**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: Лабораторная работа 5: Тестирование REST API с помощью Postman и Python**

**Выполнил: студент группы 221-371**

**Евдокимов Артемий Русланович**

**(Фамилия И.О )**

**Дата, подпись \_\_21.09.2025\_\_ *\_\_\_\_\_\_\_\_\_\_\_***

**(Дата) (Подпись)**

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**(Фамилия И.О., степень, звание)**

**Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_***

**(Дата) (Подпись)**

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2025**

**Цель работы**

Освоить основы тестирования API: отправка запросов, валидация ответов, написание скриптов для автоматических проверок с использованием библиотек requests и pytest.

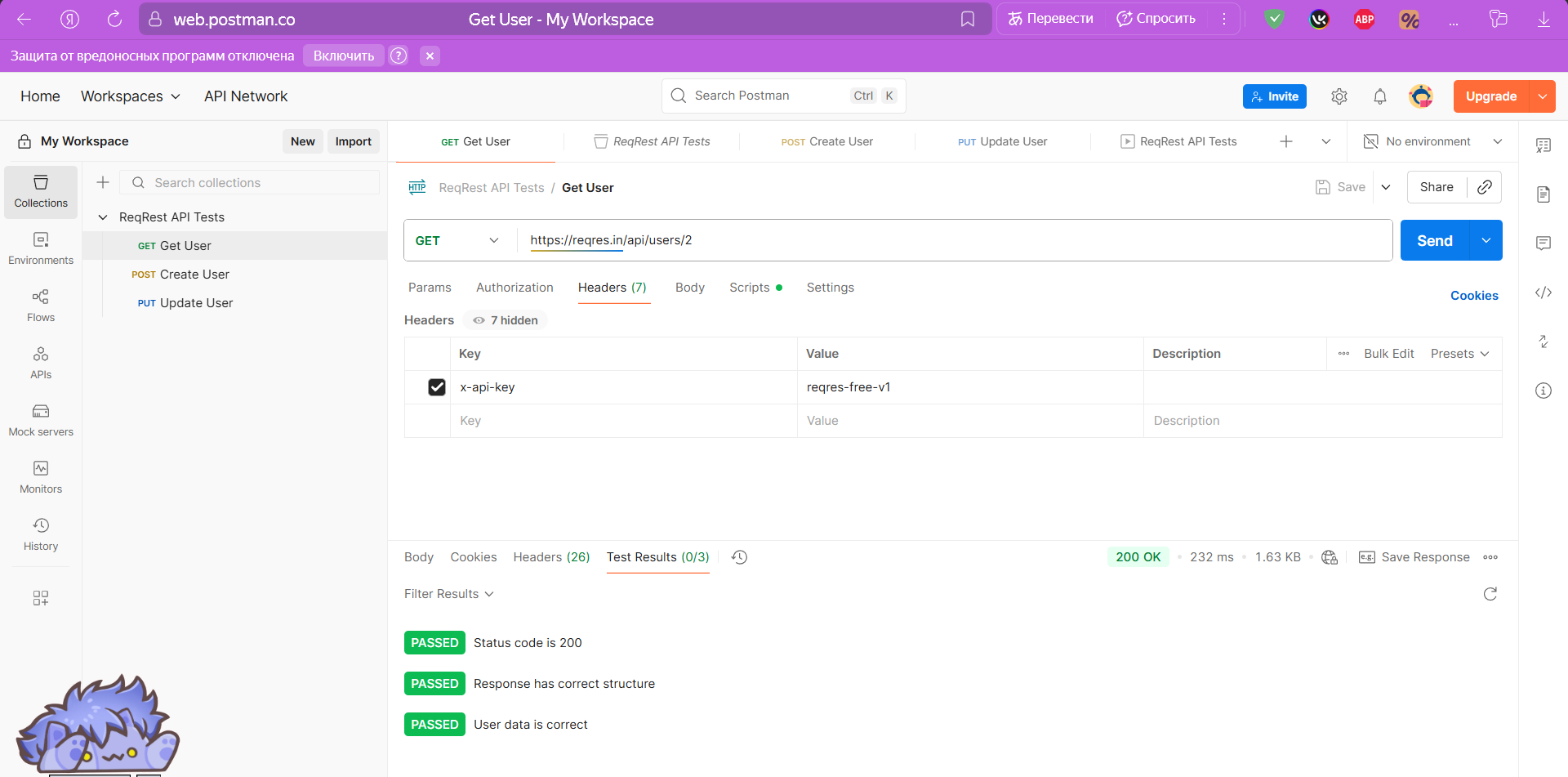
**Объект тестирования**

Публичное REST API <https://reqres.in/>

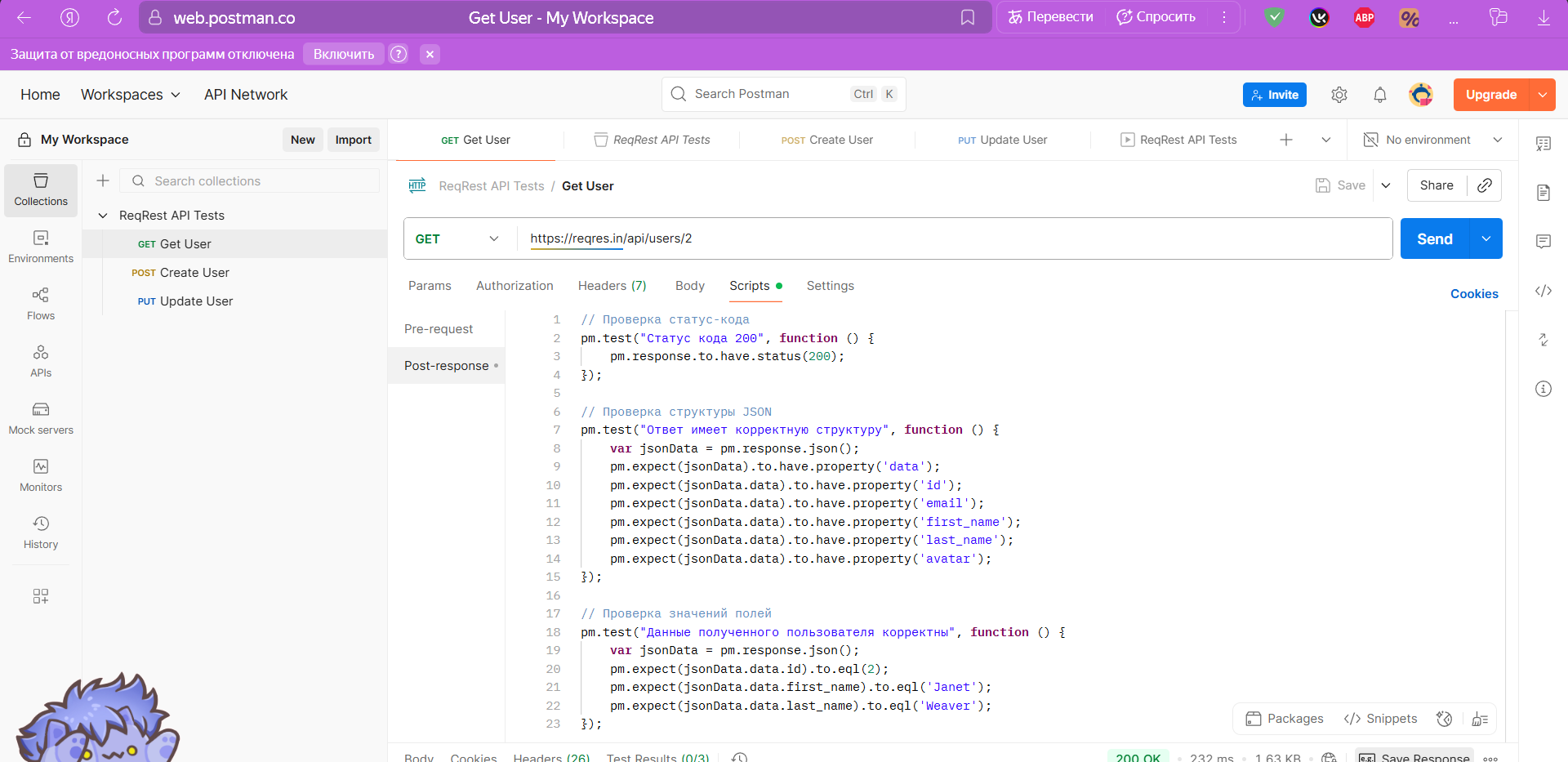
**Задание**

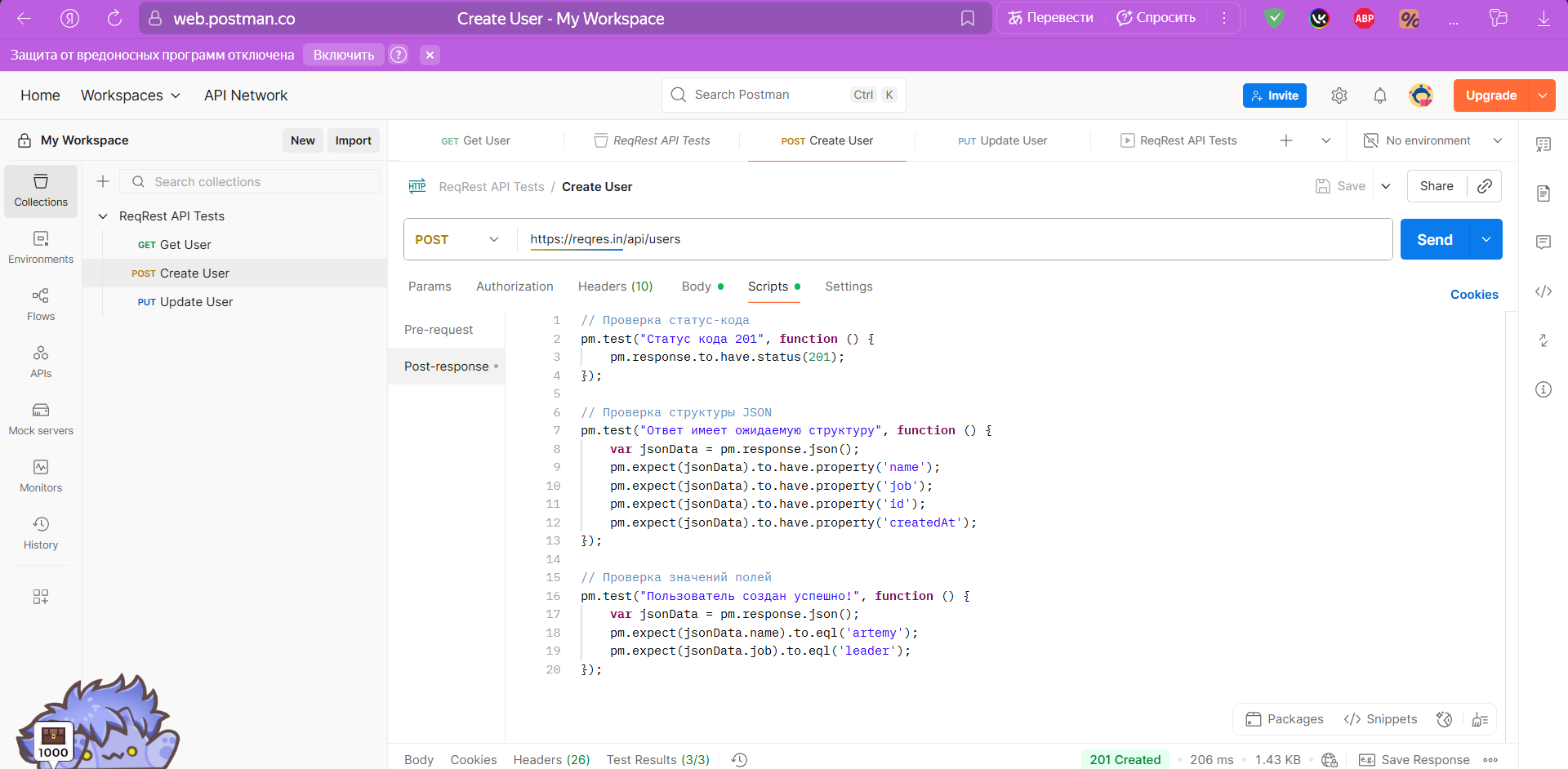
**Ручное тестирование:**

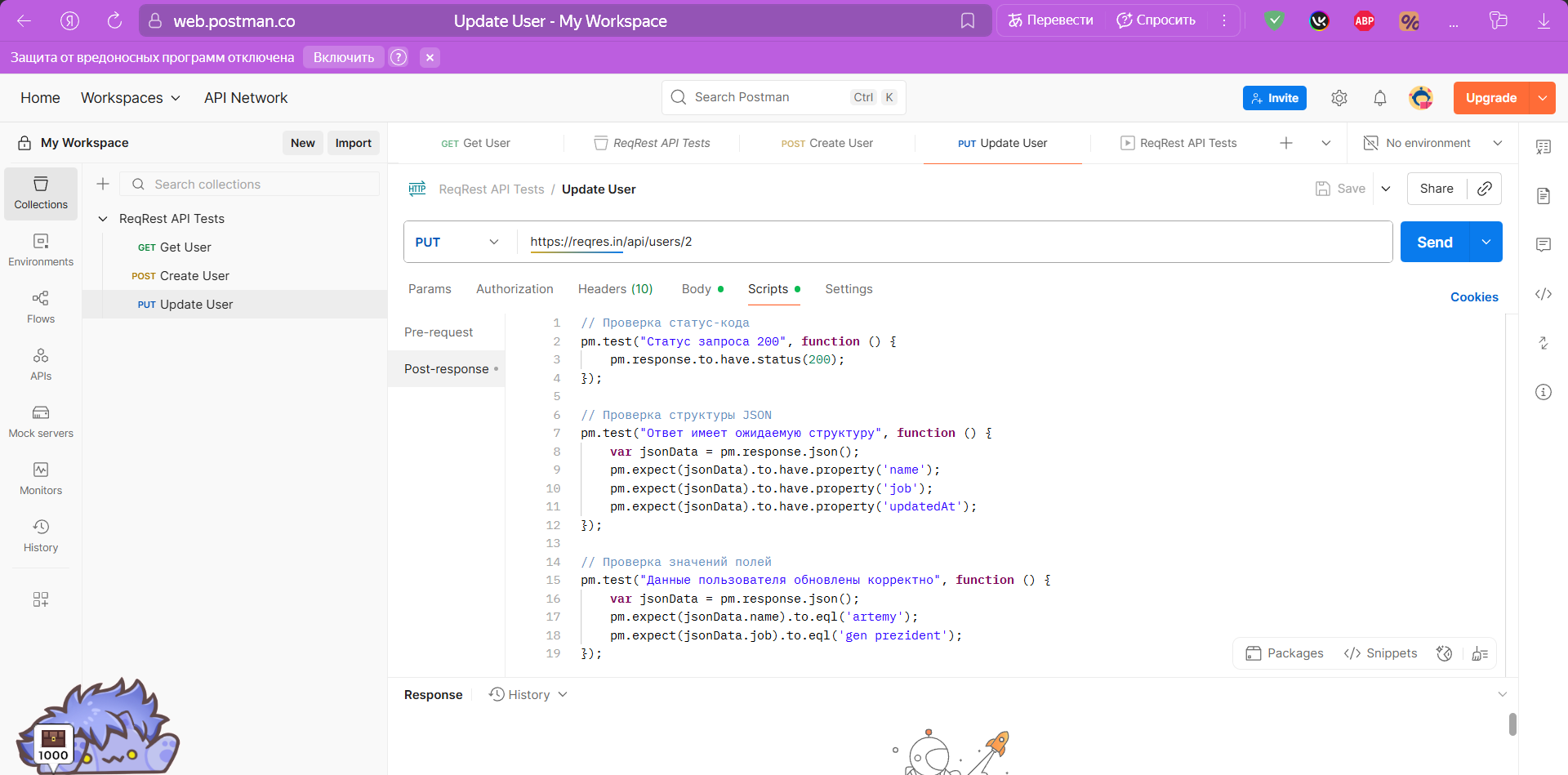
1. В Postman создать коллекцию запросов: GET (получить пользователя), POST (создать пользователя), PUT (обновить)



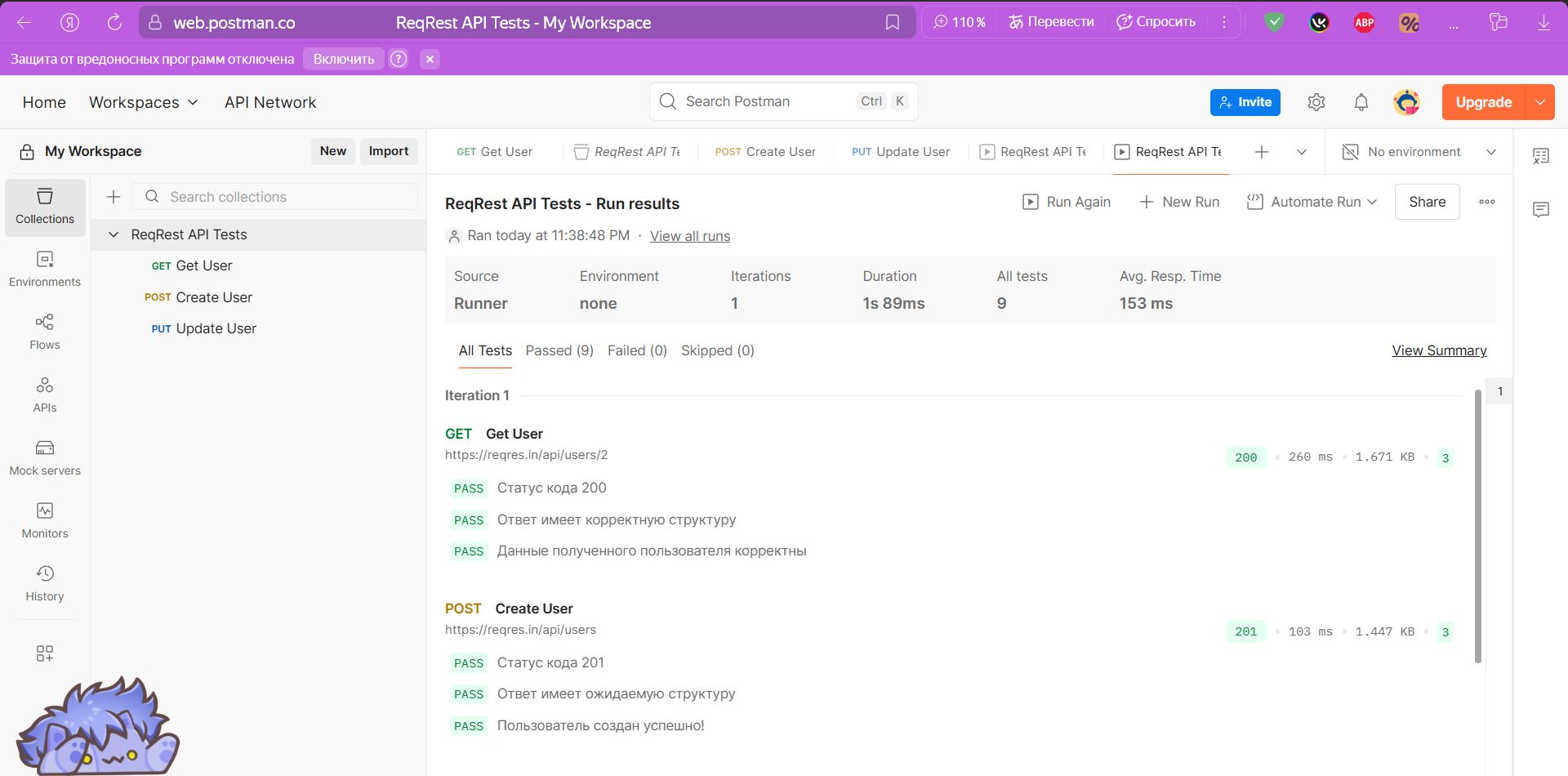
1. Для каждого запроса написать тесты на JavaScript (проверка status code, структуры JSON, значений полей)

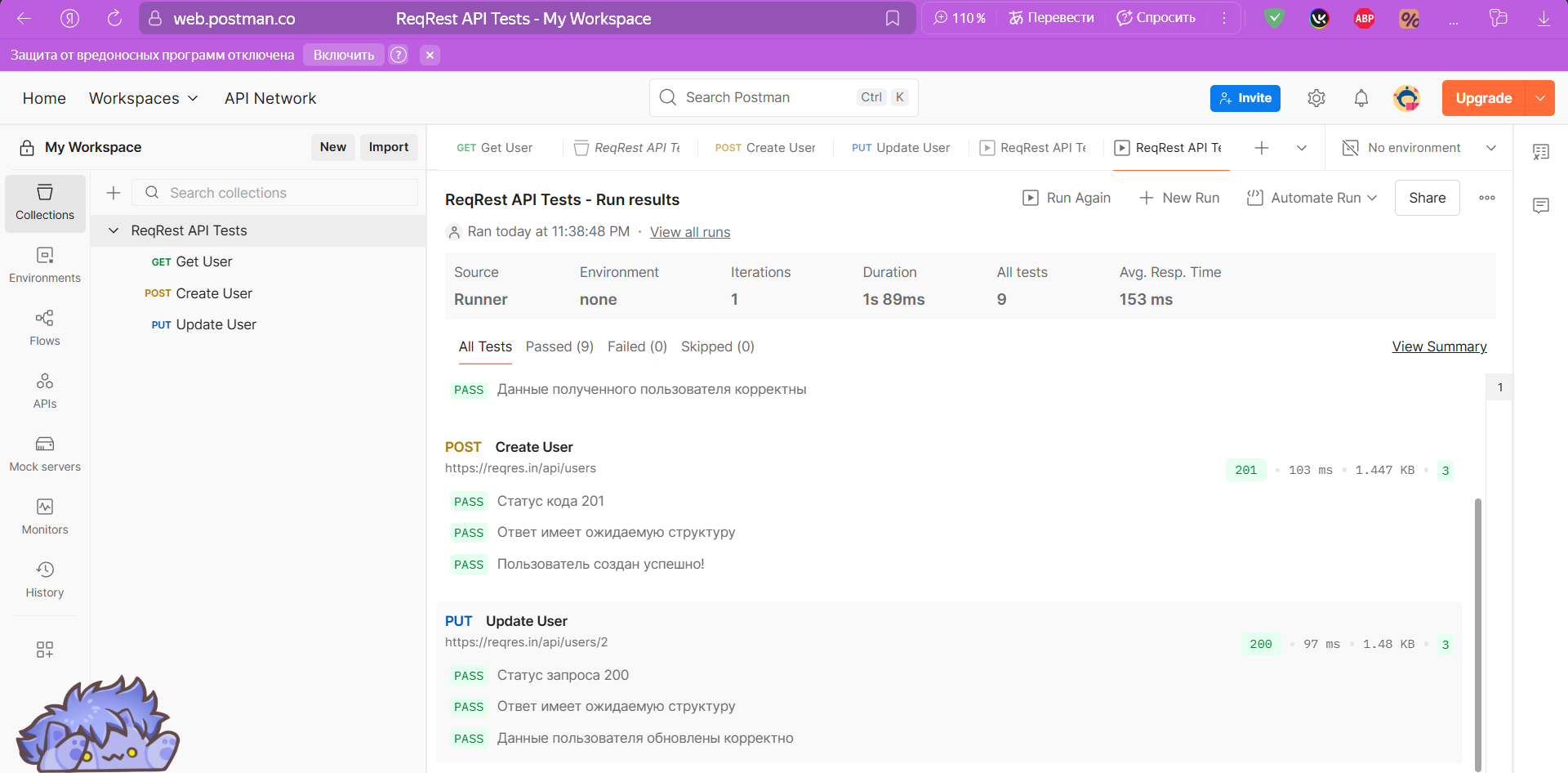






1. Запустить коллекцию как collection run и проанализировать результаты





**Автоматическое тестирование:**

1. Написать скрипт на Python с использованием библиотек requests и pytest, который повторяет логику тестов из Postman

**Реализация**

**Код автоматических тестов на Python:**

import pytest

import requests

import sys

BASE\_URL = "https://reqres.in/api"

class TestReqResAPI:

"""Тесты для API reqres.in с использованием pytest и requests"""

def \_print\_result(self, message, status="INFO"):

"""Вывод результата в stderr"""

print(f"\n{status}: {message}", file=sys.stderr)

sys.stderr.flush()

@pytest.fixture

def api\_headers(self):

"""Фикстура с заголовками для API запросов"""

self.\_print\_result("Подготавливаем заголовки для API запросов")

return {

"x-api-key": "reqres-free-v1",

"Content-Type": "application/json"

}

def test\_get\_user(self, api\_headers):

"""Тест GET запроса для получения пользователя"""

self.\_print\_result("Выполняем GET запрос для получения пользователя")

response = requests.get(f"{BASE\_URL}/users/2", headers=api\_headers)

*# Проверка статус-кода*

assert response.status\_code == 200

self.\_print\_result(f"Статус-код: {response.status\_code} - OK", "SUCCESS")

*# Проверка структуры JSON*

json\_data = response.json()

assert "data" in json\_data

assert all(key in json\_data["data"] for key in ["id", "email", "first\_name", "last\_name", "avatar"])

self.\_print\_result("Структура JSON ответа корректна", "SUCCESS")

*# Проверка значений полей*

assert json\_data["data"]["id"] == 2

assert json\_data["data"]["first\_name"] == "Janet"

assert json\_data["data"]["last\_name"] == "Weaver"

self.\_print\_result("Значения полей соответствуют ожидаемым", "SUCCESS")

self.\_print\_result("GET тест пройден успешно", "SUCCESS")

def test\_create\_user(self, api\_headers):

"""Тест POST запроса для создания пользователя"""

self.\_print\_result("Выполняем POST запрос для создания пользователя")

payload = {

"name": "artemy",

"job": "leader"

}

response = requests.post(f"{BASE\_URL}/users", json=payload, headers=api\_headers)

# Проверка статус-кода

assert response.status\_code == 201

self.\_print\_result(f"Статус-код: {response.status\_code} - Created", "SUCCESS")

# Проверка структуры JSON

json\_data = response.json()

assert all(key in json\_data for key in ["name", "job", "id", "createdAt"])

self.\_print\_result("Структура JSON ответа корректна", "SUCCESS")

# Проверка значений полей

assert json\_data["name"] == payload["name"]

assert json\_data["job"] == payload["job"]

self.\_print\_result("Значения полей соответствуют ожидаемым", "SUCCESS")

self.\_print\_result("POST тест пройден успешно", "SUCCESS")

def test\_update\_user(self, api\_headers):

"""Тест PUT запроса для обновления пользователя"""

self.\_print\_result("Выполняем PUT запрос для обновления пользователя")

payload = {

"name": "artemy",

"job": "gen prezident"

}

response = requests.put(f"{BASE\_URL}/users/2", json=payload, headers=api\_headers)

# Проверка статус-кода

assert response.status\_code == 200

self.\_print\_result(f"Статус-код: {response.status\_code} - OK", "SUCCESS")

# Проверка структуры JSON

json\_data = response.json()

assert all(key in json\_data for key in ["name", "job", "updatedAt"])

self.\_print\_result("Структура JSON ответа корректна", "SUCCESS")

# Проверка значений полей

assert json\_data["name"] == payload["name"]

assert json\_data["job"] == payload["job"]

self.\_print\_result("Значения полей соответствуют ожидаемым", "SUCCESS")

self.\_print\_result("PUT тест пройден успешно", "SUCCESS")

if \_\_name\_\_ == "\_\_main\_\_":

# Запуск pytest с аргументами

pytest.main([\_\_file\_\_, "-v", "-s"])

**Результаты тестирования**

**Успешно выполненные тесты:**

1. **GET запрос** - получение пользователя
   * Статус-код: 200
   * Структура JSON ответа корректна
   * Значения полей соответствуют ожидаемым
   * Результат: "GET тест пройден успешно"
2. **POST запрос** - создание пользователя
   * Статус-код: 201
   * Структура JSON ответа корректна
   * Значения полей соответствуют ожидаемым
   * Результат: "POST тест пройден успешно"
3. **PUT запрос** - обновление пользователя
   * Статус-код: 200
   * Структура JSON ответа корректна
   * Значения полей соответствуют ожидаемым
   * Результат: "PUT тест пройден успешно"

**Статистика тестирования:**

* Всего тестов: 3
* Успешно пройдено: 3
* Провалено: 0
* Пропущено: 0

**Выводы**

В ходе лабораторной работы были успешно выполнены все поставленные задачи:

1. Реализованы класс api запросов с автоматическими тестами в Postman.
2. Реализованы автоматические тесты для REST API с использованием библиотек requests и pytest
3. Протестированы три основных типа HTTP-запросов: GET, POST и PUT
4. Реализованы проверки статус-кодов, структуры JSON-ответов и значений полей

Основные преимущества реализованного подхода:

* Использование стандартных библиотек Python для тестирования API
* Четкая структура тестов с использованием паттерна Page Object
* Подробное логирование процесса выполнения тестов
* Простота запуска и интерпретации результатов

Разработанные тесты могут быть легко расширены для проверки дополнительных сценариев работы с API и интегрированы в процесс непрерывной интеграции.