

**Федеральное государственное бюджетное
образовательное учреждение высшего образования**

**«Белгородский государственный технологический
университет им. В.Г.Шухова»**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

**Методические указания к лабораторной работе №
3. Тестирование REST API**

по дисциплине «Тестирование программных систем»

Выполнил: ст. группы ПВ-202

Буйвало А.А.

Проверили:

Лебединская Анастасия Александровна

Бабенко Анастасия Александровна

Белгород, 2023

Лабораторная работа №3

Тестирование REST API

1. Цель работы

Научиться проектировать тестовые сценарии для тестирования REST API.

Познакомиться с инструментом тестирования REST API - postman.

Вариант 5:

№	Категория	Запрос
1	pet предварительно выполнить POST /pet	POST /pet/{petId}/uploadImage
2		PUT /pet
3		GET /pet/findByStatus
4		GET /pet/{petId}
5		POST /pet/{petId}
6		DELETE /pet/{petId}

Выполнение работы

POST /pet

POST /pet/{petId}

Тест кейсы

1. **Название:** Добавление нового питомца в магазин

Начальные условия: Шаблон json-запроса с параметрами из спецификации.

Последовательность действий:

Создание и отправка POST/pet запроса с информацией о новом питомце.

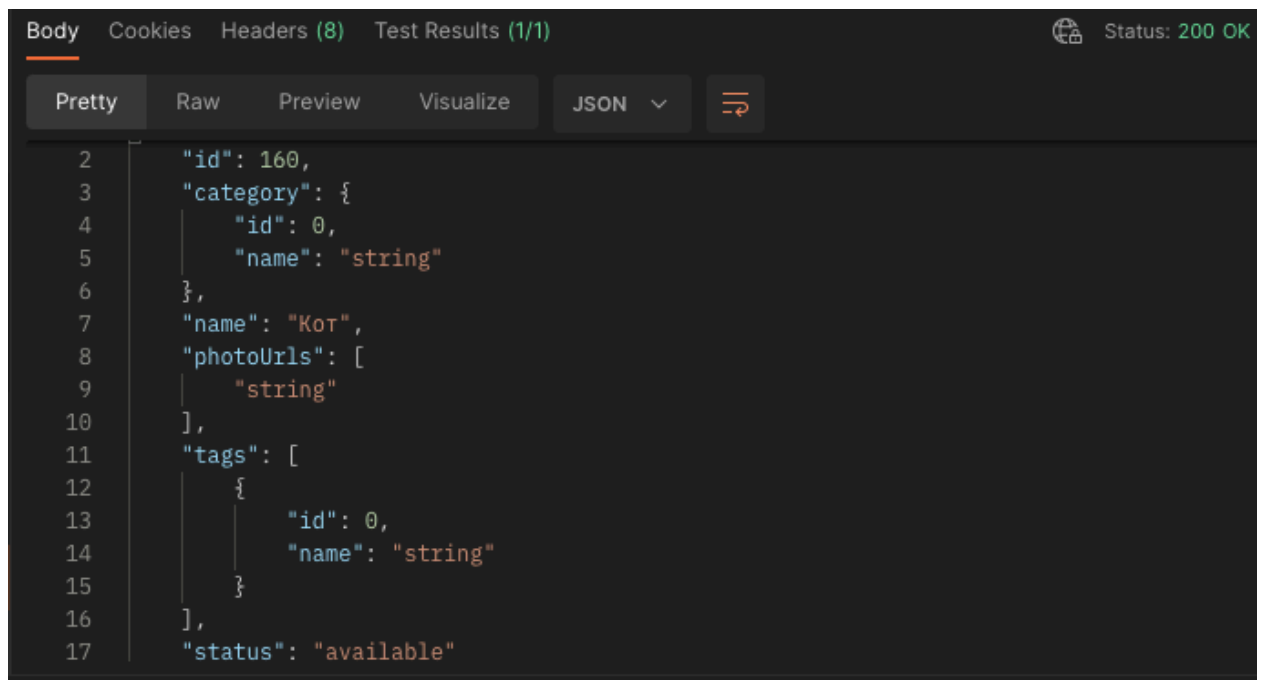
Ожидаемый результат:

- код ответа 200
- сообщение с json о новом питомце.

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "Кот",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Результат: код ответа 200

(задано id =160)

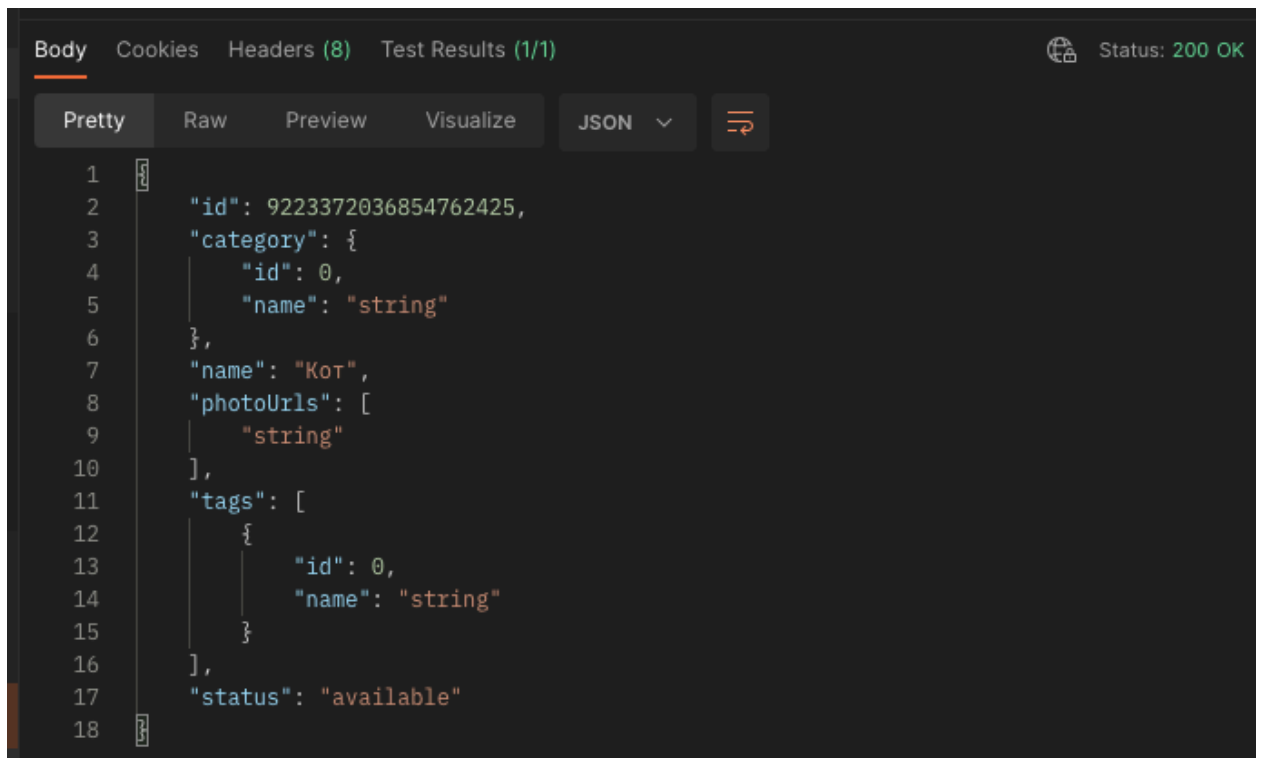


The screenshot shows a REST client interface with the following elements:

- Body** tab selected, showing a JSON response.
- Test Results (1/1)** tab showing a status of **200 OK**.
- JSON** tab selected, displaying the response body.
- The response body is a JSON object with the following structure:

```
{
  "id": 160,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "Кот",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

(задано id = 0)



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (8), and Test Results (1/1). The Body tab is active, displaying a JSON response in 'Pretty' format. The status is 200 OK. The JSON body is as follows:

```
1 {
2   "id": 9223372036854762425,
3   "category": {
4     "id": 0,
5     "name": "string"
6   },
7   "name": "Кот",
8   "photoUrls": [
9     "string"
10  ],
11  "tags": [
12    {
13      "id": 0,
14      "name": "string"
15    }
16  ],
17  "status": "available"
18 }
```

2. **Название:** Добавление нового питомца с некорректным именем в магазин

Начальные условия: Шаблон json-запроса с параметрами из спецификации.

Последовательность действий:

Создание и отправка POST/pet запроса с информацией о новом питомце.

Ожидаемый результат:

- 405 Invalid input
- сообщение об ошибке

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "Кот",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ]
}
```

```
],  
  "status": "available"  
}
```

Результат:

```
1  {}  
2    "code": 500,  
3    "type": "unknown",  
4    "message": "something bad happened"  
5  {}
```

3. **Название:** Добавление питомца с числовым значением поля имени

Начальные условия: Шаблон json-запроса с параметрами из спецификации

Последовательность действий:

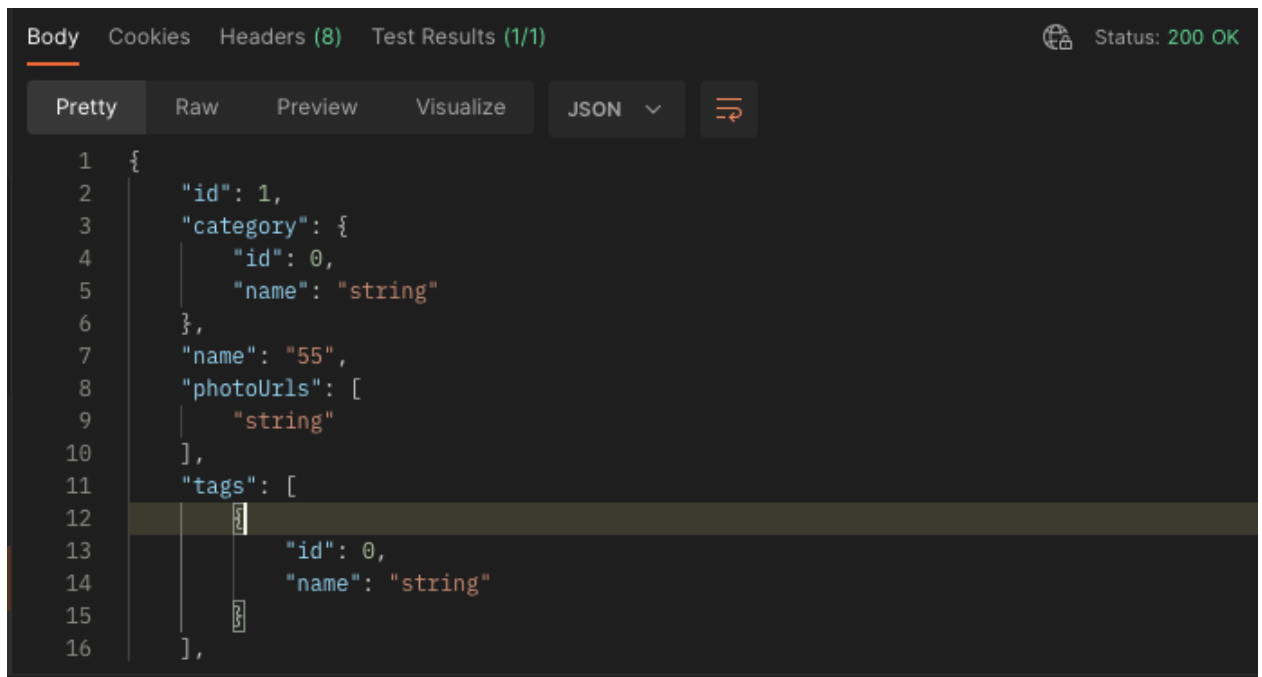
Создание и отправка POST/pet запроса с информацией о новом питомце

Ожидаемый результат:

- 405 Invalid input

```
{  
  "id": 1,  
  "category": {  
    "id": 0,  
    "name": "string"  
  },  
  "name": 55,  
  "photoUrls": [  
    "string"  
  ],  
  "tags": [  
    {  
      "id": 0,  
      "name": "string"  
    }  
  ],  
  "status": "available"  
}
```

Результат: код 200



```
Body Cookies Headers (8) Test Results (1/1) Status: 200 OK
Pretty Raw Preview Visualize JSON
1 {
2   "id": 1,
3   "category": {
4     "id": 0,
5     "name": "string"
6   },
7   "name": "55",
8   "photoUrls": [
9     "string"
10  ],
11  "tags": [
12    {
13      "id": 0,
14      "name": "string"
15    }
16  ],
```

4. **Название:** Добавление питомца с полем id, содержащим буквы

Последовательность действий:

1. Создание и отправка POST/pet запроса с информацией о новом питомце с некорректным значением id

Ожидаемый результат:

- код ответа 405 Invalid input
- сообщение об ошибке

```
{
  "id": "a",
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": 55,
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Результат (500 Server Error вместо 405 Invalid input):



The screenshot shows a REST client interface with the 'Body' tab selected. The status bar at the top right indicates 'Status: 500 Internal Server Error'. The response body is displayed in JSON format:

```
1 {
2   "code": 500,
3   "type": "unknown",
4   "message": "something bad happened"
5 }
```

5. Название: Обновление данных питомца с полем id

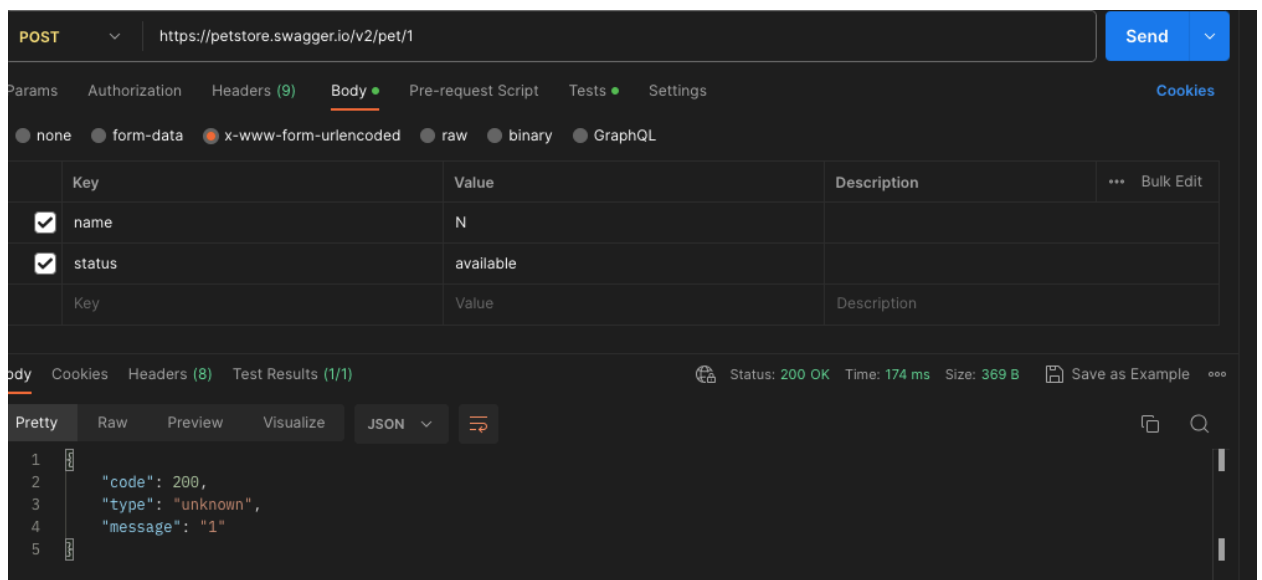
Последовательность действий:

1. Создание и отправка `POST/pet/{petId}` запроса с информацией о питомце с корректным значением `Name` и `status`

Ожидаемый результат:

- код ответа `200`

Обновление данных питомца



The screenshot shows a REST client interface with a `POST` request to `https://petstore.swagger.io/v2/pet/1`. The 'Body' tab is selected, and the request body is set to `x-www-form-urlencoded`. The request parameters are:

Key	Value	Description
<input checked="" type="checkbox"/> name	N	
<input checked="" type="checkbox"/> status	available	

The status bar at the top right indicates 'Status: 200 OK', 'Time: 174 ms', and 'Size: 369 B'. The response body is displayed in JSON format:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "1"
5 }
```

6. Название: Обновление данных питомца с полем id

Последовательность действий:

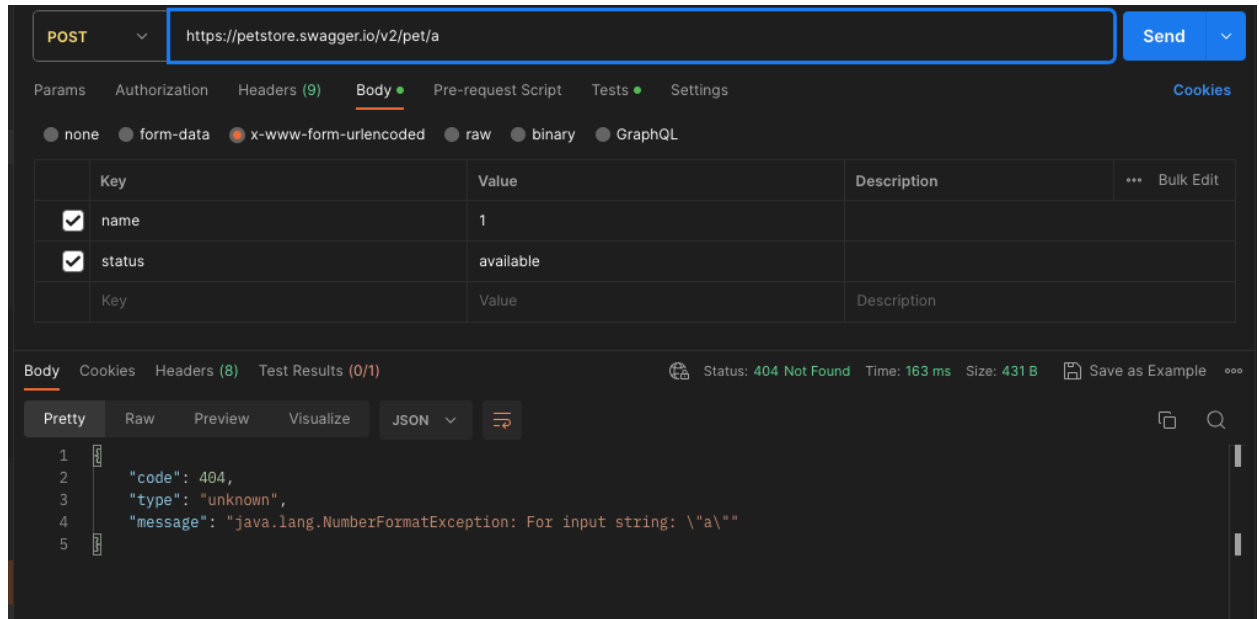
1. Создание и отправка `POST/pet/{petId}` запроса с информацией о питомце со строковым значением `petId`

Ожидаемый результат:

- код ответа `404`

Обновление данных питомца

Результат: код `404(Not found)`



7. Название: Обновление данных питомца с полем id

Последовательность действий:

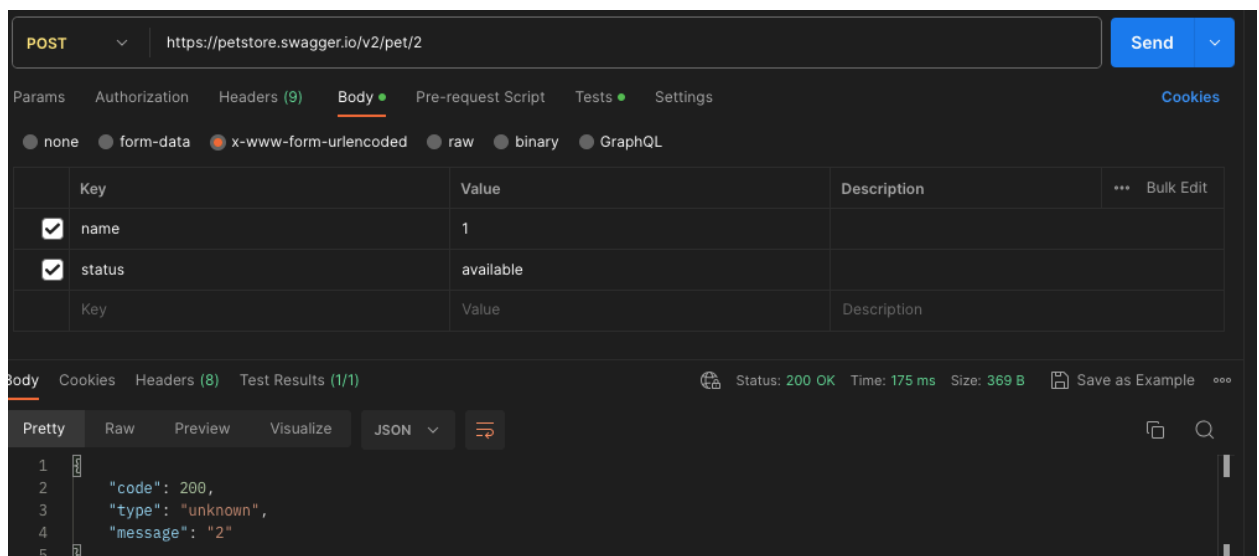
1. Создание и отправка POST/pet/{petId}запроса с информацией о питомце с числовым значением Name

Ожидаемый результат:

- код ответа 405

Обновление данных питомца

Результат: код 200



Результаты тестирования

Были выявлены следующие дефекты:

- При отправке `POST/pet` запроса с информацией о новом питомце, где информация вводится некорректная, выдается не код `405 (Invalid input)`, заявленный в спецификации, а `400 (Bad Request)`.
- При отправке `POST/pet` запроса с информацией о новом питомце с некорректным значением `id`, возвращается код ответа `500 (Server Error)`, вместо `405 (Invalid input)`.
- При отправке запроса с `id = 0` возникает переполнение.
- При отправке запроса с числовым полем имени.

Вывод

Во время выполнения лабораторной работы были получены навыки в проектировании тестовых сценариев для тестирования REST API, а также с инструментом тестирования REST API - `postman`.