`;
pm.environment.set("random_username", randomUsername);

const randomEmail = `test_${Math.floor(Math.random() * 10000)}_${Date.now()}@example.com`;
pm.environment.set("random_email", randomEmail);

pm.environment.set("user_password", "TestPassword123!");
```

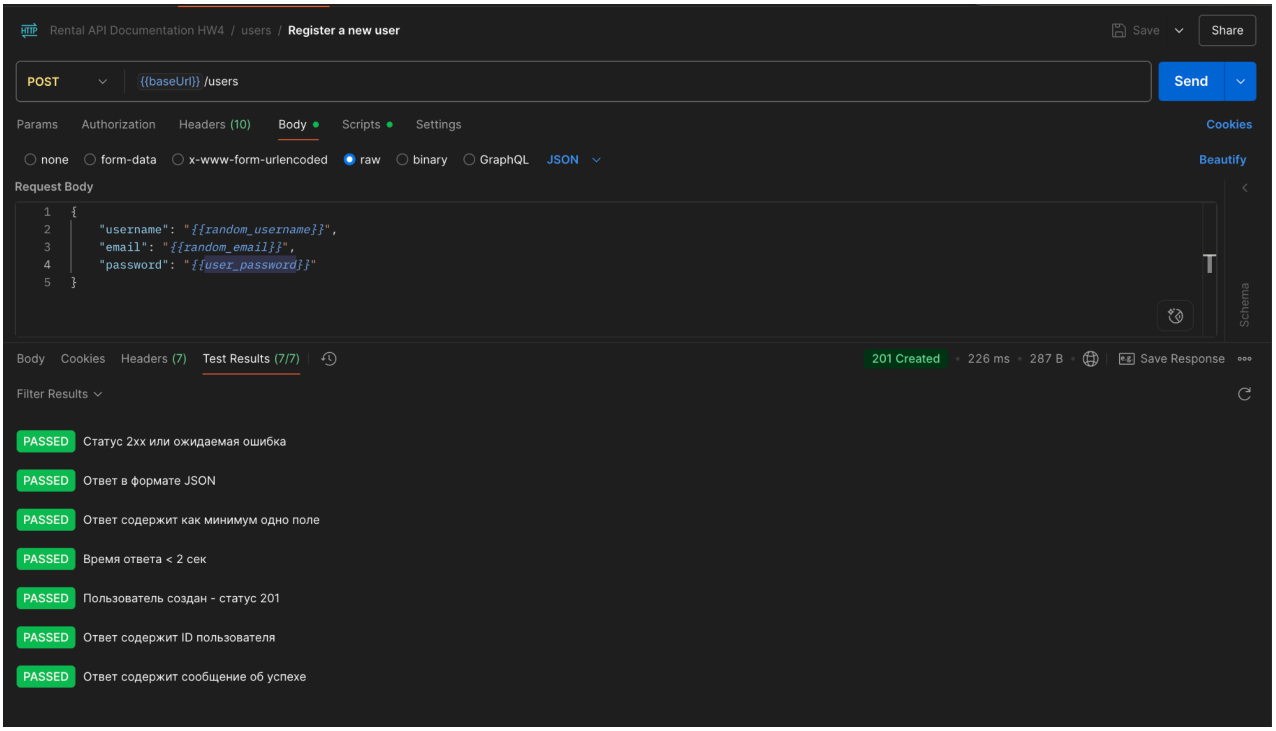

Рисунок 2 - Запрос на регистрацию в Postman

После генерации тестовых данных и отправки запроса выполняются тесты (см. Листинг 3). Тесты выполняют следующие проверки:

1. Проверка конкретного статуса 201 - успешное создание пользователя;
2. В ответе содержится id созданного пользователя и сообщение об успехе (в соответствие с логикой в контроллере).

Листинг 3 - Тесты для запроса на регистрацию

```
pm.test("Пользователь создан - статус 201", function () {
    pm.response.to.have.status(201);
});

const res = pm.response.json();
pm.test("Ответ содержит ID пользователя", function () {
    pm.expect(res).to.have.property("id");
    pm.expect(res.id).to.be.a("number");
    pm.environment.set("test_user_id", res.id);
});

pm.test("Ответ содержит сообщение об успехе", function () {
    pm.expect(res).to.have.property("message");
    pm.expect(res.message).to.include("successfully");
});
```

Зарегестрированный в рамках данного конкретного запроса пользователь может далее использовать в других запросах (в окружении сохранены id, логин, почта и пароль пользователя). Например, сразу после можно выполнить запрос на авторизацию с использованием тех же переменных (см. Рисунок 3).

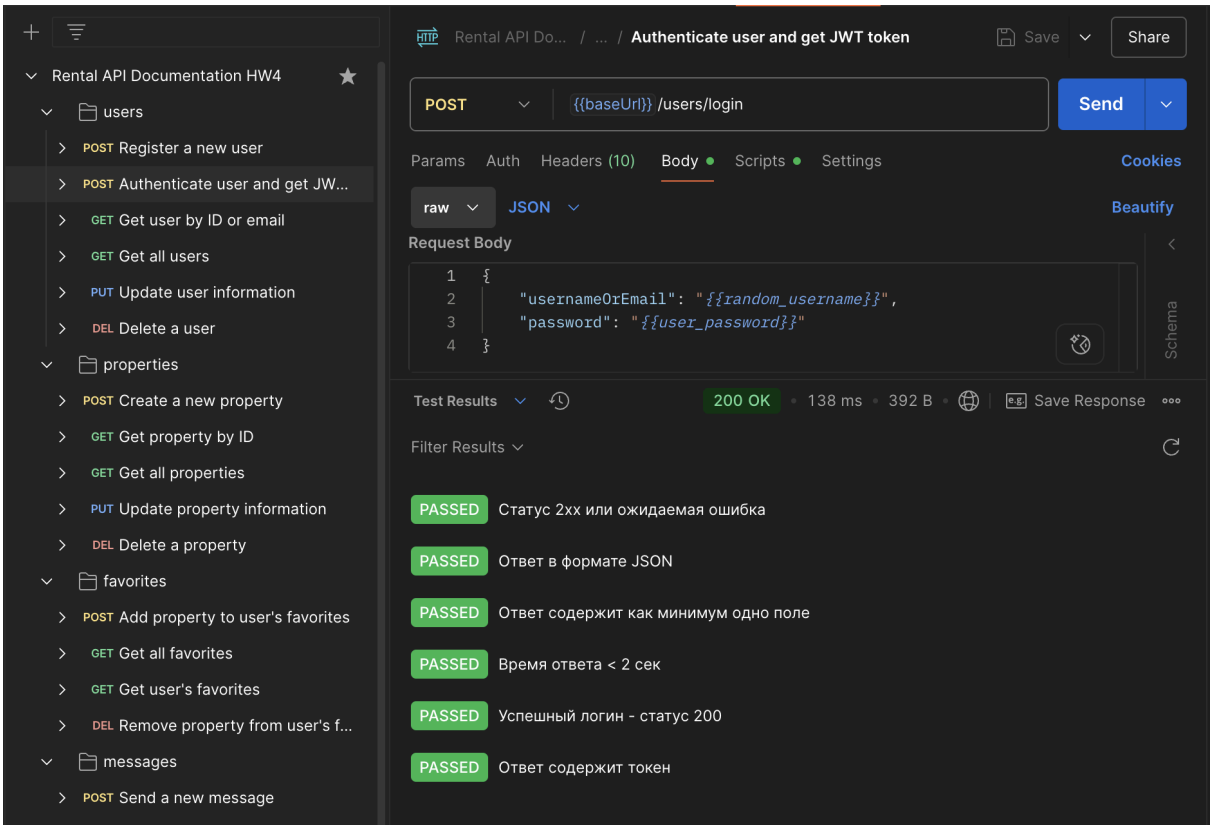

Рисунок 3 - Последующий запрос на авторизацию

Таким же образом написаны тесты для остальных запросов, созданы цепочки запросов. Например, можно запустить тесты сразу на всю коллекцию или одну конкретную цепочку (см. Рисунок 4).

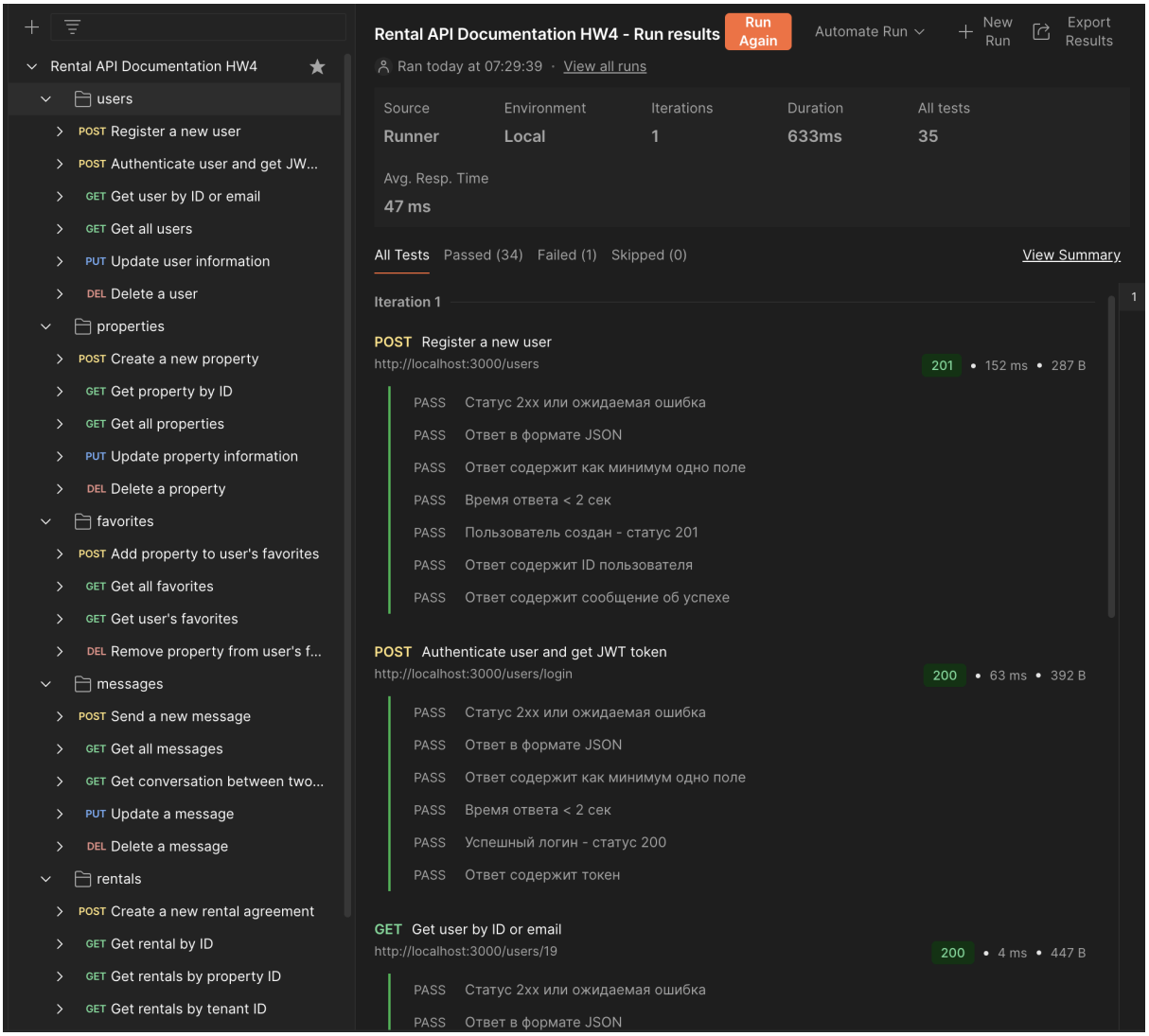

Рисунок 4 - Результат выполнения тестов для запросов по пути /users

Вывод

В рамках работы реализовано автоматическое тестирование средствами Postman, созданы общие и локальные тесты для каждого запроса.