САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №4
Тестирование API

Выполнил:
Даньшин Семён
К3340

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Реализовать тестирование API средствами Postman. Написать тесты внутри Postman для проверки функциональности API.

Ход работы

1. Структура тестов в Postman

Создана комплексная система тестирования API с использованием Postman:

Организация тестов:

```
Fitness Trainer API Tests/
├── Environment Setup/
├── Authentication Tests/
├── User Management Tests/
├── Exercise Management Tests/
├── Routine Management Tests/
├── Workout Management Tests/
└── Integration Tests/
```

2. Настройка окружения тестирования

Переменные окружения:

```
// Development Environment
{
  "baseUrl": "http://localhost:8080/api",
  "accessToken": "",
  "refreshToken": "",
  "currentUserId": "",
  "testUserId": "",
  "testRoutineId": "",
  "testWorkoutId": ""
}
```

Pre-request скрипты:

```
// Автоматическое добавление токена авторизации
if (pm.environment.get("accessToken")) {
    pm.request.headers.add({
        key: "X-Access-Token",
        value: pm.environment.get("accessToken")
    });
}
```

3. Тесты аутентификации

Тест регистрации пользователя:

```
pm.test("User registration successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();
    pm.expect(response.user).to.have.property('id');
```

```
    pm.expect(response.user).to.have.property('email');
    pm.environment.set("testUserId", response.user.id);
});
```

Тест входа в систему:

```
pm.test("Login successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.tokens).to.have.property('accessToken');
    pm.expect(response.tokens).to.have.property('refreshToken');

    pm.environment.set("accessToken", response.tokens.accessToken);
    pm.environment.set("refreshToken", response.tokens.refreshToken);
});

pm.test("Access token is valid format", function () {
    const response = pm.response.json();
    const token = response.tokens.accessToken;
    pm.expect(token).to.be.a('string');
    pm.expect(token.length).to.be.greaterThan(10);
});
```

Тест обновления токена:

```
pm.test("Token refresh successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.tokens).to.have.property('accessToken');
    pm.expect(response.tokens).to.have.property('refreshToken');

    // Проверяем, что новый токен отличается от старого
    const oldToken = pm.environment.get("accessToken");
    pm.expect(response.tokens.accessToken).to.not.equal(oldToken);

    pm.environment.set("accessToken", response.tokens.accessToken);
});
```

4. Тесты управления пользователями

Тест получения текущего пользователя:

```
pm.test("Get current user successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.user).to.have.property('id');
    pm.expect(response.user).to.have.property('email');
    pm.expect(response.user).to.have.property('firstName');
    pm.expect(response.user).to.have.property('lastName');

    pm.environment.set("currentUserId", response.user.id);
});
```

Тест обновления профиля:

```
pm.test("User profile update successful", function () {
    pm.response.to.have.status(200);
```

```
    const response = pm.response.json();

    pm.expect(response.user.firstName).to.equal("UpdatedName");
    pm.expect(response.user).to.have.property('updatedAt');
});
```

5. Тесты управления упражнениями

Тест создания упражнения:

```
pm.test("Exercise creation successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.exercise).to.have.property('id');
    pm.expect(response.exercise.name).to.equal("Test Exercise");
    pm.expect(response.exercise.targetMuscleGroups).to.be.an('array');

    pm.environment.set("testExerciseId", response.exercise.id);
});
```

Тест получения списка упражнений:

```
pm.test("Get exercises list successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.exercises).to.be.an('array');
    if (response.exercises.length > 0) {
        pm.expect(response.exercises[0]).to.have.property('id');
        pm.expect(response.exercises[0]).to.have.property('name');
    }
});
```

6. Тесты управления рутинами

Тест создания рутины:

```
pm.test("Routine creation successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.routine).to.have.property('id');
    pm.expect(response.routine.name).to.equal("Test Routine");

pm.expect(response.routine.userId).to.equal(pm.environment.get("currentUser
Id"));

    pm.environment.set("testRoutineId", response.routine.id);
});
```

Тест добавления упражнения в рутину:

```
pm.test("Add exercise to routine successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.exerciseInstance).to.have.property('id');
```

```
pm.expect(response.exerciseInstance.routineId).to.equal(pm.environment.get(
"testRoutineId"));
});
```

7. Тесты управления тренировками

Тест начала тренировки:

```
pm.test("Start workout successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.workout).to.have.property('id');

pm.expect(response.workout.userId).to.equal(pm.environment.get("currentUser
Id"));
    pm.expect(response.workout.finishedAt).to.be.null;

    pm.environment.set("testWorkoutId", response.workout.id);
});
```

Тест логирования упражнения:

```
pm.test("Log exercise successful", function () {
    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response).to.have.property('id');

pm.expect(response.workoutId).to.equal(pm.environment.get("testWorkoutId"))
;

pm.expect(response.exerciseId).to.equal(pm.environment.get("testExerciseId"
));
});
```

8. Интеграционные тесты

Полный цикл тренировки:

```
pm.test("Complete workout flow", function () {
    // Тест проверяет полный цикл:
    // 1. Создание рутины
    // 2. Добавление упражнений
    // 3. Начало тренировки
    // 4. Логирование упражнений
    // 5. Завершение тренировки

    pm.response.to.have.status(200);
    const response = pm.response.json();

    pm.expect(response.workout.finishedAt).to.not.be.null;
    pm.expect(response.exerciseLogs).to.be.an('array');
    pm.expect(response.exerciseLogs.length).to.be.greaterThan(0);
});
```

9. Тесты обработки ошибок

Тест неавторизованного доступа:

```
pm.test("Unauthorized access returns 401", function () {
    pm.response.to.have.status(401);
    const response = pm.response.json();
    pm.expect(response).to.have.property('message');
});
```

Тест валидации данных:

```
pm.test("Invalid data returns 400", function () {
    pm.response.to.have.status(400);
    const response = pm.response.json();
    pm.expect(response).to.have.property('message');
    pm.expect(response.message).to.include('validation');
});
```

10. Производительность тестов

Тест времени ответа:

```
pm.test("Response time is acceptable", function () {
    pm.expect(pm.response.responseTime).to.be.below(1000);
});
```

Вывод

Реализована комплексная система тестирования API:

1. **Полное покрытие** всех основных функций API тестами
2. **Автоматизированная настройка** окружения и переменных
3. **Интеграционные тесты** для проверки бизнес-процессов
4. **Тесты безопасности** и обработки ошибок
5. **Проверка производительности** и времени ответа

Система тестирования обеспечивает надежную проверку функциональности API и может быть интегрирована в процесс непрерывной разработки.