

ШАГИ 5

1

Отправить POST запрос <https://petstore.swagger.io/v2/pet?status=sold>

Запрос успешно отправлен на сервер, возвращается тело ответа

2

Проверить код состояния

HTTP Status: 200 OK

3

Проверить тело ответа от сервера






Тело ответа в формате JSON (сформированный объект JSON) во: от сервера и будет иметь следующий вид:  
{  
 "id": 10,  
 "category": {  
 "id": 1,  
 "name": "animal"  
 },  
 "name": "Bulldog",  
 "photoUrls": [  
 "<https://site.com/bulldog/photo>"  
 ],  
 "tags": [  
 {  
 "id": 1,  
 "name": "#dog"  
 }  
 ],  
 "status": "available"  
}

https://team-uzez.testit.software/viewer/tests

1/6

		<pre>} Каждое поле JSON объекта соответствует аргументу метода GET</pre>
4	Проверить структуру тела ответа типов данных	<p>Схема JSON отображена корректно имена и типы полей соответствуют ожидаемым, включая вложенные объекты:</p> <pre>"category": {   "id": 1,   "name": "animal" } </pre> <p>вложенная структура данных, содержащая объекты и массивы:</p> <pre>"tags": [   {     "id": 1,     "name": "#dog"   } ] </pre> <p>значения полей соответствуют ожидаемым значениям из тестов</p> <p><b>"id": 10 - number</b> <b>"category": {"name": "animal"} - object</b> <b>"name": "Bulldog" - string</b> <b>"photoUrls": [ "https://site.com/bulldog/photo" ] - array</b> <b>"tags": [ { "name": "#dog" } ] - array with objects</b> <b>"status": "available" - string</b></p>
5	Проверить заголовки ответа	<p>access-control-allow-headers: Content-Type,api_key,Authorization access-control-allow-methods: GET,POST,DELETE,PUT access-control-allow-origin: * content-type: application/json date: server: Jetty(9.2.9.v20150224)</p>

113 GET 200 OK; Find Pet by ID

Тип	Автоматизация	Автор	Длительность
 Тестовый сценарий	 Ручной	 Р Руслан Гадельшин	00:10:00
Приоритет	Статус	Секция	Дата создания
 Средний	 Не готов	Pet	09.04.2025
Дата изменения			
09.04.2025			

ПРЕДУСЛОВИЯ ТЕСТА 2






1	Headers request: Content-Type: application/json	
2	Test Data: Pet's ID: 10	

ШАГИ 4

1	Отправить GET запрос: <a href="https://petstore.swagger.io/v2/pet/10">https://petstore.swagger.io/v2/pet/10</a>	Запрос успешно отправлен на сервер
2	Проверить код состояния	200 OK
3	Проверить тело ответа от сервера	<p>Тело ответа в формате JSON возвращается от сервера и будет следующий вид:</p> <pre>"id": 10, "category": {   "id": 1,   "name": "animal" }, "name": "Bulldog", "photoUrls": [   "https://site.com/bulldog/photo" ], </pre>

		<pre>"tags": [ {   "id": 1,   "name": "#dog" } ], "status": "available" }</pre>																											
4	Проверить структуру типов данных тела ответа	<table><tr><th>Поле</th><th>Тип данных</th><th>Ожидаемое Значение</th></tr><tr><td>id</td><td>integer</td><td>10</td></tr><tr><td>category.id</td><td>integer</td><td>1</td></tr><tr><td>category.name</td><td>string</td><td>"animal"</td></tr><tr><td>name</td><td>string</td><td>"Bulldog"</td></tr><tr><td>photoUrls</td><td>array</td><td>["https://site.com/bulldog/photo"]</td></tr><tr><td>tags[0].id</td><td>integer</td><td>1</td></tr><tr><td>tags[0].name</td><td>string</td><td>"#dog"</td></tr><tr><td>status</td><td>string</td><td>"available"</td></tr></table>	Поле	Тип данных	Ожидаемое Значение	id	integer	10	category.id	integer	1	category.name	string	"animal"	name	string	"Bulldog"	photoUrls	array	["https://site.com/bulldog/photo"]	tags[0].id	integer	1	tags[0].name	string	"#dog"	status	string	"available"
Поле	Тип данных	Ожидаемое Значение																											
id	integer	10																											
category.id	integer	1																											
category.name	string	"animal"																											
name	string	"Bulldog"																											
photoUrls	array	["https://site.com/bulldog/photo"]																											
tags[0].id	integer	1																											
tags[0].name	string	"#dog"																											
status	string	"available"																											

114 GET 200 OK; Find pet by Status: sold

Тип	Автоматизация	Автор	Длительность
 Тестовый сценарий	 Ручной	 Р <a href="#">Руслан Гадельшин</a>	00:10:00
Приоритет	Статус	Секция	Дата создания
 Средний	 Не готов	<b>Pet</b>	09.04.2025
Дата изменения			
09.04.2025			

ПРЕДУСЛОВИЯ ТЕСТА 2

1	Headers request: Content-Type: application/json	
2	Test Data: Pet's status: sold Параметры создаются вручную и передаются вручную во вкладке Query Params  Test Paramaters: status = sold status = abaliable status = pending	

ШАГИ 4

1	Отправить GET запрос: <a href="https://petstore.swagger.io/v2/pet/findByStatus?status=available&amp;status=pending&amp;status=sold">https://petstore.swagger.io/v2/pet/findByStatus?status=available&amp;status=pending&amp;status=sold</a>	Запрос успешно отправляется на сервер
2	Проверить код состояния	200 OK
3	Проверить тело ответа от сервера	Тело ответа в формате JSON возвращается от сервера и показывает корректные значения о доступных питомцах, выбранных по стат
4	Проверить структуру типов данных тела ответа.	Ответ содержит корректный JSON-объект с полями: id, name, st; Вложенные структуры соответствуют отправленным данным.

115 PUT 200 OK; Update an existing pet

Тип	Автоматизация	Автор	Длительность
 Тестовый сценарий	 Ручной	 Р <a href="#">Руслан Гадельшин</a>	00:10:00
Приоритет	Статус	Секция	Дата создания
 Средний	 Не готов	Pet	09.04.2025
Дата изменения 09.04.2025			

ПРЕДУСЛОВИЯ ТЕСТА 2






1	Headers request: Content-Type: application/json	
2	Test Data: Параметры в случае PUT передаются в теле запроса в JSON объекте. Поменять status = available > sold  Request Body: { "id": 10, "category": { "id": 1, "name": "animal" }, "name": "Bulldog", "photoUrls": [ "https://site.com/bulldog/photo" ], "tags": [ { "id": 1, "name": "#dog" } ], "status": "sold" }	

ШАГИ 4

1	Отправить POST запрос <a href="https://petstore.swagger.io/v2/pet">https://petstore.swagger.io/v2/pet</a>	Запрос успешно отправлен на сервер
2	Проверить статус код	200 OK
3	Проверить тело ответа от сервера	Тело ответа возвращается от сервера в формате JSON, и имеет следующий вид: { "id": 10, "category": { "id": 1, "name": "animal" }, "name": "Bulldog", "photoUrls": [ "https://site.com/bulldog/photo" ], "tags": [ { "id": 1, "name": "#dog" } ], "status": "sold" }  Изменен статус питомца: "status": "sold"

4	Проверить структуру типов данных в теле ответа	Ответ содержит корректный JSON-объект с полями: id, name, status. Вложенные структуры соответствуют отправленным данным.
---	--	--

116 DEL 200 OK; Delete an existing pet

Тип	Автоматизация	Автор	Длительность
 Тестовый сценарий	 Ручной	 Р <a href="#">Руслан Гадельшин</a>	00:10:00
Приоритет	Статус	Секция	Дата создания
 Средний	 Не готов	Pet	09.04.2025
Дата изменения			
09.04.2025			






ПРЕДУСЛОВИЯ ТЕСТА 2

1	Headers request: Content-Type: application/json	
2	Test Data: Pet's ID: 10	

ШАГИ 3

1	Отправить GET запрос: <a href="https://petstore.swagger.io/v2/pet/10">https://petstore.swagger.io/v2/pet/10</a>	Запрос успешно отправлен на сервер, питомец с id=10 успешно удален из списка
2	Проверить статус-код	200 OK
3	Проверить тело ответа от сервера	Тело ответа приходит в формате JSON и имеет следующий вид: { "code": 200, "type": "unknown", "message": "10" } Питомец с id = 10 удален из списка.

117 Negative Tests

Тип	Автоматизация	Автор	Длительность
 Чек-лист	 Ручной	 Р <a href="#">Руслан Гадельшин</a>	00:10:00
Приоритет	Статус	Секция	Дата создания
 Средний	 Не готов	Negative Tests	09.04.2025
Дата изменения			
14.04.2025			

ШАГИ 30

1	Попытка отправить запрос с помощью некорректного метода (GET, PUT, DELETE). Недопустимое значение > отображение правильного статус код об ошибке (400, 401, 403)
2	Правильность написания url и endpoint в соответствии с контрактом. Негативно: некорректный url. Недопустимое значение > отображение правильного статус кода об ошибке (400, 401, 403)
3	Правильность значения ключей в endpoint. Негативно: отправка запроса с некорректным значением (отрицательный, буква, спецсимвол, другой тип данных). Недопустимое значение > отображение правильного статус кода об ошибке (400, 401, 403)
4	Проверка статус - кодов ответа. Положительный кейс - 200 OK; Отрицательный кейс - 400 или какой либо другой показывающий ошибку.
5	Проверка заголовка:
6	Использовать application/xml вместо json. > отображение правильного статус кода об ошибке 415 (400, 401, 403)
7	Оставить заголовки пустыми > отображение правильного статус кода об ошибке 415 (400, 401, 403)
8	Проверки в теле ответа:

9	Ключи и значения: проверка на опечатки в документации и в реальном АПИ. Негативно: некорректные значения в теле запроса, а именно ключей значений. Недопустимое значение > отображение правильного статус кода об ошибке (400, 401, 403)
10	Соответствие типов данных (строка, число, массив). Негативно: некорректные значения(вместо числа буквы или спецсимволы или наоборот, исп. классов эквивалентности). Недопустимое значение > отображение правильного статус кода об ошибке (400, 401, 403)
11	Проверка корректности даты.
12	Проверка ID (всегда должно быть на 1 больше). Негативно: невалидный ID (отрицательный, буква, спецсимвол, другой тип данных), несуществующее максимальное значение, null, NULL. При вводе недопустимого символа > отображение правильного статус кода об ошибке (400, 401, 403)
13	Проверка на максимальное / минимальное значение с помощью техники граничных значений. При вводе больше максимального / минимального > отображение правильного статус кода об ошибке (400, 401, 403)
14	Попытка отправить запрос с null вместо какого либо значения. > отображение правильного статус кода об ошибке (400, 401, 403)
15	Пустое тело значения. > отображение правильного статус кода об ошибке (400, 401, 403)
16	<b>Проверки на корректность обработки структуры JSON:</b>
17	Изменить порядок следования пар "ключ: значение" в теле JSON-объекта. Сервер должен обработать такие запросы идентично, независимо от порядка ключей. Ошибок быть не должно. JSON-объекты не чувствительны к порядку полей, поэтому изменение порядка не должно влиять на результат > вызывать ошибку.
18	Вставить запятую в конец объекта, заменить запятую на точку с запятой или удалить запятую между элементами. Ожидаемый результат — сервер возвращает корректный статус-код ошибки (400, 401, 403), с указанием проблемы в структуре. <b>Типичная ошибка при сериализации данных</b> , особенно при ручном вводе.
19	Попробовать поменять фигурные скобки <code>{ }</code> (объекты) на квадратные <code>[ ]</code> (массивы) и наоборот. Ожидаемое поведение — сервер должен ответить соответствующей ошибкой, так как структура JSON нарушена.
20	Проверить, что сервер корректно обрабатывает некорректный JSON-формат. В случае синтаксической ошибки (например, отсутствует закрывающая лишняя запятая и т.п.) — возвращается соответствующий статус-код ошибки (400 Bad Request и т.п.).
21	<b>Проверки в авторизации; Authorization / Authentication</b> (если API требует токены/ключи)
22	Отправка запроса без токена → 401
23	С невалидным токеном → 401
24	С истекшим токеном → 401 или 403
25	<b>Rate limiting / Throttling:</b>
26	Попробовать отправить X запросов за короткое время → 429 Too Many Requests (если применяется)
27	<b>Access control (роль пользователя):</b>
28	Попробовать доступ к защищённому ресурсу под пользователем без нужной роли → 403 Forbidden
29	<b>Unsupported Media Type:</b>
30	Попробовать отправить <code>Content-Type: text/plain</code> вместо <code>application/json</code> → 415