API Testing

From the next swagger: <https://petstore3.swagger.io/>

12 test cases are performed for pets. (**Please, read the note at the end**)

**Case 1**: Validate service response when updating the data of a pet in the shop.

**Input data:**

{"id":7,"category":{"id":4,"name":"Brandon"},"name":"Lion 1","photoUrls":["url1","url2"],"tags":[{"id":1,"name":"tag1"},{"id":2,"name":"tag2"}],"status":"sold"}

**Pre-condition**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet)

The pet must exist.

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet](http://localhost:8080/api/v3/pet)

3. Select the PUT method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 200.

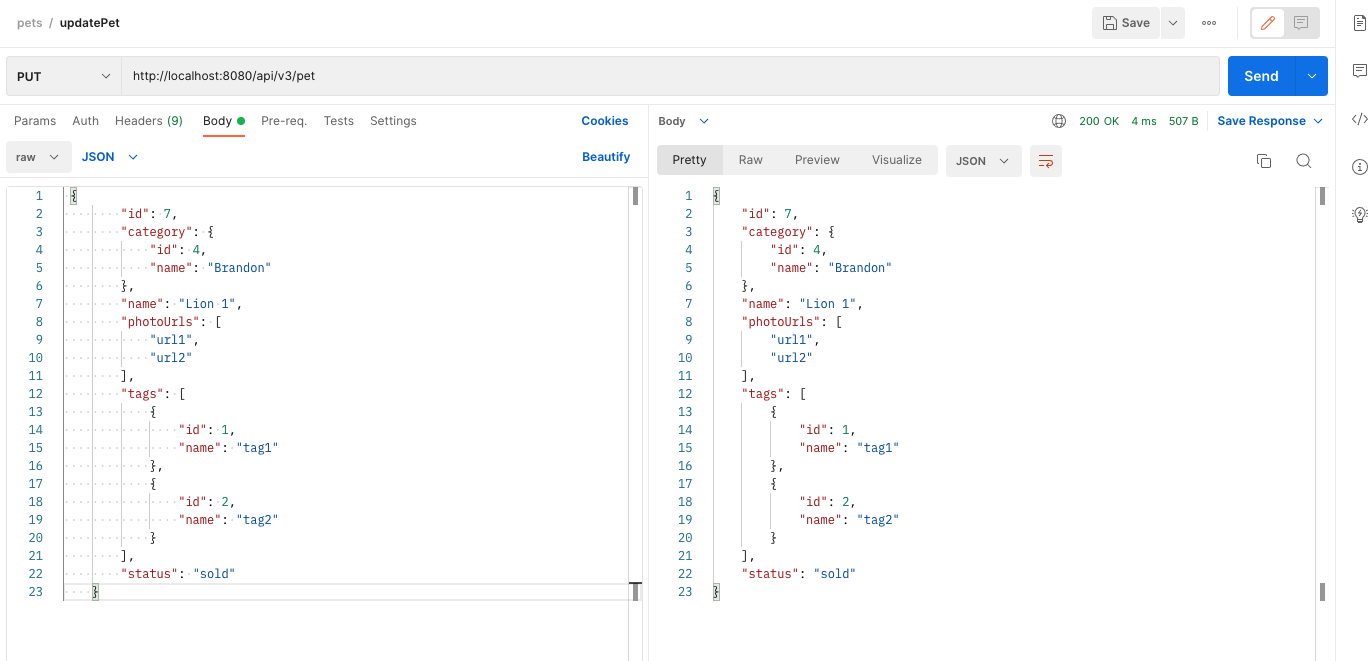
**Obtained result:**

Service responds status 200. OK

**Test status:**

Successful test.

**Evidence:**



**Case 2:** Validate service response when entering a new pet into the store.

**Pre-condition:**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet)

Add a pet that doesn't exist.

**Input data:**

{"id":170,"name":"Neron","category":{"id":1,"name":"Neron 1"},"photoUrls":["string"],"tags":[{"id":0,"name":"string"}],"status":"available"}

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet](http://localhost:8080/api/v3/pet)

3. Select the POST method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 200.

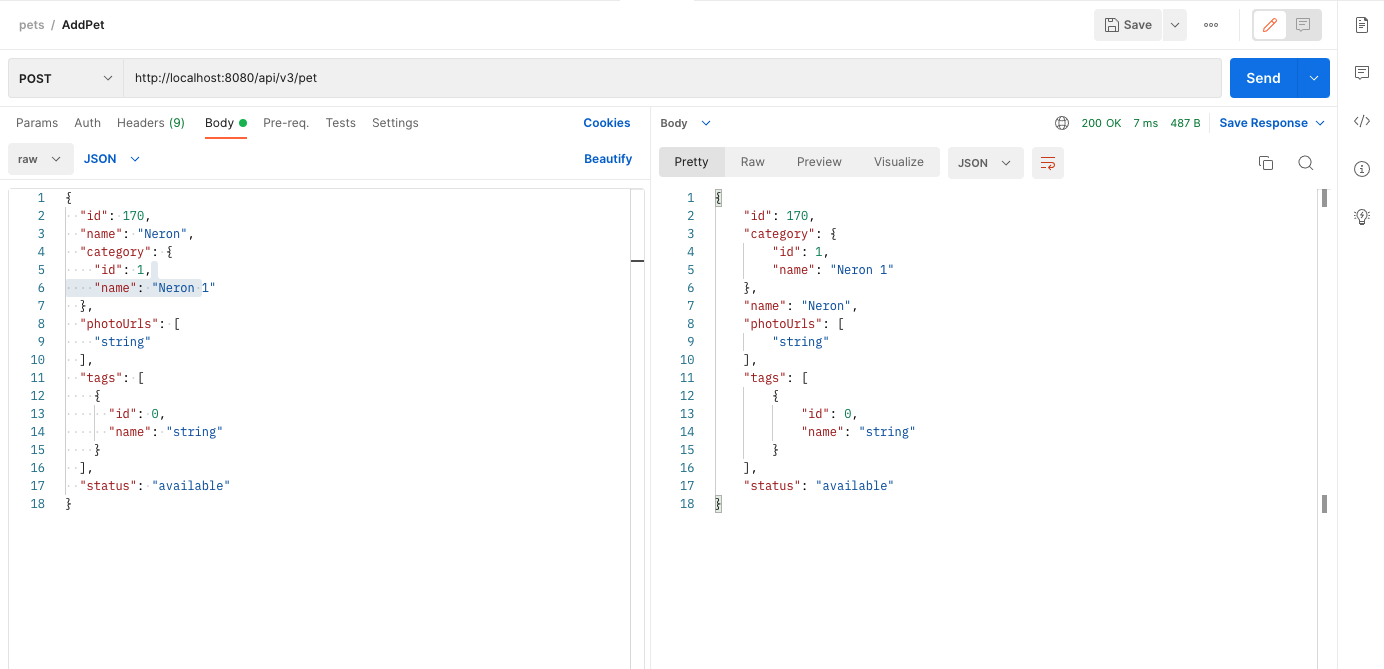
**Obtained result:**

Service responds status 200. OK

**Test status:**

Successful test.

**Evidence:**



**Case 3:** Validate service response when entering a new pet into the store with an existing ID.

**Pre-condition:**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet)

Add a pet with an existing ID.

**Input data:**

{"id":170,"name"capricho"category":{"id":1,"name":"Neron 1"},"photoUrls":["string"],"tags":[{"id":0,"name":"string"}],"status":"available"}

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet](http://localhost:8080/api/v3/pet)

3. Select the POST method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

El servicio responde status 4xx.

**Resultado Obtenido:**

El servicio responde status 200. **NOK**

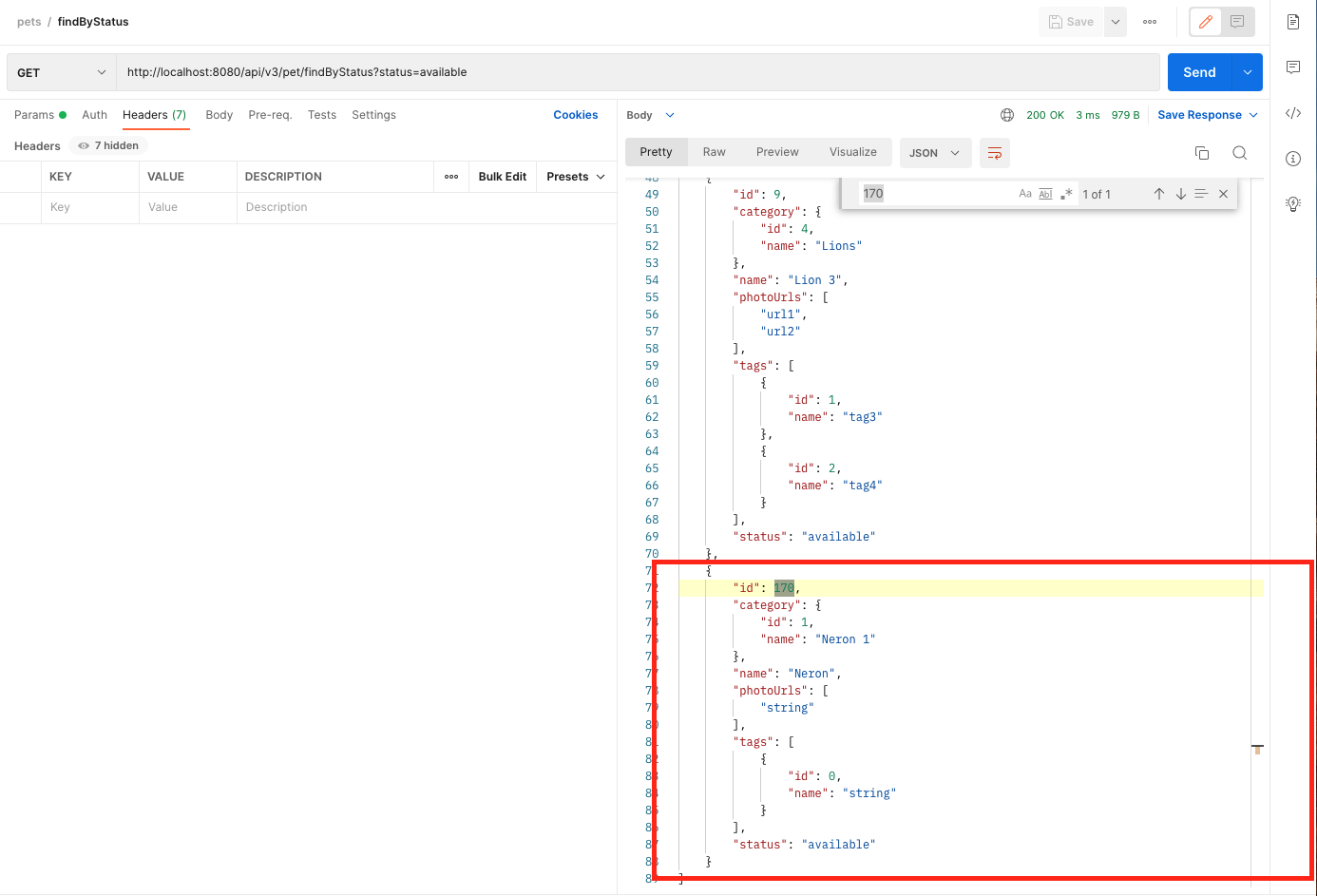
**Test status:**

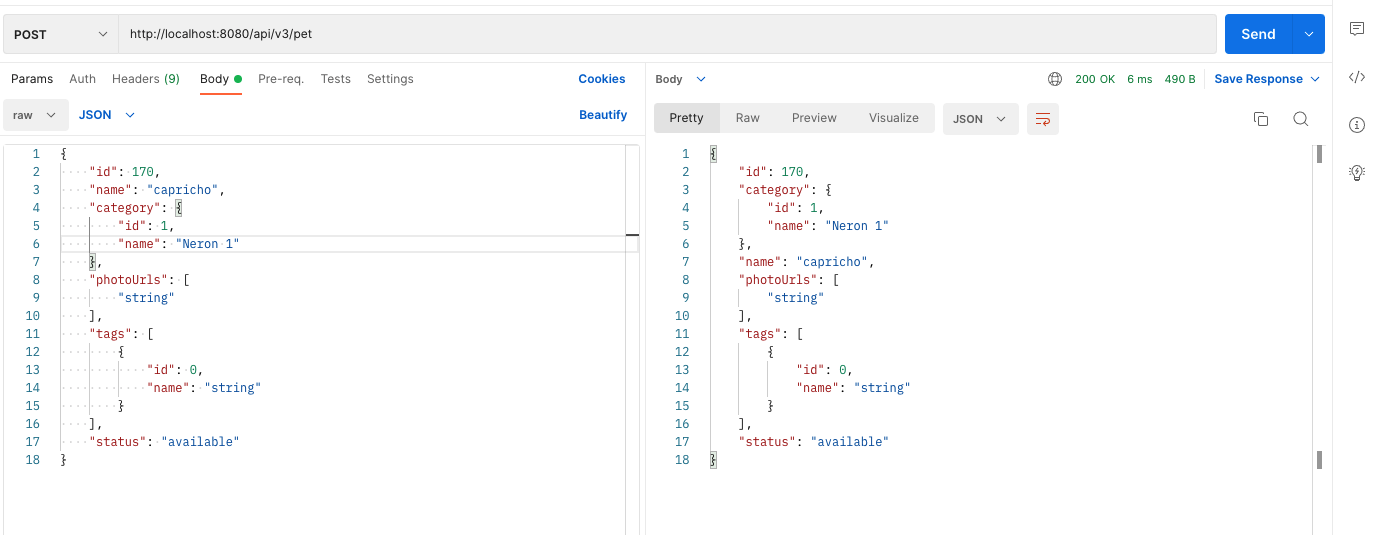
Successful test.

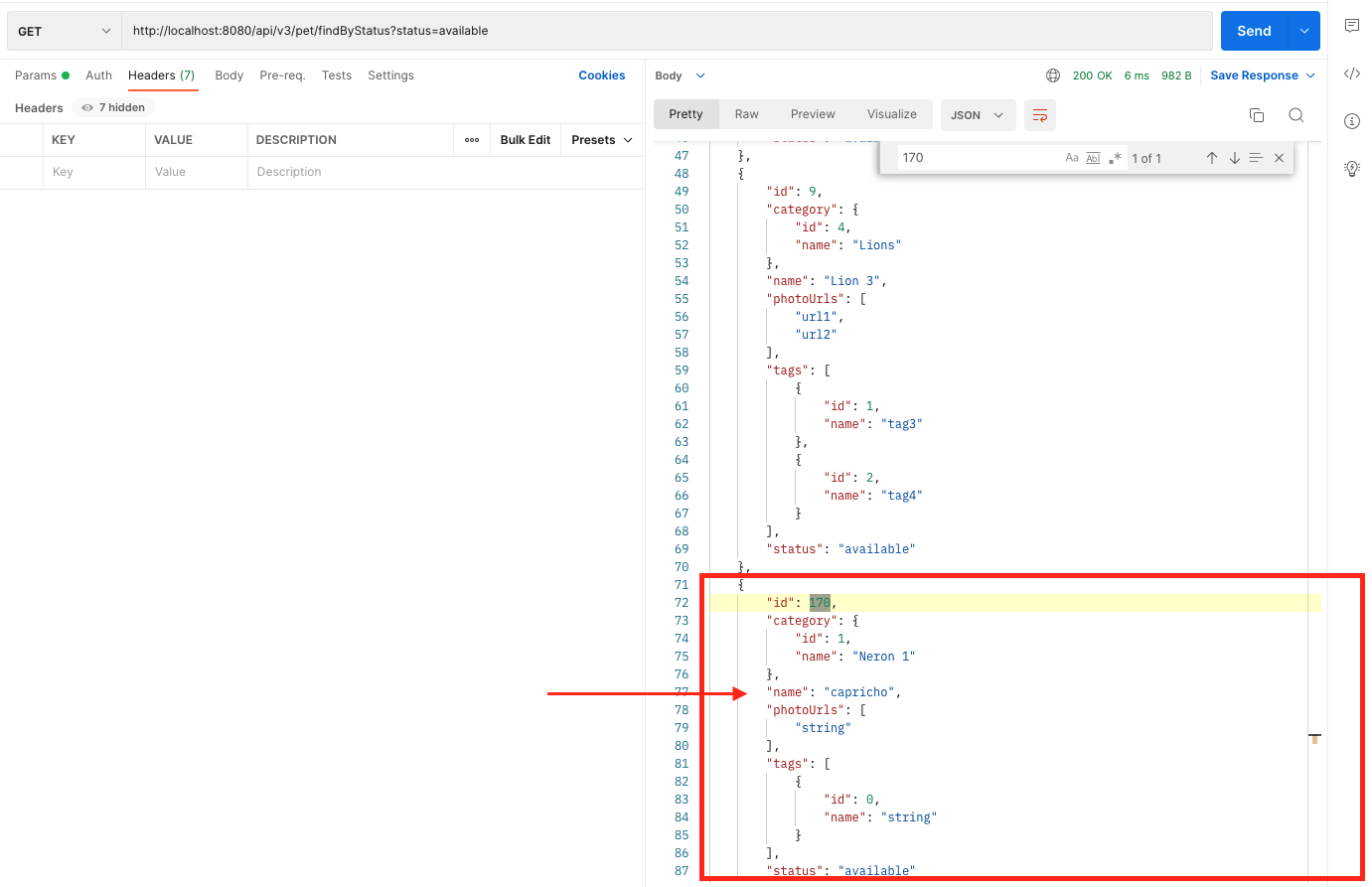
**Test status:**

Test failed; it should display an allusive message stating "ID in use". In this case, it is replacing the current data.

**Evidence:**







**Case 4:** Validate service response when querying pets by state.

**Pre-condition:**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet/findByStatus?status=available)

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet/findByStatus?status=available](http://localhost:8080/api/v3/pet/findByStatus?status=available)

3.- Select GET method

4.-Click send button

**Expected result:**

Service responds status 200.

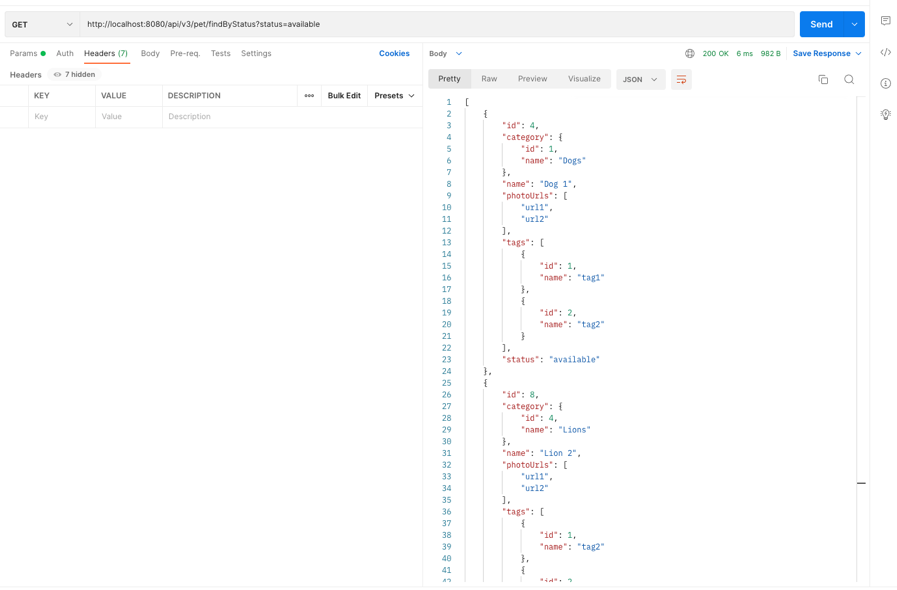
**Obtained result:**

Service responds status 200. OK

**Test status:**

Successful test. Pets are displayed with the status "available".

**Evidence:**



**Case 5:** Validate service response when querying pets by tags.

**Pre-condition:**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet/findByTags?tags=tag1)

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet/findByTags?tags=tag1](http://localhost:8080/api/v3/pet/findByTags?tags=tag1)

3.- Select GET method

4.-Click send button

**Expected result:**

Service responds status 200.

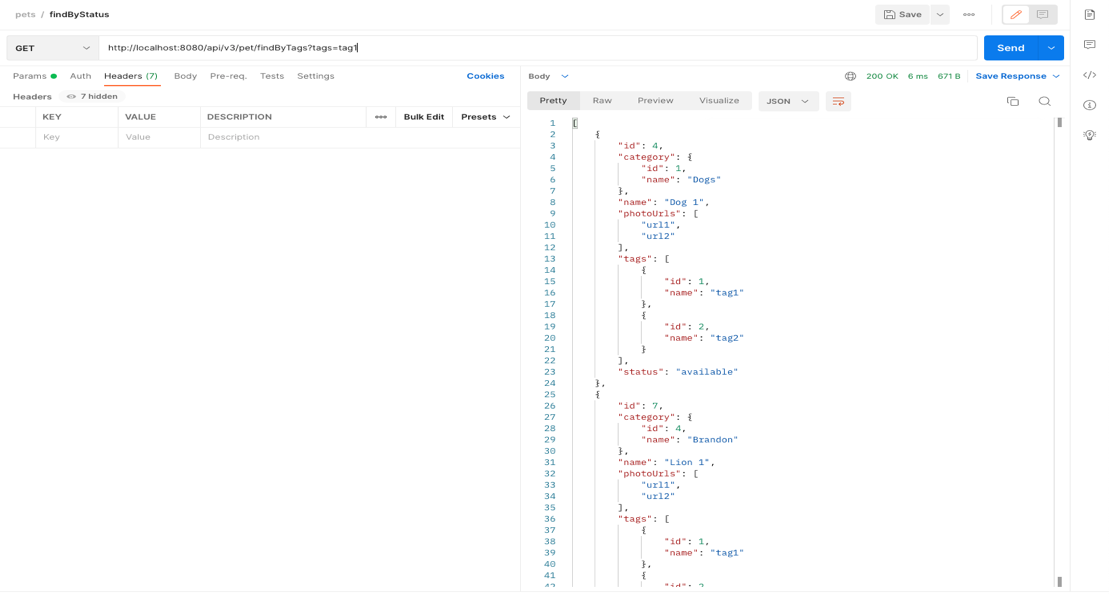
**Obtained result:**

Service responds status 200. OK

**Test status:**

Successful test. Pets are displayed with tags.

**Evidence:**



**Case 6:** Validate service response when querying pets for a tag that doesn't exist.

**Pre-condition:**

URL <http://localhost:8080>

It mustn’t exist that tag.

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet/findByTags?tags=tag1](http://localhost:8080/api/v3/pet/findByTags?tags=tag1)231

3.- Select GET method

4.-Click send button

**Expected result:**

Service responds status 404.

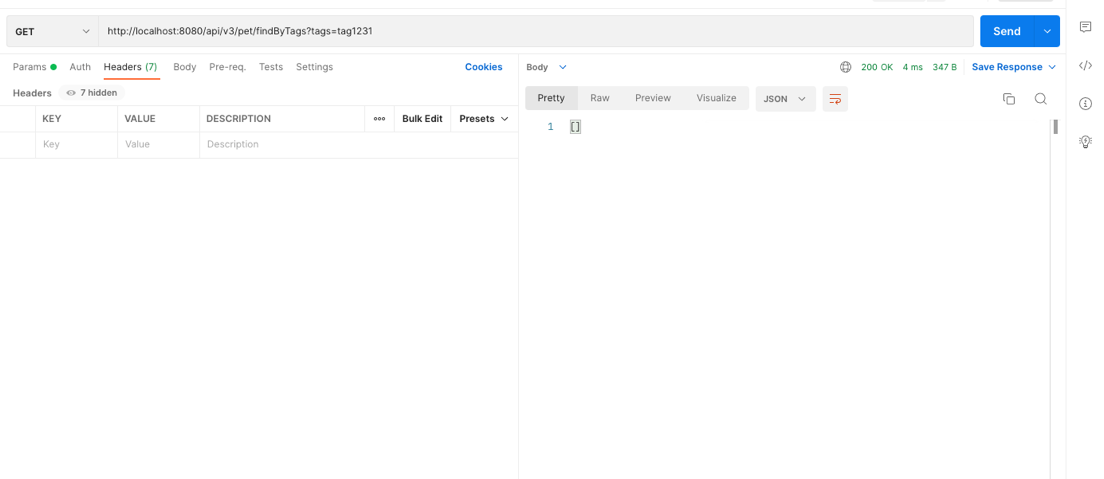
**Obtained result:**

Service responds status 200 **NOK**

**Test status:**

Test failed; it should display an allusive message stating: pet not found.

**Evidence:**



**Case 7:** Validate service response when querying pets by ID.

**Pre-condition:**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet/12)

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet](http://localhost:8080/api/v3/pet)/12

3. Select GET method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 200.

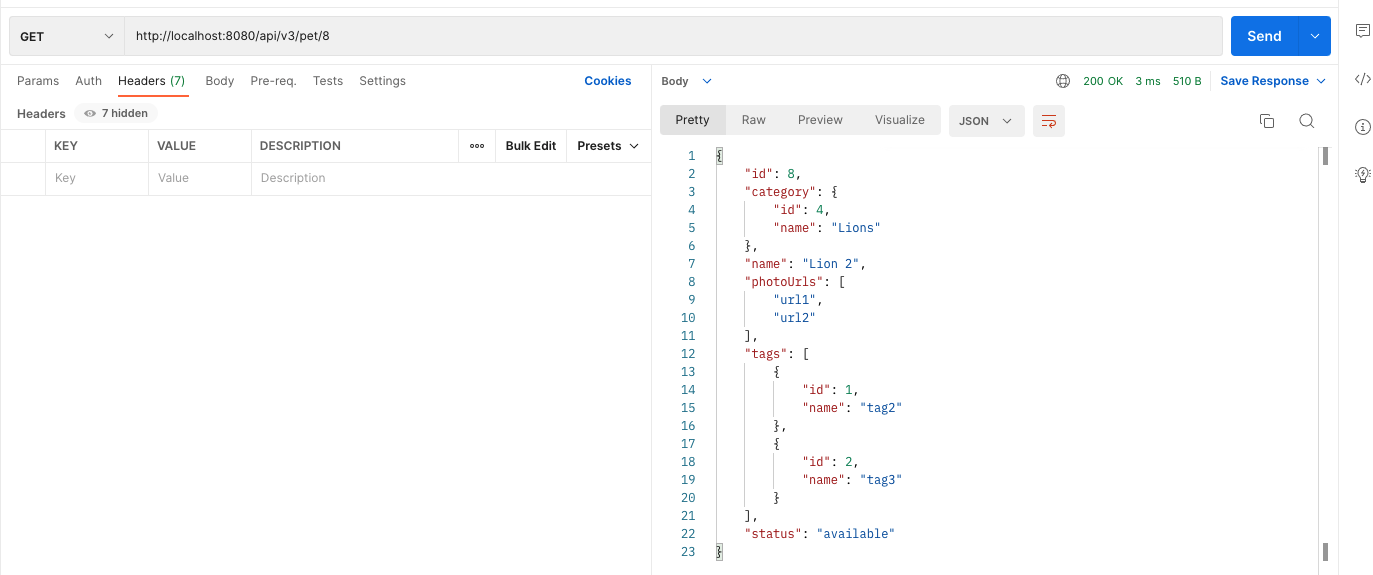
**Obtained result:**

Service responds status 200. OK

**Test status:**

Successful test.

**Evidence:**

****

**Case 8:** Validate service response when querying pets entering an ID that doesn't exist.

**Pre-condition:**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet/12)

ID mustn't exist.

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet](http://localhost:8080/api/v3/pet)/111222

3. Select GET method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 404.

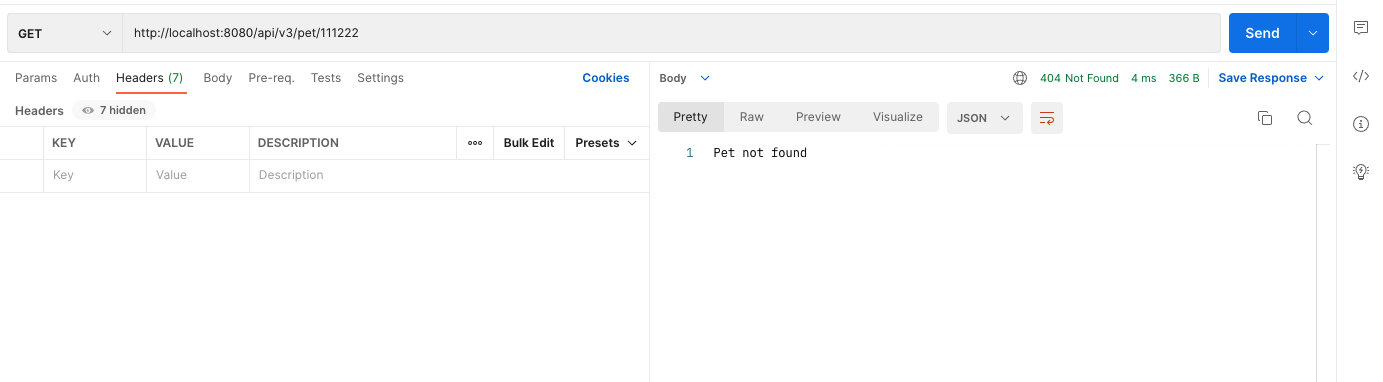
**Obtained result:**

Service responds status 404. OK

**Test status:**

Successful test. Display message: “Pet not found”

**Evidencia:**

****

**Case 9:** Validate service response when querying pets entering an incorrect ID.

**Pre-condition:**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet/hola)

ID must be in string format.

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet](http://localhost:8080/api/v3/pet)/hola

3. Select GET method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 400.

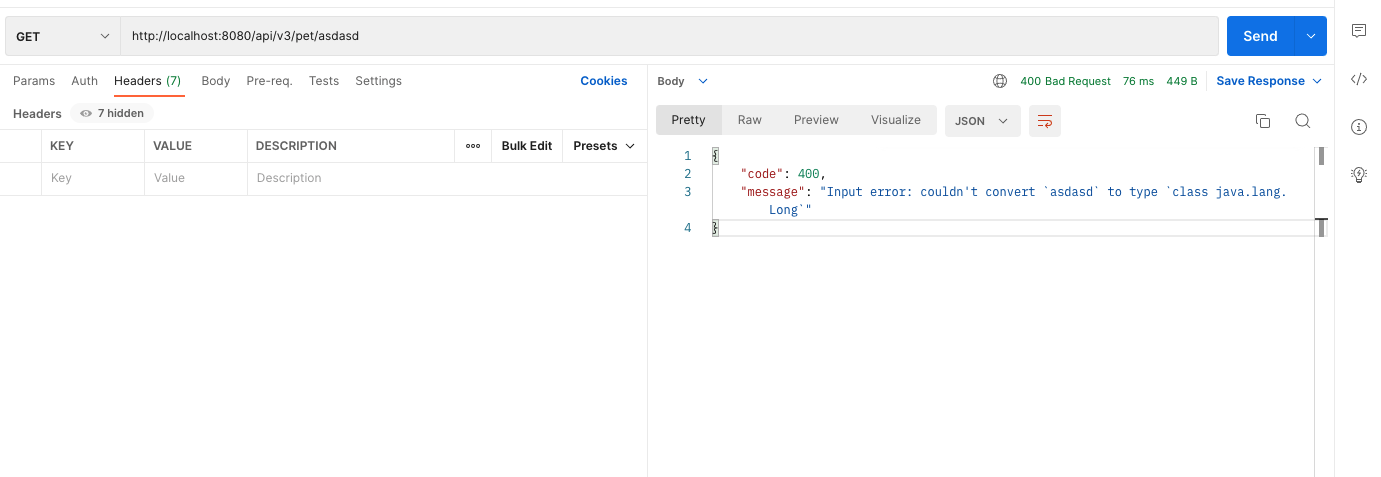
**Obtained result:**

Service responds status 400. OK

**Test status:**

Successful test. Display message: "Input error: couldn't convert `hola` to type `class java.lang.Long`"

**Evidence:**

****

**Case 10**: Validate service response when updating the data of a pet by sending the DNI in String format.

**Input data:**

{"id":”s”,"category":{"id":4,"name":"Brandon"},"name":"Lion 1","photoUrls":["url1","url2"],"tags":[{"id":1,"name":"tag1"},{"id":2,"name":"tag2"}],"status":"sold"}

**Pre-condition**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet)

Send ID in string format.

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet](http://localhost:8080/api/v3/pet)

3. Select the PUT method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 400.

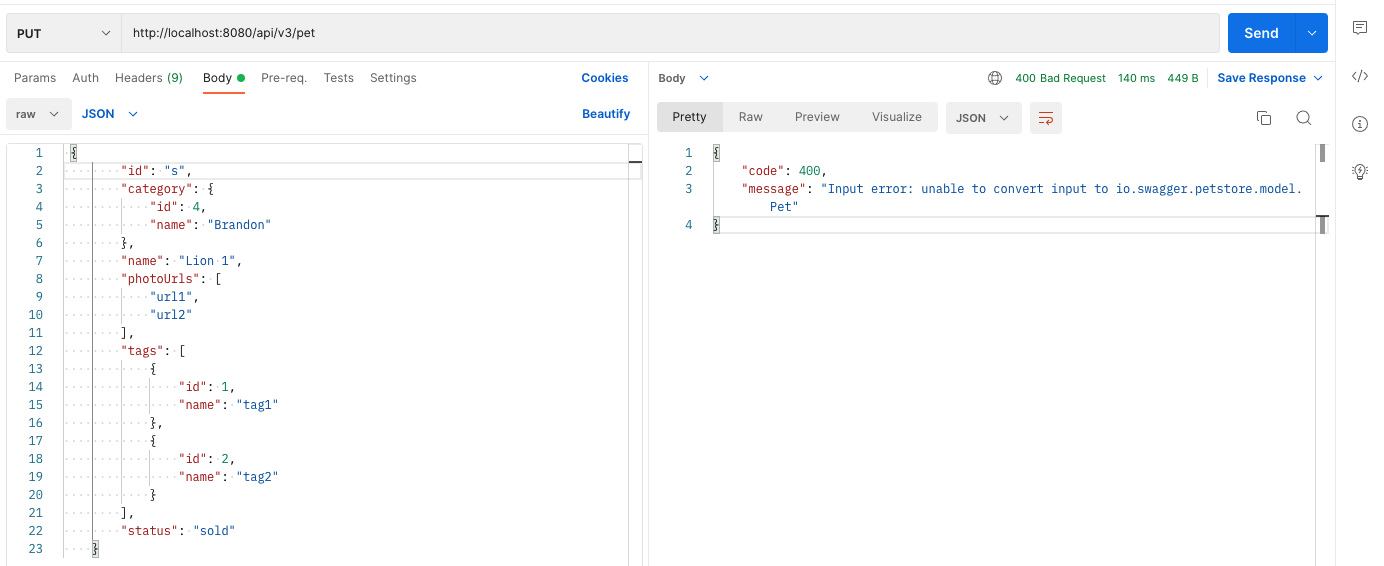
**Obtained result:**

Service responds status 400. OK

**Test status:**

Successful test. Display message "Input error: unable to convert input to io.swagger.petstore.model.Pet"

**Evidence:**



**Case 11**: Validate service response when deleting a pet.

**Pre-condition**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet/4)

The pet must exist.

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet/4](http://localhost:8080/api/v3/pet/4)

3.- Select DELETE method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 200.

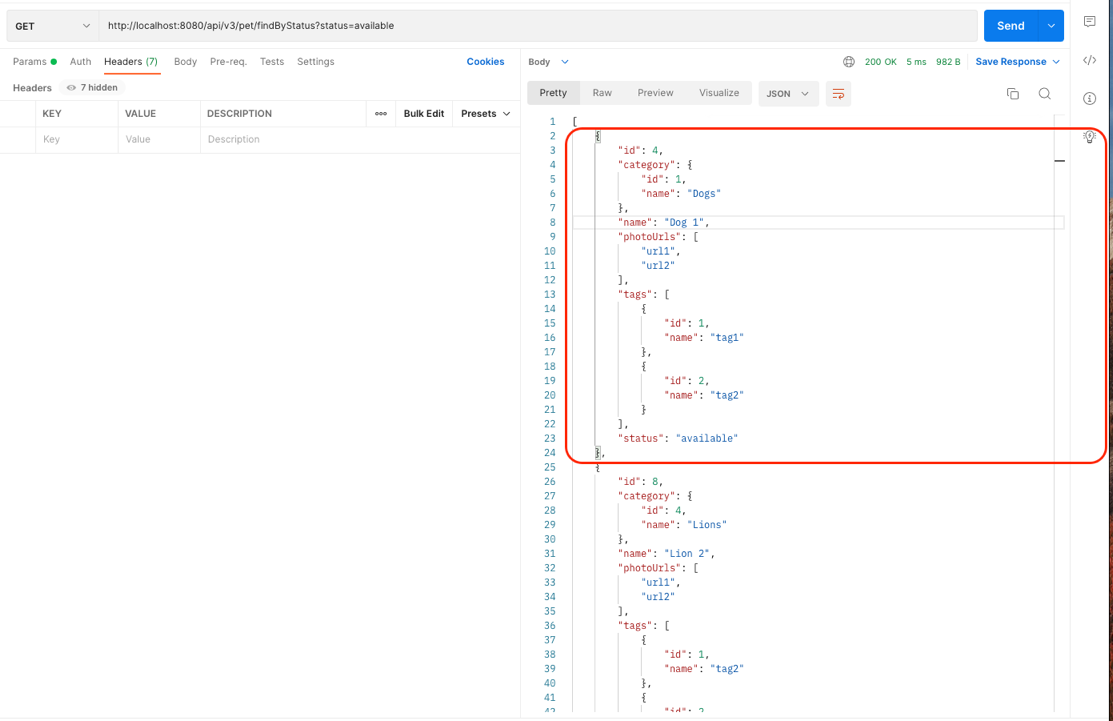
**Obtained result:**

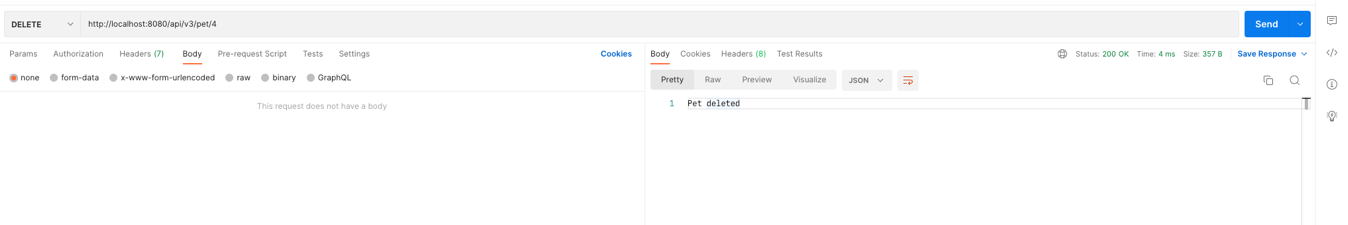
Service responds status 200. OK

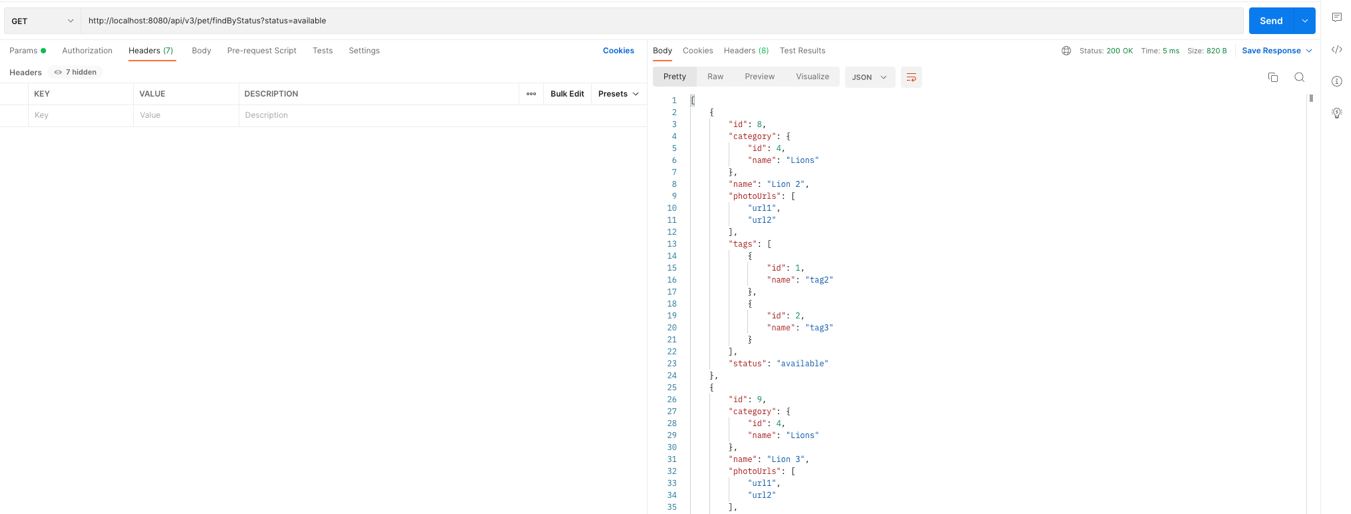
**Test status:**

Successful test. Pet is removed.

**Evidence:**







**Case 12**: Validate service response when deleting a pet that doesn't exist.

**Pre-condition**

URL [http://localhost:8080](http://localhost:8080/api/v3/pet/4)

The pet mustn’t exist.

**Steps:**

1. Enter postman

2. Add endpoint: [/api/v3/pet/4](http://localhost:8080/api/v3/pet/4)1234124124124

3.- Select DELETE method

4. Introduce input data in body request.

5. Click send button

**Expected result:**

Service responds status 404.

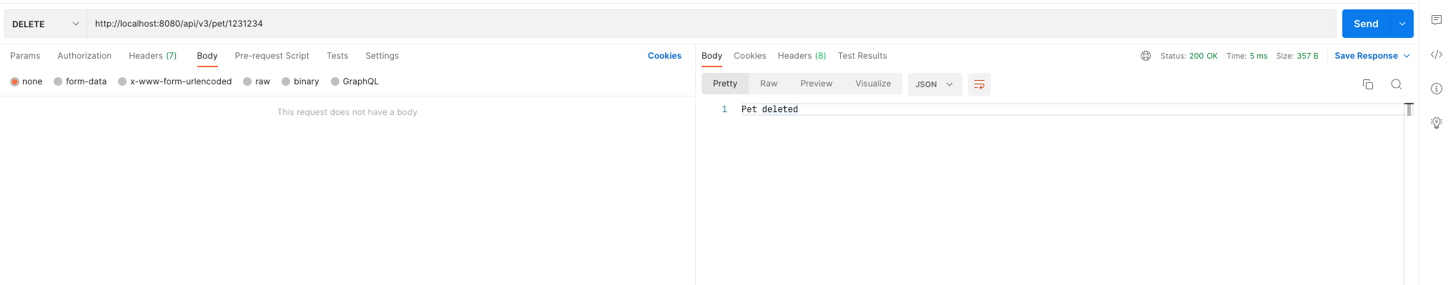
**Obtained result:**

Service responds status 200 **NOK**

**Test status:**

Fail test. It’s displaying a message: Pet deleted with code 200, and it should display an allusive message stating: pet not found.

**Evidence:**



**NOTE:** Test cases **3, 6, and 9,** were not automated, because in real life, first, we must fix these issues and then add them to the suite.