

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №4

Выполнила:

Гусейнова Марьям

БР 1.2

Проверил:

Добряков Д. И.

Санкт-Петербург

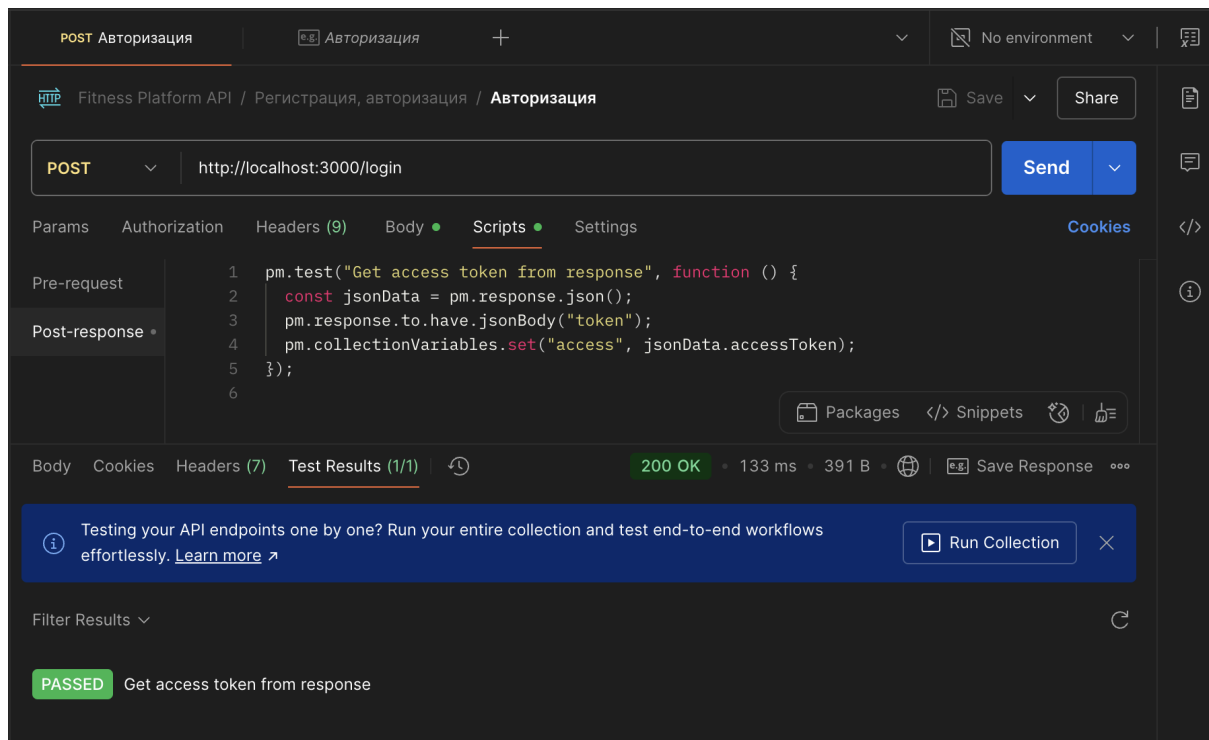
2025 г.

Задание:

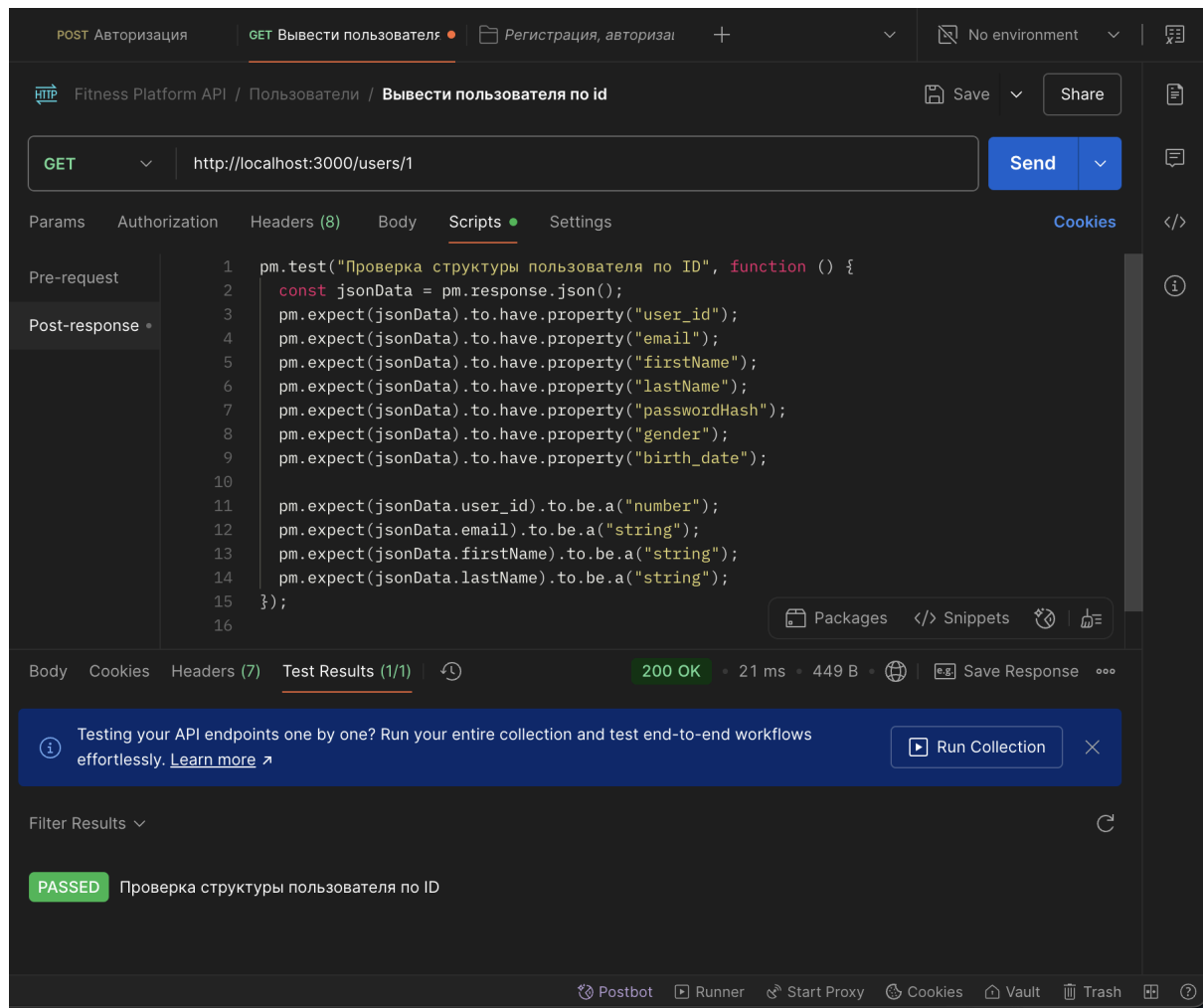
- реализовать тестирование API средствами Postman;
- написать тесты внутри Postman.

Ход работы

Первым делом был реализован тест из лекции с авторизацией:

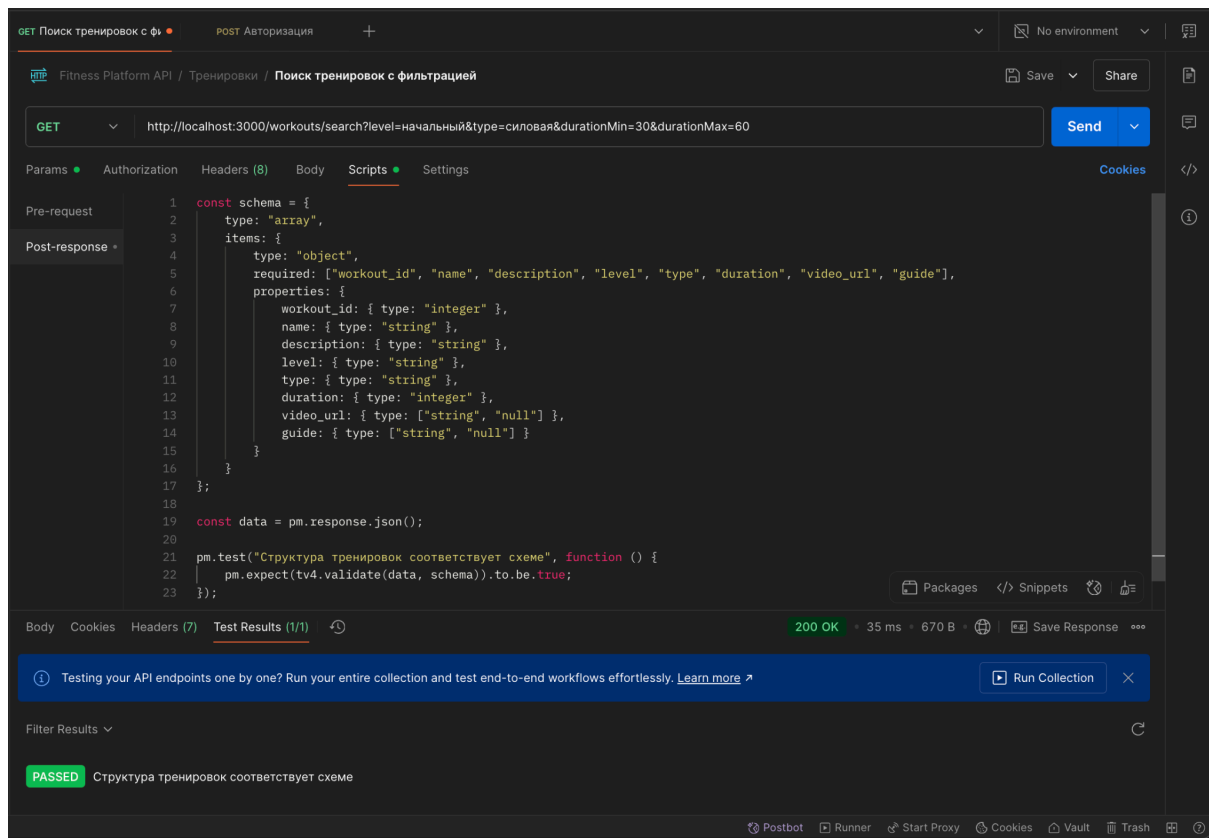


Второй тест был реализован к запросу получения пользователя по id:

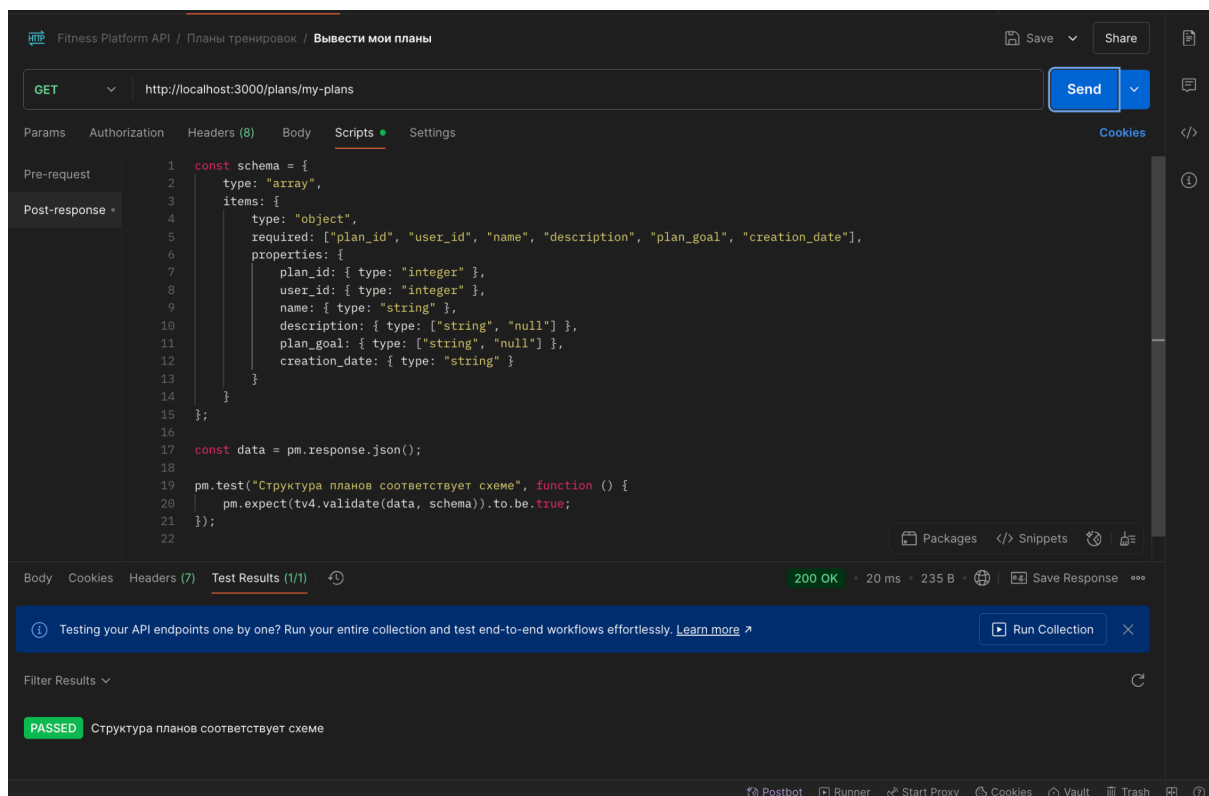


Также тут присутствует валидация некоторых полей (главные поля, которые обязательны при регистрации).

Третий тест был реализован к запросу с поиском тренировок с применением фильтров.



Четвертый тест был реализован к запросу с выводом всех планов тренировок авторизованного пользователя (вывести все мои планы):



Пятый тест был реализован к запросу с выводом прогресса по плану по id. Это вложенный запрос.

The screenshot shows the Postman interface for a GET request to `http://localhost:3000/plan-progress/2`. The **Scripts** tab is active, displaying a JSON schema for validation. The schema defines the structure of the response, including fields like `plan_progress_id`, `user_id`, `plan_id`, `plan_date`, `duration`, `notes`, and `user` with their respective data types and required status.

```
1 const schema = {
2   type: "object",
3   required: ["plan_progress_id", "user_id", "plan_id", "plan_date", "duration", "notes", "user", "workoutPlan"],
4   properties: {
5     plan_progress_id: { type: "integer" },
6     user_id: { type: "integer" },
7     plan_id: { type: "integer" },
8     plan_date: { type: "string" },
9     duration: { type: "integer" },
10    notes: { type: ["string", "null"] },
11    user: {
12      type: "object",
13      required: ["user_id", "firstName", "lastName", "email", "passwordHash", "gender", "birth_date"],
14      properties: {
15        user_id: { type: "integer" },
16        firstName: { type: "string" },
17        lastName: { type: "string" },
18        email: { type: "string" },
19        passwordHash: { type: "string" },
20        gender: { type: "string" },
21        birth_date: { type: "string" }
22      }
23    },
24    workoutPlan: {
```

The **Test Results** section shows a **PASSED** status with the message: "Структура прогресса плана соответствует схеме".

The screenshot shows the Postman interface for the same GET request. The **Scripts** tab is active, displaying a JavaScript test script that uses the `pm.test` function to validate the response against the schema defined in the previous screenshot. The script includes a `const data = pm.response.json();` line and a `pm.test` block that calls `pm.expect(tv4.validate(data, schema)).to.be.true;`.

```
22   }
23   },
24   workoutPlan: {
25     type: "object",
26     required: ["plan_id", "user_id", "name", "description", "plan_goal", "creation_date"],
27     properties: {
28       plan_id: { type: "integer" },
29       user_id: { type: "integer" },
30       name: { type: "string" },
31       description: { type: ["string", "null"] },
32       plan_goal: { type: ["string", "null"] },
33       creation_date: { type: "string" }
34     }
35   }
36 }
37 };
38
39 const data = pm.response.json();
40
41 pm.test("Структура прогресса плана соответствует схеме", function () {
42   pm.expect(tv4.validate(data, schema)).to.be.true;
43 });
```

The **Test Results** section shows a **PASSED** status with the message: "Структура прогресса плана соответствует схеме".

Вывод

Таким образом, в данной домашней работе было реализовано 5 тестов в Postman. Во всех тестах использовалась библиотека tv4 и встроенные возможности Postman.