САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 5

Выполнил:

Никитин Павел

Группа

К33402

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.

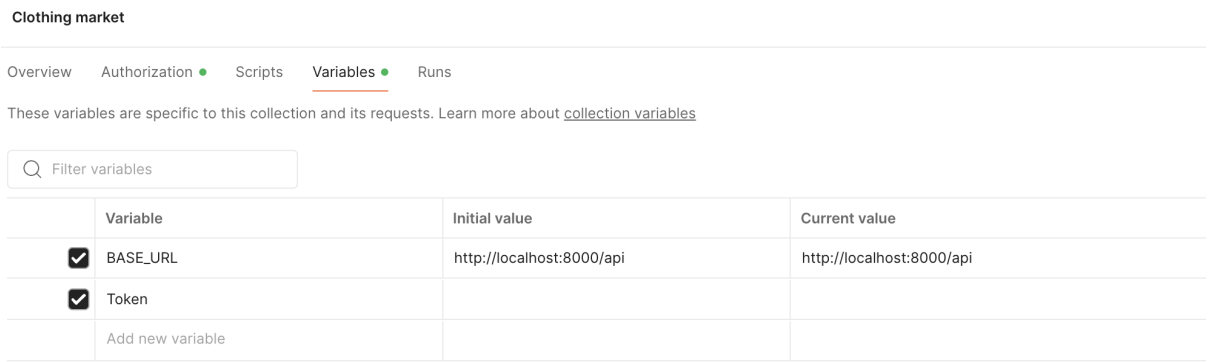

Рисунок 1 - переменные

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

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

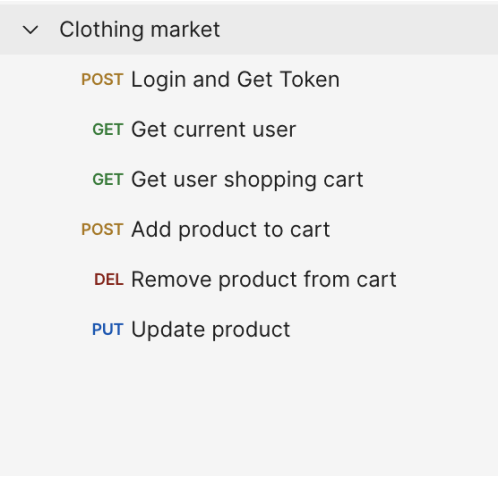

Рисунок 2 - тесты

Для каждого из тестов были написаны скрипты

```
pm.test("Token received", function() {
    var jsonData = pm.response.json();
    pm.collectionVariables.set("Token", jsonData.tokens.accessToken);
});


pm.test("Response status code is 201", function () {
  pm.expect(pm.response.code).to.equal(201);
});


pm.test("Access token and refresh token are non-empty strings", function () {
    const responseData = pm.response.json();

    pm.expect(responseData.tokens.accessToken).to.be.a('string').and.to.have.lengthOf.at.least(1);
    pm.expect(responseData.tokens.refreshToken).to.be.a('string').and.to.have.lengthOf.at.least(1);
});


pm.test("Response has valid JSON structure", function() {
    var jsonData = pm.response.json();

    pm.expect(jsonData).to.have.property("user");
    pm.expect(jsonData.user).to.have.property("id", 3);
    pm.expect(jsonData.user).to.have.property("fullName", "string");
    pm.expect(jsonData.user).to.have.property("email", "string@mail.com");
    pm.expect(
              (local var) jsonData: any
    pm.expect(jsonData).to.have.property("tokens");
    pm.expect(jsonData.tokens).to.have.property("accessToken");
    pm.expect(jsonData.tokens.accessToken).to.match(/^[A-Za-z0-9-_]+\.[A-Za-z0-9-_]+\.[A-Za-z0-9-_]+$/); // Проверка JWT
        токена

    pm.expect(jsonData.tokens).to.have.property("refreshToken");
    pm.expect(jsonData.tokens.refreshToken).to.match(/^[A-Za-z0-9-_]+\.[A-Za-z0-9-_]+\.[A-Za-z0-9-_]+$/); // Проверка JWT
        токена
});
```

Рисунок 3 - получение токена

Токен сохраняется в переменную и используется при дальнейших запросах.

```
pm.test("Response status code is 200", function () {
  pm.response.to.have.status(200);
});

pm.test("Content-Type header is application/json", function () {
    pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
});

pm.test("Response has valid JSON structure", function() {
    var jsonData = pm.response.json();

    pm.expect(jsonData).to.have.property("user");

    pm.expect(jsonData.user).to.have.property("id", 3);
    pm.expect(jsonData.user).to.have.property("fullName", "string");
    pm.expect(jsonData.user).to.have.property("email", "string@mail.com");
    pm.expect(jsonData.user).to.have.property("cartId", 3);
});
```

Рисунок 4 - получение пользователя

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

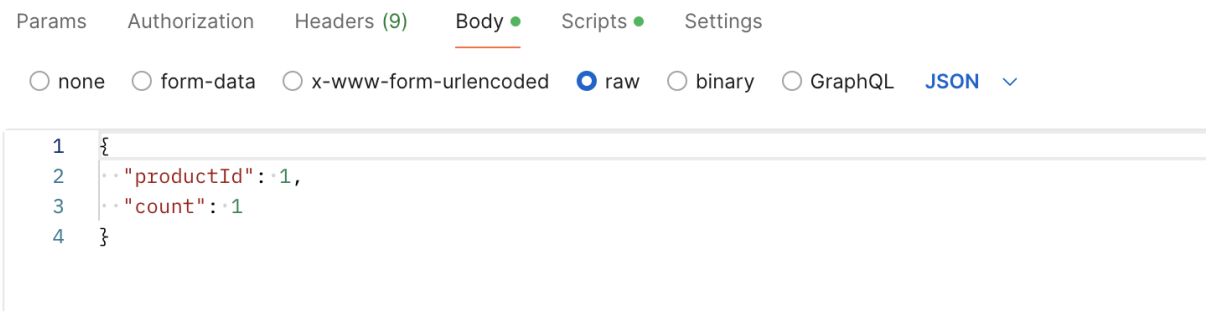

Рисунок 5 - body запроса на добавление товара в корзину

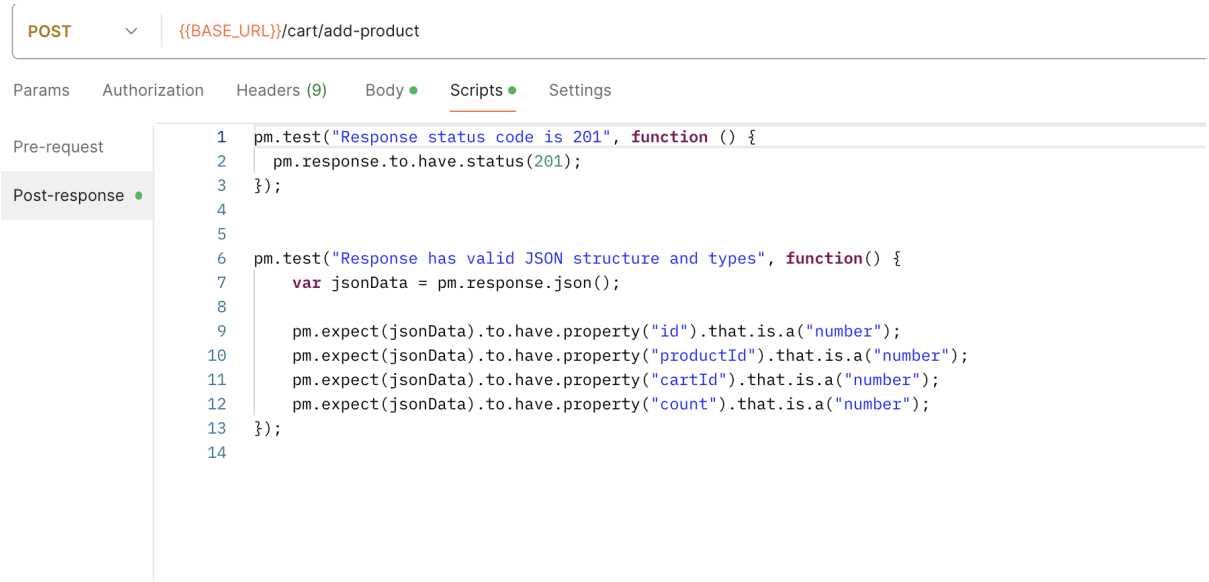

Рисунок 6 - скрипт запроса

Также я написал один общий тест

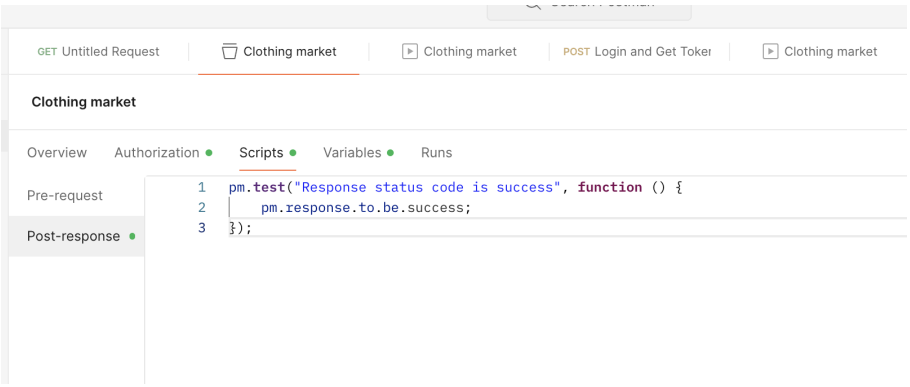

Рисунок 7 - общий тест

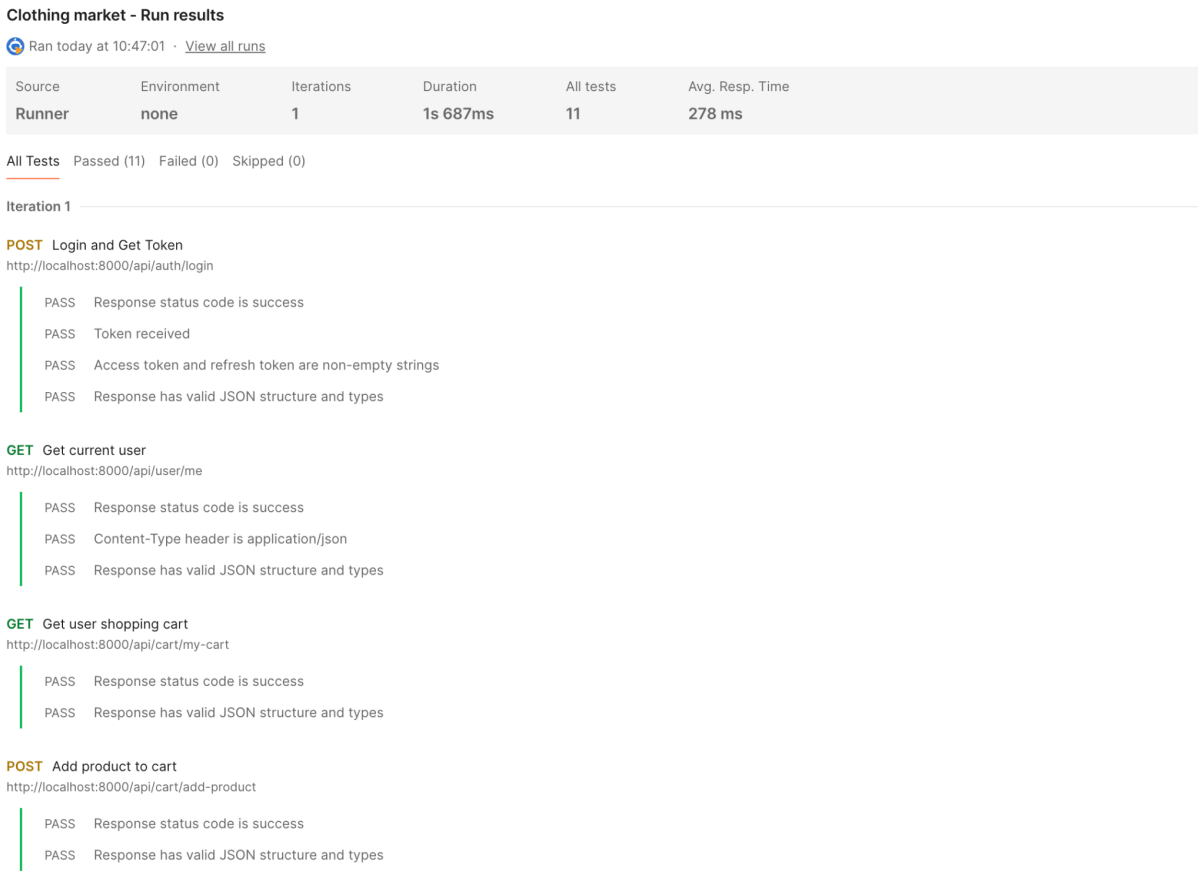

Рисунок 8 - результат

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.