

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 4-5

Выполнил: Рашевский Вячеслав
Васильевич

Группа К33402

Проверил:
Добряков Д. И.

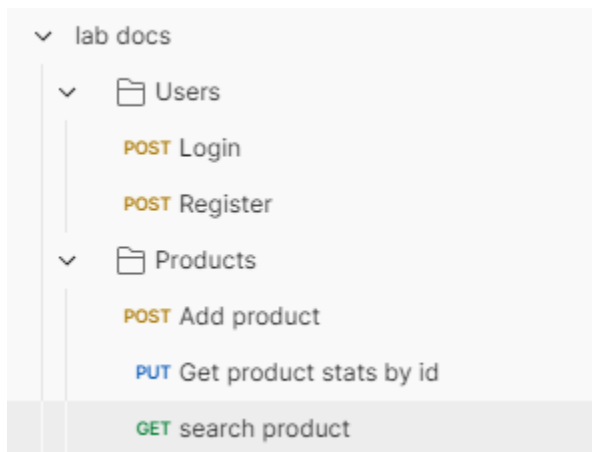
Санкт-Петербург

Задача:

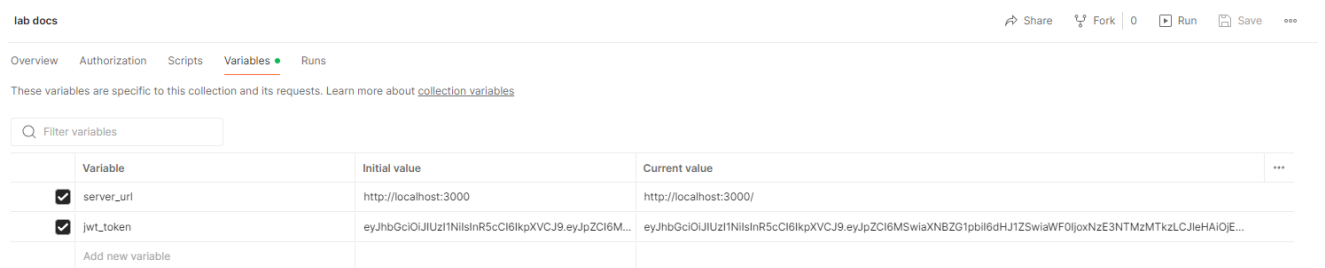
Документирование API из ЛР2 средствами Swagger, Postman

Тестирование API из ЛР2 средствами Postman приложения;

1. Создадим новую коллекцию и добавим в нее шаблонные примеры запросов



2. Добавим в переменные коллекции url адрес API и полученный JWT-token



Соответственно в запросах вместо фактического адреса будем использовать сокращенную переменную, а в авторизации полученный токен:



Token

{{jwt_token}}

3. Выгрузим нашу документацию в json формате

Export collection



lab docs will be exported as a JSON file. [Learn more about collection formats](#)

Export as:

- ☐ Collection v2
- ☒ Collection v2.1



Sending your collection to a **teammate**?
Save time by **sharing** it instead.

[Share Collection](#)

[Learn more](#)

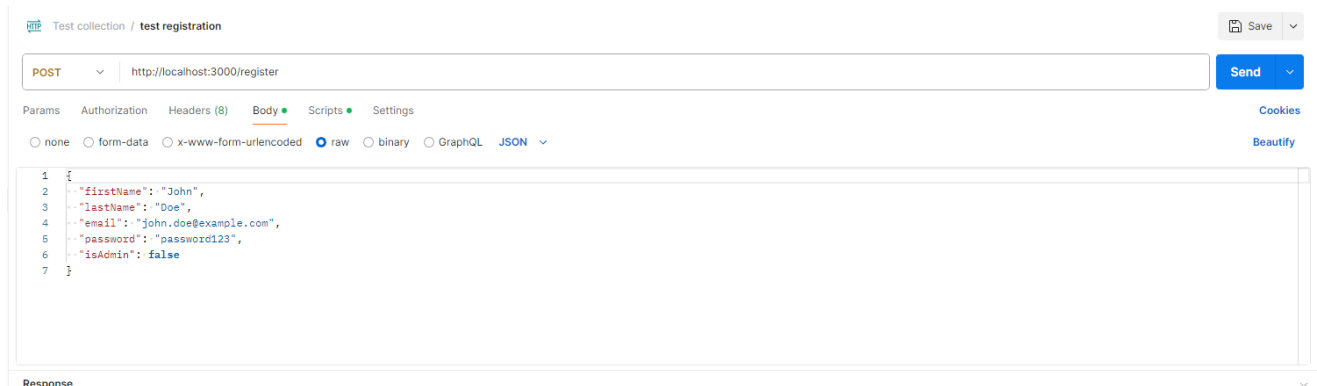
Cancel

Export

ДЗ-5:

Напишем тесты для некоторых эндпоинтов и для всего приложения в целом.

1-Создадим условного пользователя



И проверим, что данные корректно сохраняются, а сервер возвращает успех:

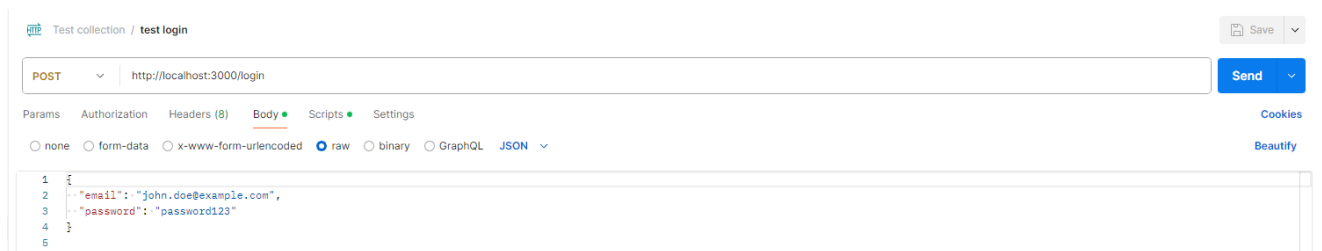


После выполнения операции имеем результат:

PASS Регистрация успешна

2-Проверим правильность работы входа

Введем корректные данные зарегистрированного пользователя



А в тесте проверим, что нам возвращается токен и что он валиден:



В результате имеем:

PASS

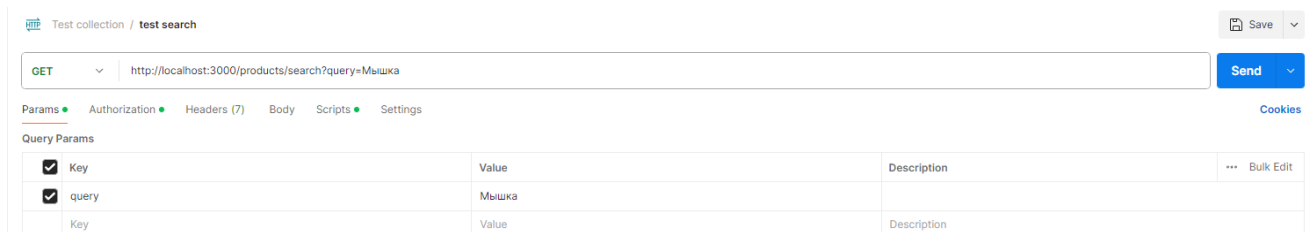
Логин успешен

PASS

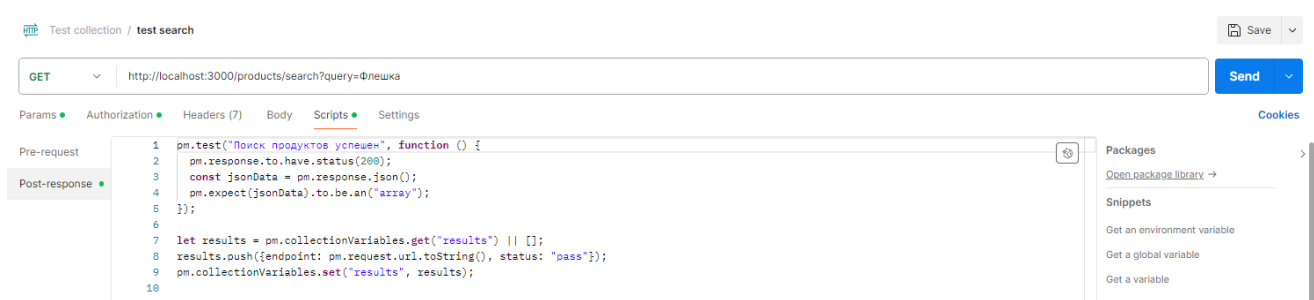
Токен валидный

3-Проверка функции поиска

Создадим корректный запрос:



Добавим тест к этому запросу:



PASS

Поиск продуктов успешен

4-Добавим новый эндпоинт для проверки всего приложения, где будем проверять успешность всех тестов выше:

Test collection / Test all

GET http://localhost:3000/health-check

Send

Params Authorization Headers (6) Body Scripts Settings

Pre-request

Post-response

```
1 pm.test("Health check успешен", function () {
2   pm.response.to.have.status(200);
3   const jsonData = pm.response.json();
4   pm.expect(jsonData).to.have.property("status", "ok");
5 });
6
7 pm.test("Все запросы выполнены успешно", function () {
8   const results = pm.collectionVariables.get("results");
9   pm.expect(results).to.be.an("array").that.is.not.empty;
10  results.forEach(result => {
11    pm.expect(result).to.have.property("status", "pass");
12  });
13 });
```

Packages

Open package library →

Snippets

Get an environment variable

Get a global variable

Get a variable

Get a collection variable

Set an environment variable

Set a global variable

Set a collection variable

Clear an environment variable

Clear a global variable

Clear a collection variable

PASS Health check успешен

PASS Все запросы выполнены успешно

Запуск всех коллекции:

Test collection - Run results

Run Again

Automate Run

New Run

Export Results

Ran today at 12:09:54 · View all runs

| Source | Environment | Iterations | Duration | All tests | Avg. Resp. Time |
|--------|-------------|------------|----------|-----------|-----------------|
| Runner | none | 1 | 1s 12ms | 6 | 32 ms |

All Tests Passed (6) Failed (0) Skipped (0) View Summary

Iteration 1

POST test registration

http://localhost:3000/register

201 Created 65 ms 484 B

PASS Регистрация успешна

POST test login

http://localhost:3000/login

200 OK 55 ms 407 B

PASS>Login успешен

PASS>Токен валидный

GET test search

http://localhost:3000/products/search?query=Мышка

200 OK 6 ms 235 B

PASS>Поиск продуктов успешен

GET Test all

http://localhost:3000/health-check

200 OK 2 ms 249 B

PASS>Health check успешен

PASS>Все запросы выполнены успешно

Вывод:

В этой домашней работе была разработана серия тестов для некоторых конечных точек, созданных в рамках лабораторной работы №2. Для тестирования API использовался инструмент Postman, который позволил эффективно создавать и выполнять тесты. Помимо этого, было подготовлено подробное описание документации для данного API, обеспечивающее четкое понимание всех доступных методов и их применения. Документация также была создана с использованием Postman, что способствовало её структурированности и легкому восприятию.