

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 5

Выполнил:

Никитин Павел

Группа

K33402

Проверил:

Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Тестирование API из ЛР2 средствами Postman (написать 2-3 теста для конкретных эндпоинтов + 1 общий тест для всех эндпоинтов)

Ход работы

Для начала мы сохранили адрес api на который мы будем делать запросы в дальнейшем.

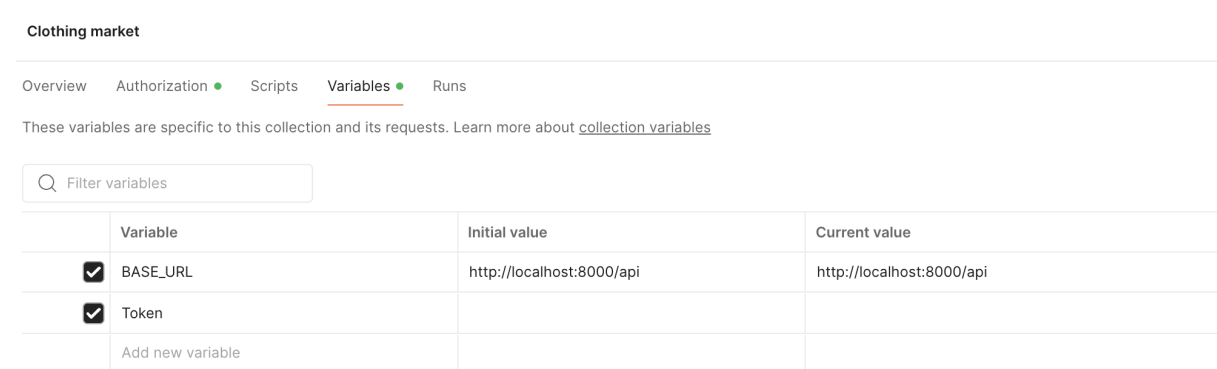


Рисунок 1 - переменные

В дальнейшем в поле JWT Token будет лежать токен для дальнейших запросов.

Для тестов мы выбрали несколько endpoints, которые отображают все CRUD операции

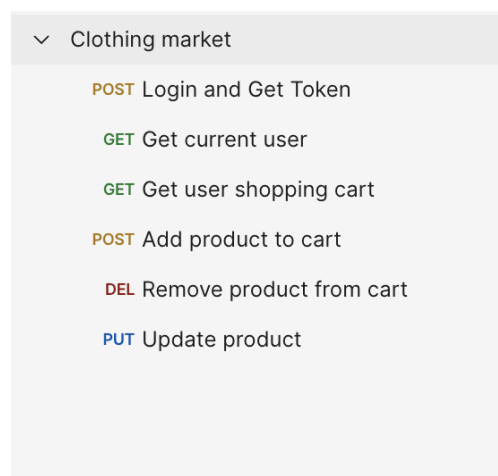


Рисунок 2 - тесты

Для каждого из тестов были написаны скрипты

```
1 pm.test("Token received", function() {
2     var jsonData = pm.response.json();
3     pm.collectionVariables.set("Token", jsonData.tokens.accessToken);
4 });
5
6
7 pm.test("Response status code is 201", function () {
8     pm.expect(pm.response.code).to.equal(201);
9 });
10
11
12 pm.test("Access token and refresh token are non-empty strings", function () {
13     const responseData = pm.response.json();
14
15     pm.expect(responseData.tokens.accessToken).to.be.a('string').and.to.have.lengthOf.at.least(1);
16     pm.expect(responseData.tokens.refreshToken).to.be.a('string').and.to.have.lengthOf.at.least(1);
17 });
18
19
20 pm.test("Response has valid JSON structure", function() {
21     var jsonData = pm.response.json();
22
23     pm.expect(jsonData).to.have.property("user");
24     pm.expect(jsonData.user).to.have.property("id", 3);
25     pm.expect(jsonData.user).to.have.property("fullName", "string");
26     pm.expect(jsonData.user).to.have.property("email", "string@mail.com");
27     pm.expect(jsonData.user).to.have.property("cartId", 3);
28     (local var) jsonData: any
29
30     pm.expect(jsonData).to.have.property("tokens");
31     pm.expect(jsonData.tokens).to.have.property("accessToken");
32     pm.expect(jsonData.tokens.accessToken).to.match(/^[A-Za-z0-9-]+\.[A-Za-z0-9-]+\.[A-Za-z0-9-]+$/); // Проверка JWT
33     токена
34
35     pm.expect(jsonData.tokens).to.have.property("refreshToken");
36     pm.expect(jsonData.tokens.refreshToken).to.match(/^[A-Za-z0-9-]+\.[A-Za-z0-9-]+\.[A-Za-z0-9-]+$/); // Проверка JWT
37     токена
38 });
```

Рисунок 3 - получение токена

Токен сохраняется в переменную и используется при дальнейших запросах.

```
1 pm.test("Response status code is 200", function () {
2     pm.response.to.have.status(200);
3 });
4
5 pm.test("Content-Type header is application/json", function () {
6     pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
7 });
8
9 pm.test("Response has valid JSON structure", function() {
10     var jsonData = pm.response.json();
11
12     pm.expect(jsonData).to.have.property("user");
13
14     pm.expect(jsonData.user).to.have.property("id", 3);
15     pm.expect(jsonData.user).to.have.property("fullName", "string");
16     pm.expect(jsonData.user).to.have.property("email", "string@mail.com");
17     pm.expect(jsonData.user).to.have.property("cartId", 3);
18 });
```

Рисунок 4 - получение пользователя

Также мы можем подставлять mock данные и прогонять тесты с такими значениями:

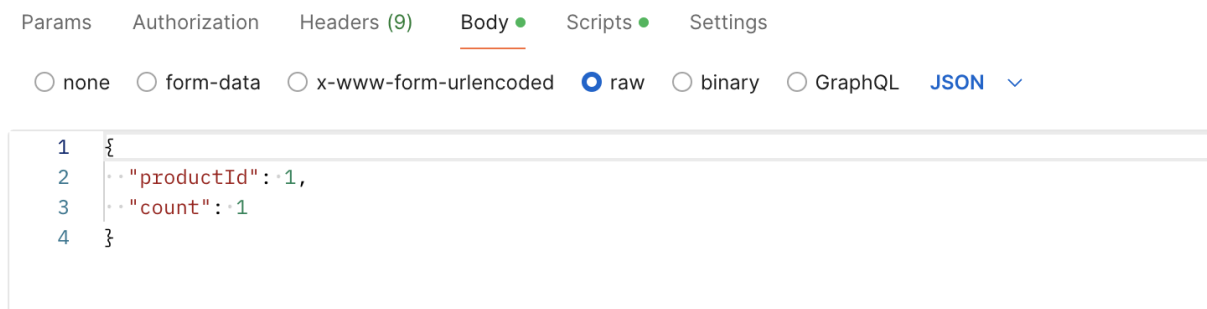


Рисунок 5 - body запроса на добавление товара в корзину



Рисунок 6 - скрипт запроса

Также я написал один общий тест

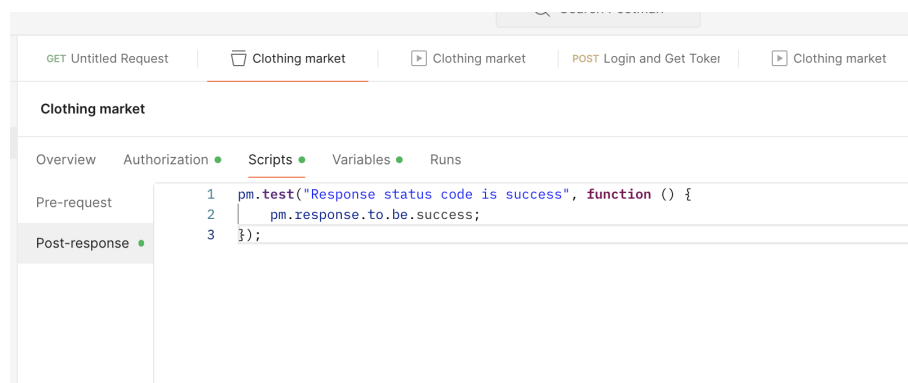


Рисунок 7 - общий тест

Clothing market - Run results

Ran today at 10:47:01 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	1s 687ms	11	278 ms

All Tests Passed (11) Failed (0) Skipped (0)

Iteration 1

POST Login and Get Token

http://localhost:8000/api/auth/login

PASS Response status code is success
PASS Token received
PASS Access token and refresh token are non-empty strings
PASS Response has valid JSON structure and types

GET Get current user

http://localhost:8000/api/user/me

PASS Response status code is success
PASS Content-Type header is application/json
PASS Response has valid JSON structure and types

GET Get user shopping cart

http://localhost:8000/api/cart/my-cart

PASS Response status code is success
PASS Response has valid JSON structure and types

POST Add product to cart

http://localhost:8000/api/cart/add-product

PASS Response status code is success
PASS Response has valid JSON structure and types

Рисунок 8 - результат

Вывод:

В ходе работы мы научились писать тесты при помощи платформы postman запускать их а также работать с переменными, это интересный и репрезентативный инструмент для тестирования.