**Задание 2**

|  |  |
| --- | --- |
| Название программы | **«Инженер - тестировщик»** |
| Группа обучения | **«ИТ-3»** |
| Срок обучения | **«7 июля 2025 – 8 августа 2025»** |
| **«‎Асакаева Рена Алиевна»** | |
| Номер/Название Кейса | **«Разработка и запуск простого автоматизированного теста для проверки API»** |

Москва 2025 г.

API

Импортируем библиотеки:

import pytest

import requests

import json

Объявляем базовые данные:

BASE\_URL = "https://petstore.swagger.io/v2"

Пишем код для заголовка теста:

def print\_test\_header(test\_name):

    print(f"\n{'=' \* 50}")

    print(f"ТЕСТ: {test\_name}")

    print('=' \* 50)

Делаем форматированный вывод:

def print\_request\_response(request\_method, url, response, request\_data=None):

    print(f"\n[ЗАПРОС]")

    print(f"Метод: {request\_method}")

    print(f"URL: {url}")

    if request\_data:

        print("Тело запроса:")

        print(json.dumps(request\_data, indent=2, ensure\_ascii=False))

    print(f"\n[ОТВЕТ]")

    print(f"Статус код: {response.status\_code}")

    if response.text:

        try:

            print("Тело ответа:")

            print(json.dumps(response.json(), indent=2, ensure\_ascii=False))

        except ValueError:

            print(response.text)

Пишем сам тест для создания и получения питомца:

class TestPetstoreAPI:

    def test\_create\_and\_get\_pet(self):

        print\_test\_header("Создание и получение питомца")

        1. Создаем нового питомца

        new\_pet = {

            "id": 987654321,

            "name": "TestDog",

            "status": "available"

        }

        POST запрос

        post\_url = f"{BASE\_URL}/pet"

        post\_response = requests.post(post\_url, json=new\_pet)

        print\_request\_response("POST", post\_url, post\_response, new\_pet)

        assert post\_response.status\_code == 200

        Получаем созданного питомца

        pet\_id = new\_pet["id"]

        get\_url = f"{BASE\_URL}/pet/{pet\_id}"

        get\_response = requests.get(get\_url)

        print\_request\_response("GET", get\_url, get\_response)

        Пишем проверки:

        assert get\_response.status\_code == 200

        pet\_data = get\_response.json()

        assert pet\_data["id"] == pet\_id

        assert pet\_data["name"] == new\_pet["name"]

        assert pet\_data["status"] == new\_pet["status"]

        print("\nРЕЗУЛЬТАТ: УСПЕХ (питомец создан и получен)")

        И, пожалуй, удалим за собой питомца:

        requests.delete(f"{BASE\_URL}/pet/{pet\_id}")

Далее, сделаем проверку заведомо несуществующего питомца:

    def test\_get\_nonexistent\_pet(self):

        print\_test\_header("Запрос несуществующего питомца")

        non\_existent\_id = 999999999

        url = f"{BASE\_URL}/pet/{non\_existent\_id}"

        response = requests.get(url)

        print\_request\_response("GET", url, response)

        assert response.status\_code == 404

        print("\nРЕЗУЛЬТАТ: УСПЕХ (корректная обработка отсутствия питомца)")

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

    pytest.main(["-v", "--no-header", "-rP"])

**Получаем такой результат:**

PS C:\api-tests> & C:/Users/fierf/AppData/Local/Programs/Python/Python313/python.exe c:/api-tests/test\_petstore.py

============================= test session starts =============================

collected 2 items

test\_petstore.py::TestPetstoreAPI::test\_create\_and\_get\_pet PASSED [ 50%]

test\_petstore.py::TestPetstoreAPI::test\_get\_nonexistent\_pet PASSED [100%]

=================================== PASSES ====================================

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TestPetstoreAPI.test\_create\_and\_get\_pet \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

---------------------------- Captured stdout call -----------------------------

==================================================

ТЕСТ: Создание и получение питомца

==================================================

[ЗАПРОС]

Метод: POST

URL: https://petstore.swagger.io/v2/pet

Тело запроса:

{

"id": 987654321,

"name": "TestDog",

"status": "available"

}

[ОТВЕТ]

Статус код: 200

Тело ответа:

{

"id": 987654321,

"name": "TestDog",

"photoUrls": [],

"tags": [],

"status": "available"

}

[ЗАПРОС]

Метод: GET

URL: https://petstore.swagger.io/v2/pet/987654321

[ОТВЕТ]

Статус код: 200

Тело ответа:

{

"id": 987654321,

"name": "TestDog",

"photoUrls": [],

"tags": [],

"status": "available"

}

РЕЗУЛЬТАТ: УСПЕХ (питомец создан и получен)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ TestPetstoreAPI.test\_get\_nonexistent\_pet \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

---------------------------- Captured stdout call -----------------------------

==================================================

ТЕСТ: Запрос несуществующего питомца

==================================================

[ЗАПРОС]

Метод: GET

URL: https://petstore.swagger.io/v2/pet/999999999

[ОТВЕТ]

Статус код: 404

Тело ответа:

{

"code": 1,

"type": "error",

"message": "Pet not found"

}

РЕЗУЛЬТАТ: УСПЕХ (корректная обработка отсутствия питомца)

============================== 2 passed in 2.61s ==============================

PS C:\api-tests>

**Выводы:**

1. **Тест создания и получения питомца (test\_create\_and\_get\_pet)**

Успешно создан новый питомец (POST-запрос)

* + - ID: 987654321
    - Имя: TestDog
    - Статус: available
    - Статус код: 200 (OK)

Успешно получены данные созданного питомца (GET-запрос)

* + - Все данные совпадают с отправленными
    - Статус код: 200 (OK)

Результат: "УСПЕХ (питомец создан и получен)"

1. **Тест запроса несуществующего питомца (test\_get\_nonexistent\_pet)**

Отправлен запрос для несуществующего ID (999999999)

Получен ожидаемый ответ:

* + - Статус код: 404 (Not Found)
    - Сообщение об ошибке: "Pet not found"

Результат: "УСПЕХ (корректная обработка отсутствия питомца)"

**Таким образом:**

* Оба теста прошли успешно
* Общее время выполнения: 2.61 секунды
* Все проверки статус-кодов и содержимого ответов выполнены
* Тестовая среда работает корректно
* API правильно обрабатывает как существующих, так и несуществующих питомцев

**Вывод**: тестовая система работает стабильно, все проверки пройдены, API Petstore функционирует согласно ожиданиям.

Для генерации HTML-отчёта добавим параметр *--html=report.html* в строку

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

pytest.main(["-v", "--html=report.html", "--no-header"])

**После запуска теста сформировался HTML-отчёт:**

