

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 4

Выполнила:

Фролова Кристина

Группа К3339

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

- Реализовать тестирование API средствами Postman;
- Написать тесты внутри Postman.

Ход работы

Вариант: Сервис для аренды недвижимости

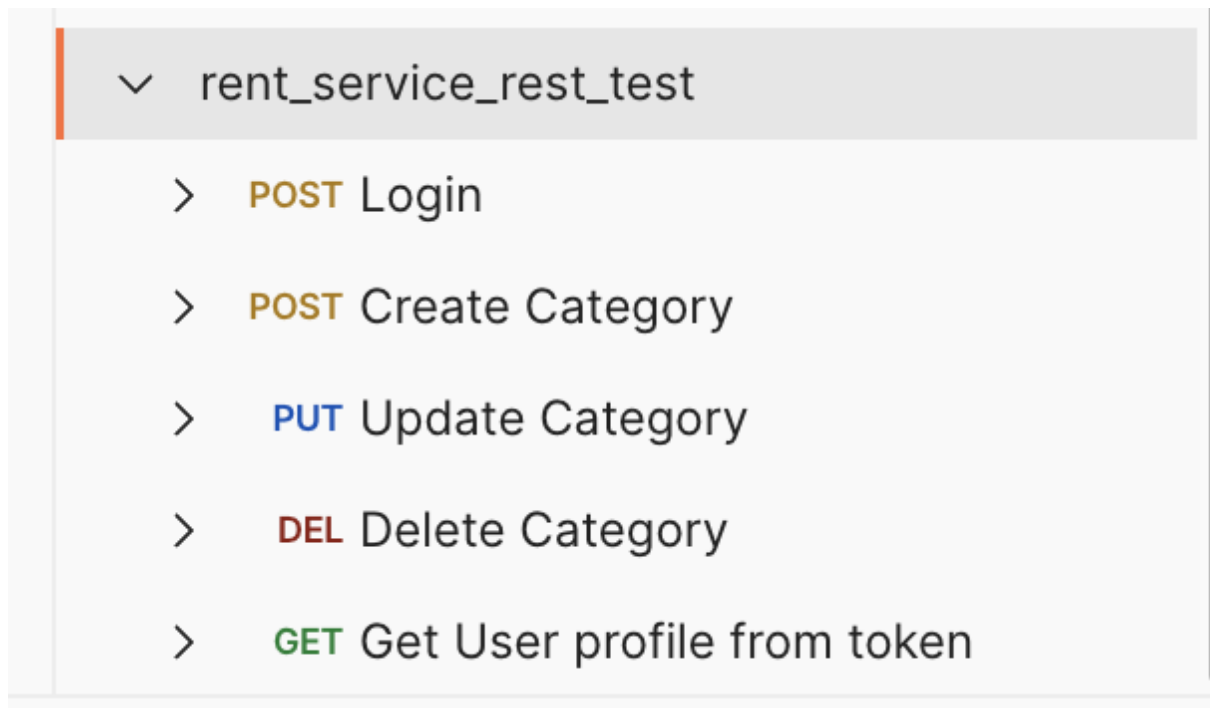


Рисунок 1 – Коллекция запросов для тестирования

rent_service_rest / auth / login / **Login** Save Share

POST ⌵ `{{baseUrl}}/auth/login` Send ⌵

Params Authorization Headers (11) Body **Scripts** Settings Cookies

Pre-request

Post-response

```
1 pm.test("Get access token from response", function () {
2   const jsonData = pm.response.json();
3   pm.response.to.have.jsonBody("accessToken");
4   pm.collectionVariables.set("access", jsonData.accessToken);
5 });
```

⌵ Packages </> Snippets 🔄 📋

Body Cookies Headers (7) **Test Results (1/1)** 🕒 **200 OK** • 41 ms • 494 B • 🌐 📄 Save Response ⋮

ℹ️ Testing your API endpoints one by one? Run your entire collection and test end-to-end workflows effortlessly. [Learn more](#) ➤ ▶ Run Collection ✕

Filter Results ⌵ 🔄

PASSED Get access token from response

Рисунок 2 – Тест для проверки получения acesstokena и его вставки в среду

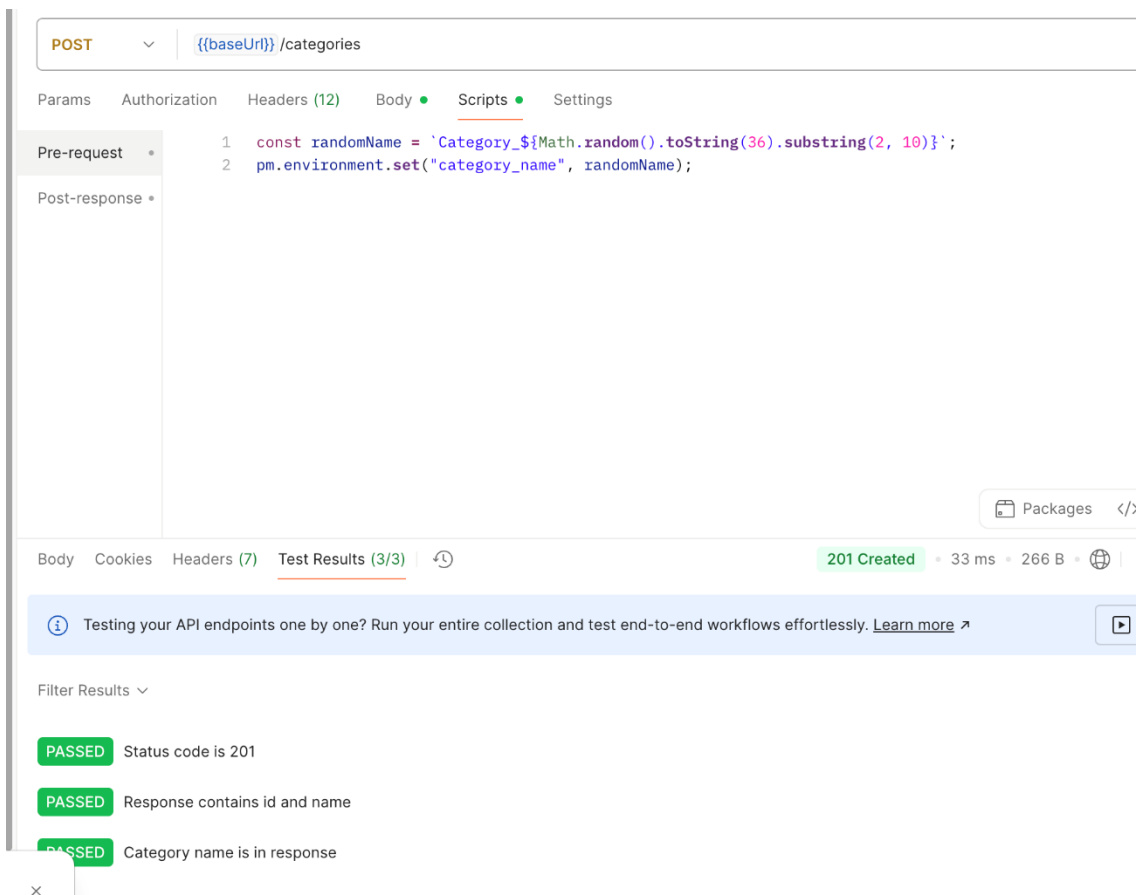


Рисунок 3 – Предварительное создание случайного имени для категории

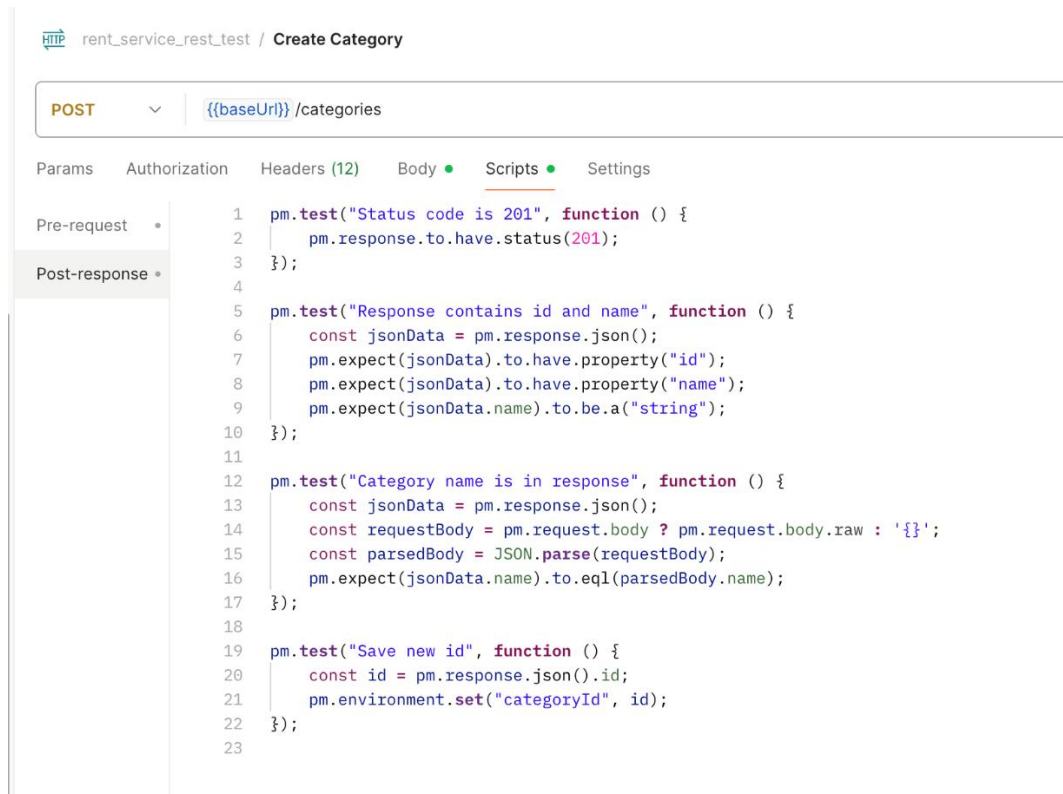


Рисунок 4 – Тесты для проверки запроса создания категории

rent_service_rest_test / Update Category

PUT {{baseUrl}}/categories/{{categoryId}}

Params Authorization Headers (12) Body Scripts Settings

Pre-request

```
1 if (!pm.environment.get("categoryId")) {
2   |   throw new Error("categoryId is not set! Firstly call POST /categories.");
3   |
4   |
5   const newName = `Updated_${Math.random().toString(36).substring(2, 10)}`;
6   pm.environment.set("updatedCategoryName", newName);
7 }
```

Post-response

Body Cookies Headers (7) Test Results (3/3) 200 OK • 11 ms

{ } JSON Preview Visualize

```
1 {
2   |   "id": 13,
3   |   "name": "Updated_4nkn60xe"
4   | }
```

Рисунок 5 – Предварительные действия для теста на обновление категории

PUT

▼

{{baseUrl}}/categories/{{categoryId}}

Params

Authorization ●

Headers (12)

Body ●

Scripts ●

Settings

Pre-request •

Post-response •

```
1 pm.test("Response status is 200", function () {
2   pm.expect(pm.response.code).toEqual(200);
3 });
4
5 pm.test("Response contains id and new name", function () {
6   const jsonData = pm.response.json();
7   pm.expect(jsonData).to.have.property("id");
8   pm.expect(jsonData).to.have.property("name");
9   pm.expect(jsonData.name).toEqual(pm.variables.get("updatedCategoryName"));
10 });
11
12 pm.test("Category name is validated", function () {
13   const jsonData = pm.response.json();
14   const nameLength = jsonData.name.length;
15   pm.expect(nameLength).toBeWithin(2, 50);
16 });
17
```

Body

Cookies

Headers (7)

Test Results (3/3)

🕒

200 OK • 11 ms



{ } JSON ▼

▶ Preview

🔗 Visualize ▼

```
1 {
2   "id": 13,
3   "name": "Updated_4nkn60xe"
4 }
```

Рисунок 6 - Тесты для проверки запроса обновления категории

 rent_service_rest_test / Delete Category 

DELETE

▼

{{baseUrl}}/categories/{{categoryId}}

Params

Authorization ●

Headers (8)

Body

Scripts ●

Settings

Pre-request •

Post-response •

```
1 pm.test("Response status is 204", function () {
2   pm.expect(pm.response.code).toEqual(204);
3 });
4
5 pm.test("Response is empty or contains confirmation message", function () {
6   const resText = pm.response.text();
7   if (pm.response.code === 204) {
8     pm.expect(resText).toBe.empty;
9   } else {
10    const jsonData = pm.response.json();
11    pm.expect(jsonData.message || jsonData.status || jsonData.success).to.not.be.undefined;
12  }
13 });
14 pm.test("categoryId no longer needed – remove it from environment", function () {
15   pm.environment.unset("categoryId");
16 });
```

Рисунок 7 - Тесты для проверки запроса удаления категории

The screenshot shows the Postman interface for a REST client request. The request is a GET to `{{baseUrl}}/users/me`. The 'Scripts' tab is active, showing two PM scripts:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Response contains user data", function () {
6   const json = pm.response.json();
7   pm.expect(json).to.have.property("id").that.is.a("number");
8   pm.expect(json).to.have.property("mail").that.is.a("string");
9   pm.expect(json).to.have.property("firstName").that.is.a("string");
10  pm.expect(json).to.have.property("lastName").that.is.a("string");
11  pm.expect(json).to.have.property("createdAt").that.is.a("string");
12  pm.expect(json).to.have.property("updatedAt").that.is.a("string");
13 });
14
```

The 'Test Results' tab shows two passed tests:

- PASSED** Status code is 200
- PASSED** Response contains user data

Рисунок 7 - Тесты для проверки запроса получения пользователя за счёт токена

Вывод

В рамках работы были созданы тесты внутри Postman для реализации тестирования API.