федеральное государственное автономное образовательное учреждение высшего образования

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № _5_

Дисциплина: Тестирование программного обеспечения

Тема: Тестирование REST API с помощью Postman и Python

Выполнил(а): студент(ка) группы __221-3711__

Морозов К.А.
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

Проверил: _____ _____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания: _____

Москва 2025

Лабораторная работа 5 – Тестирование REST API с помощью Postman и Python

Цель работы

Освоить основы тестирования API: отправка запросов, валидация ответов, написание скриптов для автоматических проверок.

Задание

Объект тестирования: Публичное REST API (например, https://reqres.in/ или https://jsonplaceholder.typicode.com/).
Задание (Ручное тестирование):
1. В Postman создайте коллекцию запросов: GET (получить пользователя), POST (создать пользователя), PUT (обновить).

2. Для каждого запроса напишите тесты на JavaScript (проверка status code, структуры JSON, значений полей).

3. Запустите коллекцию как collection run и проанализируйте результаты.

Задание (Автоматическое тестирование):
1. Напишите скрипт на Python с использованием библиотеки `requests` и `pytest`, который повторяет логику тестов из Postman.

Практическая часть

В качестве объекта лабораторной работы я выбрал сайт https://reqres.in/. В Postman я создал для него отдельную коллекцию с тремя запросами: Get, Post и Put. В каждый из них я вставил нужный адрес запроса (/api/users/2 для Get и Put, /api/users для Post). А также, чтобы запросы отправились без ошибки, я вставил в header параметр x-api-key = reqres-free-v1.

Для каждого запроса я написал 3 похожих скрипта для автотеста: проверка статуса, равняется ли параметр заданному и есть ли определенная строка в ответе:

Get

```
pm.test("Status 200", () => pm.response.to.have.status(200));
pm.test("data.id = 2", () => pm.expect(pm.response.json().data.id).to.eql(2));
pm.test("Has email", () => pm.expect(pm.response.json().data.email).to.match(/@/));
```

Post

```
pm.test("Status 201", () => pm.response.to.have.status(201));
pm.test("name = test1", () => pm.expect(pm.response.json().name).to.eql("test1"));
pm.test("Has id", () => pm.expect(pm.response.json().id).to.be.a("string"));
```

Put

```
pm.test("Status 200", () => pm.response.to.have.status(200));
pm.test("name = test3", () => pm.expect(pm.response.json().name).to.eql("test3"));
pm.test("Has updatedAt", () =>
pm.expect(pm.response.json().updatedAt).to.be.a("string"));
```

Также для двух запросов я написал тело в виде json-строки:

Post

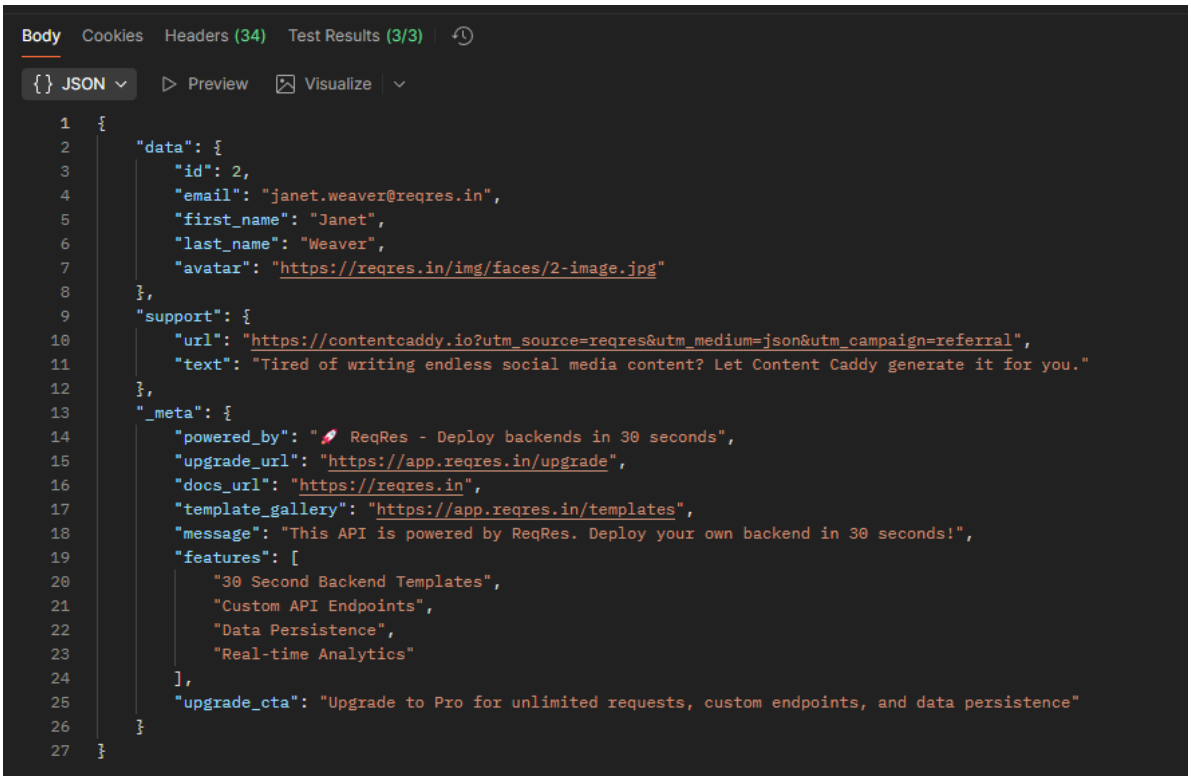
```
{ "name": "test1", "job": "test2" }
```

Put

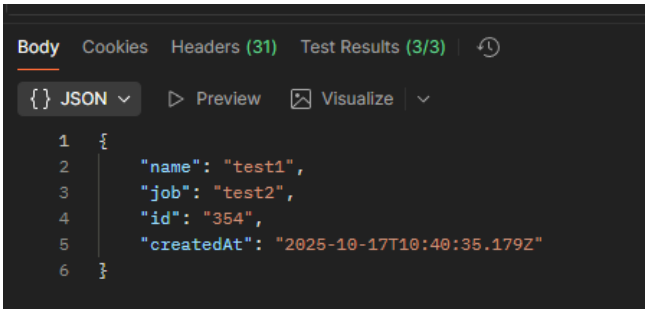
```
{ "name": "test3", "job": "test4" }
```

Со всех запросов пришел положительный ответ, а также все тесты были успешно выполнены:

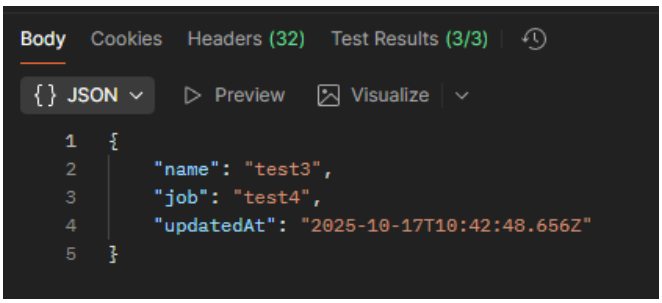
Get



Post



Put



Также по заданию лабораторной работы нужно было написать скрипт на python, копирующий все то, что мы проделали в postman. У меня получилась следующая конструкция:

```python
import requests

def test_get_user_2():
    r = requests.get(f"https://reqres.in/api/users/2", headers={"x-api-key": "reqres-free-v1"}, timeout=10)
    assert r.status_code == 200, f"Unexpected status: {r.status_code}"
    data = r.json().get("data", {})
    assert data.get("id") == 2, "User ID mismatch"
    assert "email" in data and "@" in data["email"], "error"

def test_post_create_user():
    payload = {"name": "test1", "job": "test2"}
    r = requests.post(f"https://reqres.in/api/users", json=payload, headers={"x-api-key": "reqres-free-v1"}, timeout=10)
    assert r.status_code == 201, f"Unexpected status: {r.status_code}"
    j = r.json()
    assert j.get("name") == "test1", "Name mismatch in response"
    assert isinstance(j.get("id"), str), "error"

def test_put_update_user():
    payload = {"name": "test3", "job": "test4"}
    r = requests.put(f"https://reqres.in/api/users/2", json=payload, headers={"x-api-key": "reqres-free-v1"}, timeout=10)
    assert r.status_code == 200, f"Unexpected status: {r.status_code}"
    j = r.json()
    assert j.get("name") == "test3", "Name mismatch in PUT response"
    assert isinstance(j.get("updatedAt"), str), "error"
```

Код на пайтоне также успешно выполняется:

```
C:\Users\sacho\OneDrive\Рабочий стол\test5>pytest -q
...
3 passed in 3.74s

C:\Users\sacho\OneDrive\Рабочий стол\test5>
```

Ссылка на github с кодом лабораторной:

https://github.com/Sollimba/Testing_5_TPO_Morozov

Ссылка на яндекс диск с видео-защитой:

https://disk.yandex.ru/d/Jg3tB_oH4RDkOA