

**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

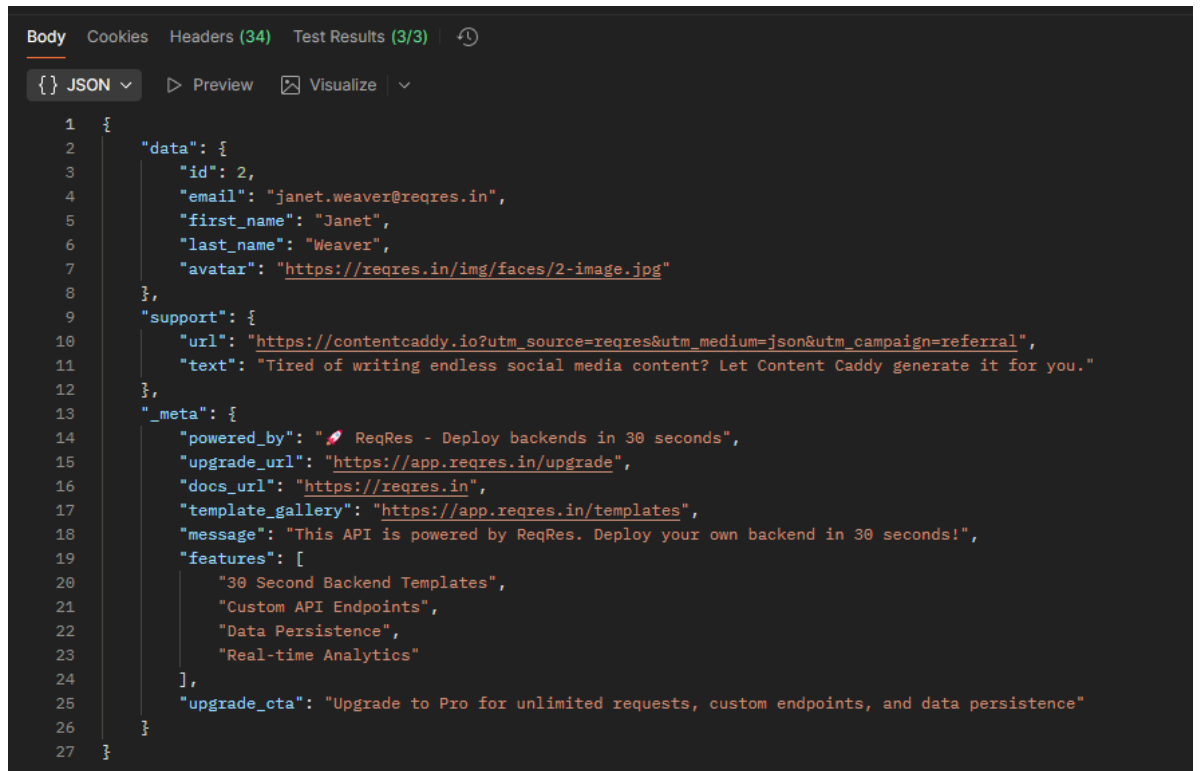
```
{ "name": "test1", "job": "test2" }
```

### Put

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

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

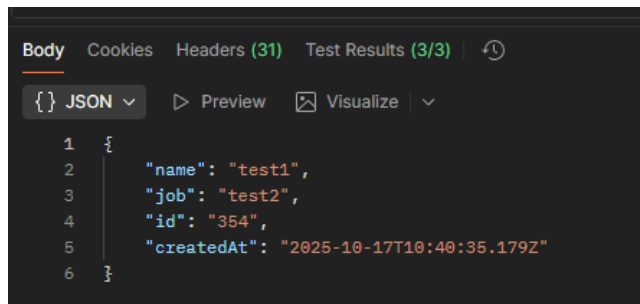
## Get



The screenshot shows a REST client interface with the 'Body' tab selected. The response is a JSON object with the following structure:

```
1 {
2   "data": {
3     "id": 2,
4     "email": "janet.weaver@reqres.in",
5     "first_name": "Janet",
6     "last_name": "Weaver",
7     "avatar": "https://reqres.in/img/faces/2-image.jpg"
8   },
9   "support": {
10    "url": "https://contentcaddy.io?utm_source=reqres&utm_medium=json&utm_campaign=referral",
11    "text": "Tired of writing endless social media content? Let Content Caddy generate it for you."
12  },
13  "_meta": {
14    "powered_by": "🔥 ReqRes - Deploy backends in 30 seconds",
15    "upgrade_url": "https://app.reqres.in/upgrade",
16    "docs_url": "https://reqres.in",
17    "template_gallery": "https://app.reqres.in/templates",
18    "message": "This API is powered by ReqRes. Deploy your own backend in 30 seconds!",
19    "features": [
20      "30 Second Backend Templates",
21      "Custom API Endpoints",
22      "Data Persistence",
23      "Real-time Analytics"
24    ],
25    "upgrade_cta": "Upgrade to Pro for unlimited requests, custom endpoints, and data persistence"
26  }
27 }
```

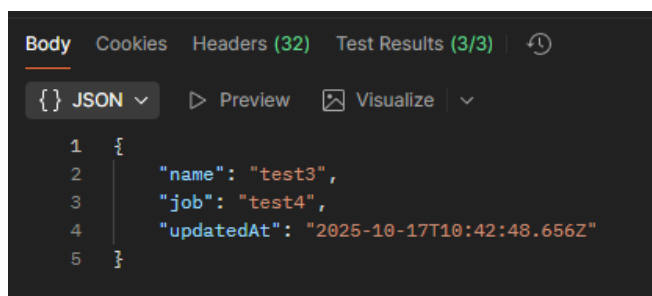
## Post



The screenshot shows a REST client interface with the 'Body' tab selected. The response is a JSON object with the following structure:

```
1 {
2   "name": "test1",
3   "job": "test2",
4   "id": "354",
5   "createdAt": "2025-10-17T10:40:35.179Z"
6 }
```

## Put



The screenshot shows a REST client interface with the 'Body' tab selected. The response is a JSON object with the following structure:

```
1 {
2   "name": "test3",
3   "job": "test4",
4   "updatedAt": "2025-10-17T10:42:48.656Z"
5 }
```

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

```
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](https://github.com/Sollimba/Testing_5_TPO_Morozov)

Ссылка на яндекс диск с видео-защитой:

[https://disk.yandex.ru/d/Jg3tB\\_oH4RDkOA](https://disk.yandex.ru/d/Jg3tB_oH4RDkOA)