

This Postman collection demonstrates CRUD operations using the Swagger Petstore API. Each request contains valid parameters, headers, and example responses.

GET (Find pet by status)

The screenshot displays the Postman application interface. On the left sidebar, the 'Test Workspace' is visible, containing a collection named 'PetStore'. Under 'PetStore', there are several folders: 'GET', 'POST', 'PUT', and 'DELETE'. The 'GET' folder is expanded, showing two requests: 'GET Find pets by status' (selected) and 'GET Find a pet by id'. The main panel shows the details of the 'GET Find pets by status' request. The URL is `https://petstore.swagger.io/v2/pet/findByStatus?status=pending`. The 'Query Params' section shows a table with two entries: 'Key' and 'status', both with values. The 'Body' tab is selected, showing a JSON response. The response status is '200 OK' with a response time of 672 ms and a size of 17.31 KB. The JSON response is as follows:

```
1 [
2   {
3     "id": 324,
4     "category": {
5       "id": 12,
6       "name": "Fish"
7     },
8     "name": "Chelsey",
9     "photoUrls": [
10      "https://loremflickr.com/1218/3955/animals?lock=155564964981491"
11    ],
12     "tags": [
```

The bottom of the screen shows the Windows taskbar with various application icons and the system clock indicating 18:39 on 08.10.2025.

POST (Add a new pet)

The screenshot displays the Postman interface with a REST client request configured for a POST method to the endpoint `[[url]] /pet`. The request body is set to raw JSON, containing the following data:

```
1 {
2   "id": 1804,
3   "category": {
4     "id": 0,
5     "name": "mause"
6   },
7   "name": "Milky",
8   "photoUrls": [
9     "photo"
10  ],
11  "tags": [
12    {

```

The response section shows a successful status of **200 OK** with a response time of 199 ms and a body size of 464 B. The response body is displayed in JSON format, mirroring the request structure with double quotes escaped:

```
1 {
2   "id": 1804,
3   "category": {
4     "id": 0,
5     "name": "mause"
6   },
7   "name": "Milky",
8   "photoUrls": [
9     "photo"
10  ],
11  "tags": [
12    {

```

The interface includes a sidebar with collections (PetStore, GET, POST, PUT, DELETE) and environments. The bottom status bar shows system information including the date 09.10.2025 and time 9:45.

PUT (Update a pet)

The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows a 'Test Workspace' with a 'PetStore' collection containing 'GET', 'POST', 'PUT', and 'DELETE' folders. The 'PUT' folder is selected, showing a request 'PUT Update a pet'. The main panel shows the request details for 'PUT /pet'. The 'Body' tab is active, displaying a JSON payload in raw format. The response is shown as '200 OK' with a status of '200 OK', a response time of '686 ms', and a size of '465 B'. The response body is displayed in JSON format.

Request Details:

- Method: PUT
- URL: `{{url}}/pet`
- Body Type: raw
- Body Content:

```
2  "id": 1804,  
3  "category": {  
4    "id": 0,  
5    "name": "mause"  
6  },  
7  "name": "Cheese",  
8  "photoUrls": [  
9    "photo"  
10 ],  
11 "tags": [  
12   {  
13     "id": 0,
```

Response Details:

- Status: 200 OK
- Response Time: 686 ms
- Size: 465 B
- Body Type: JSON
- Body Content:

```
1  {  
2    "id": 1804,  
3    "category": {  
4      "id": 0,  
5      "name": "mause"  
6    },  
7    "name": "Cheese",  
8    "photoUrls": [  
9      "photo"  
10   ],  
11   "tags": [  
12     {
```

DELETE (Delete pet)

The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows a 'Test Workspace' with a 'PetStore' collection containing several endpoints, including 'DELETE Delete pet'. The main panel shows the details of the selected endpoint: a DELETE request to 'http://{{url}}/pet/1804'. The 'Headers' tab is active, showing a list of headers including 'Postman-Token', 'Host', 'User-Agent', 'Accept', 'Accept-Encoding', 'Connection', and 'api_key'. The 'Body' tab is also visible, showing a JSON response:

```
{  "code": 200,  "type": "unknown",  "message": "1804"}
```

. The status bar at the bottom indicates a successful response with '200 OK', '791 ms', and '372 B'.

Test Workspace

New Import

Overview GET Find pets by s GET Find a pet by POST Add a new p PUT Update a pet ReqresCollectio DEL Delete pet + No environment

PetStore / DELETE / Delete pet

DELETE {{url}}/pet/1804

Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

Key	Value	Description
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.45.0	
<input checked="" type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> api_key	special-key	

Body Cookies Headers (8) Test Results

200 OK • 791 ms • 372 B • Save Response

JSON Preview Visualize

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "1804"
5 }
```

Postbot Runner Start Proxy Cookies Vault Trash

9:59 09.10.2025

Expected response (200 OK):

The image shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows a 'Test Workspace' with a collection named 'PetStore' containing several endpoints. The main panel displays a POST request to 'PetStore / POST / Add a new pet'. The request body is a JSON object. The response is a 200 OK status with a response time of 865 ms and a body size of 464 B. The response body is a JSON object representing a pet.

Request:

```
POST {{url}}/pet
```

Scripts:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
```

Response:

```
1 {
2   "id": 1804,
3   "category": {
4     "id": 0,
5     "name": "mause"
6   },
7   "name": "Milky",
8   "photoUrls": [
9     "photo"
10  ],
11  "tags": [
12    {
13      "id": 0,
14      "name": "string"
15    }
16  ],
17  "status": "available"
18 }
```