

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 shows the Postman application interface with the following details:

- Left Sidebar:** Test Workspace, Collections, Environments, Flows, History.
- Top Bar:** Home, Workspaces, API Network, Search Postman, Invite, Upgrade, Save, Share, No environment.
- Middle Section:**
 - Collection:** PetStore / GET / Find pets by status
 - Method:** GET, URL: https://petstore.swagger.io/v2/pet/findByStatus?status=pending
 - Params:** status = pending
 - Body:** JSON (Preview shows a single pet object with id 324, category Fish, name Chelsey, photo URL, and tags).
 - Headers:** (8 items)
 - Tests:** (Empty)
 - Settings:** (Empty)
- Bottom Status Bar:** Online, Find and replace, Console, Postbot, Runner, Start Proxy, Cookies, Vault, Trash, 18:39, ENG, 08.10.2025.

POST (Add a new pet)

The screenshot shows the Postman application interface. The left sidebar displays a 'Test Workspace' with collections like 'PetStore', environments, flows, and history. The main workspace shows a 'PetStore / POST / Add a new pet' request. The 'Body' tab is selected, showing a raw JSON payload:

```
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 ]
```

The response section shows a successful 200 OK status with a response time of 199 ms and a body size of 464 B. The response body is identical to the request body.

At the bottom, the taskbar includes icons for Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and system notifications. The system tray shows the date and time as 09.10.2025 at 9:45, along with battery and network status.

PUT (Update a pet)

The screenshot shows the Postman application interface. The left sidebar displays a 'Test Workspace' with a 'PetStore' collection containing various requests like GET Find pets by status, GET Find a pet by id, POST Add a new pet, and PUT Update a pet. The 'PUT Update a pet' request is currently selected. The main panel shows the request details for 'PetStore / PUT / Update a pet'. The method is set to 'PUT' with the URL template '{url}/pet'. The 'Body' tab is active, showing a raw JSON payload:

```
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,
14     }
15 ]
```

Below the body, the response section shows a successful '200 OK' status with a response time of 686 ms and a size of 465 B. The response body is identical to the one in the body panel. The bottom navigation bar includes icons for Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and a help icon.

DELETE (Delete pet)

The screenshot shows the Postman application interface. The left sidebar displays a 'Test Workspace' with various collections, environments, flows, and history. The main workspace shows a 'PetStore / DELETE / Delete pet' request. The 'Headers (8)' tab is selected, listing the following headers:

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

The 'Body' tab shows a JSON response:

```
{ } JSON ▾
```

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

The status bar at the bottom shows the Windows taskbar with icons for File, Home, Start, Task View, and Edge browser. The system tray shows battery level, signal strength, and the date/time (09.10.2025).

Expected response (200 OK):

The screenshot shows the Postman application interface. The left sidebar displays a 'Test Workspace' with a 'PetStore' collection containing various HTTP methods like GET, POST, PUT, and DELETE. The main workspace shows a 'PetStore / POST / Add a new pet' request. The 'Body' tab is selected, showing a JSON response with status 200 OK. The response body is a JSON object representing a new pet:

```
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 }
```