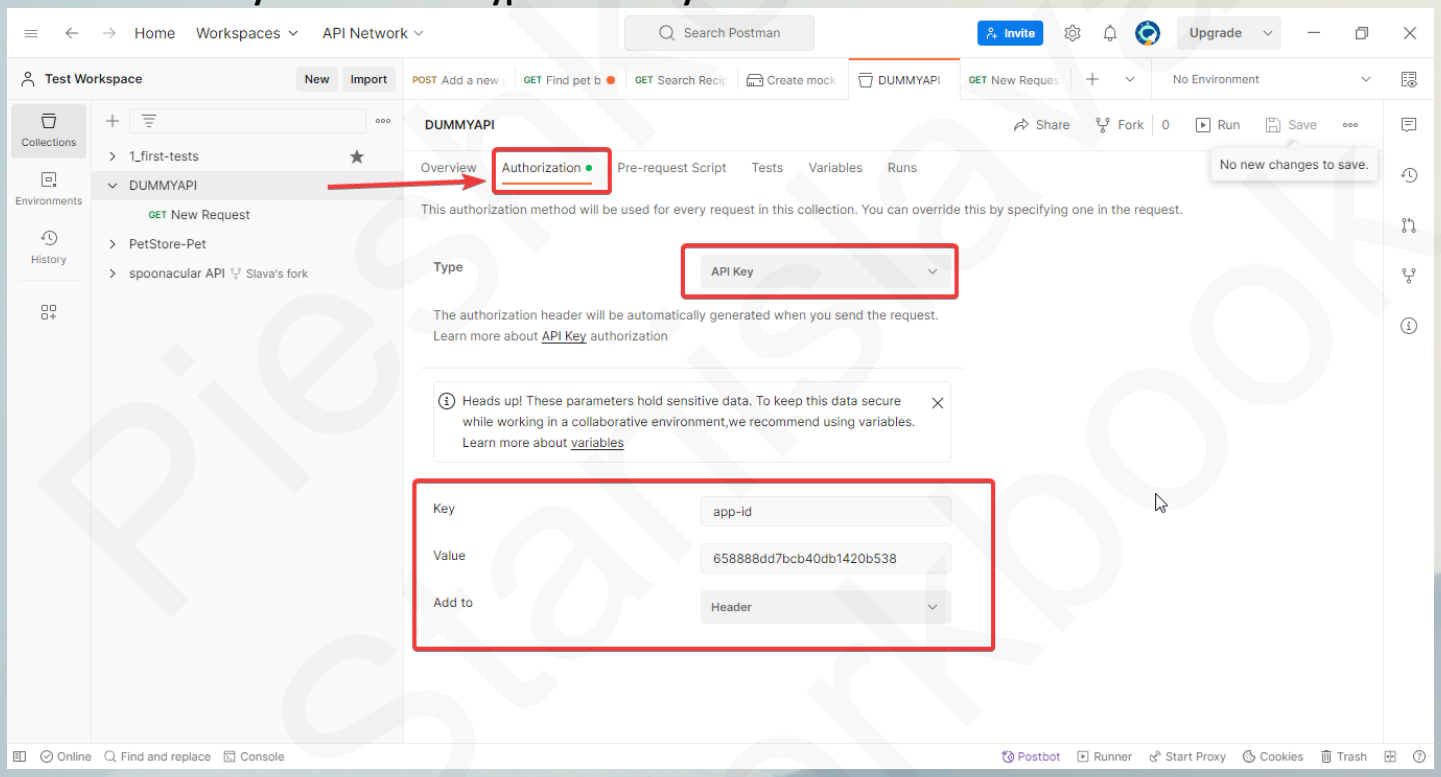
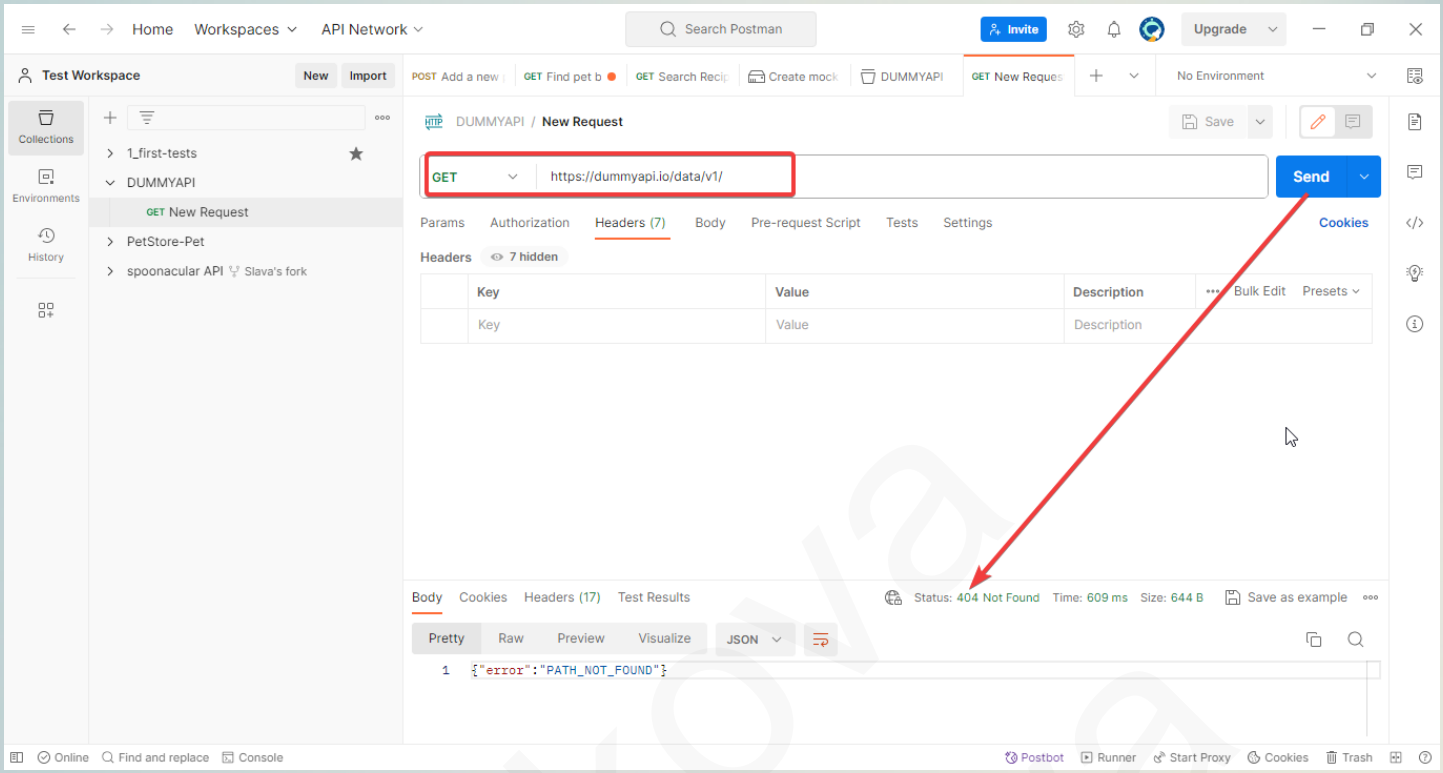
API Practice

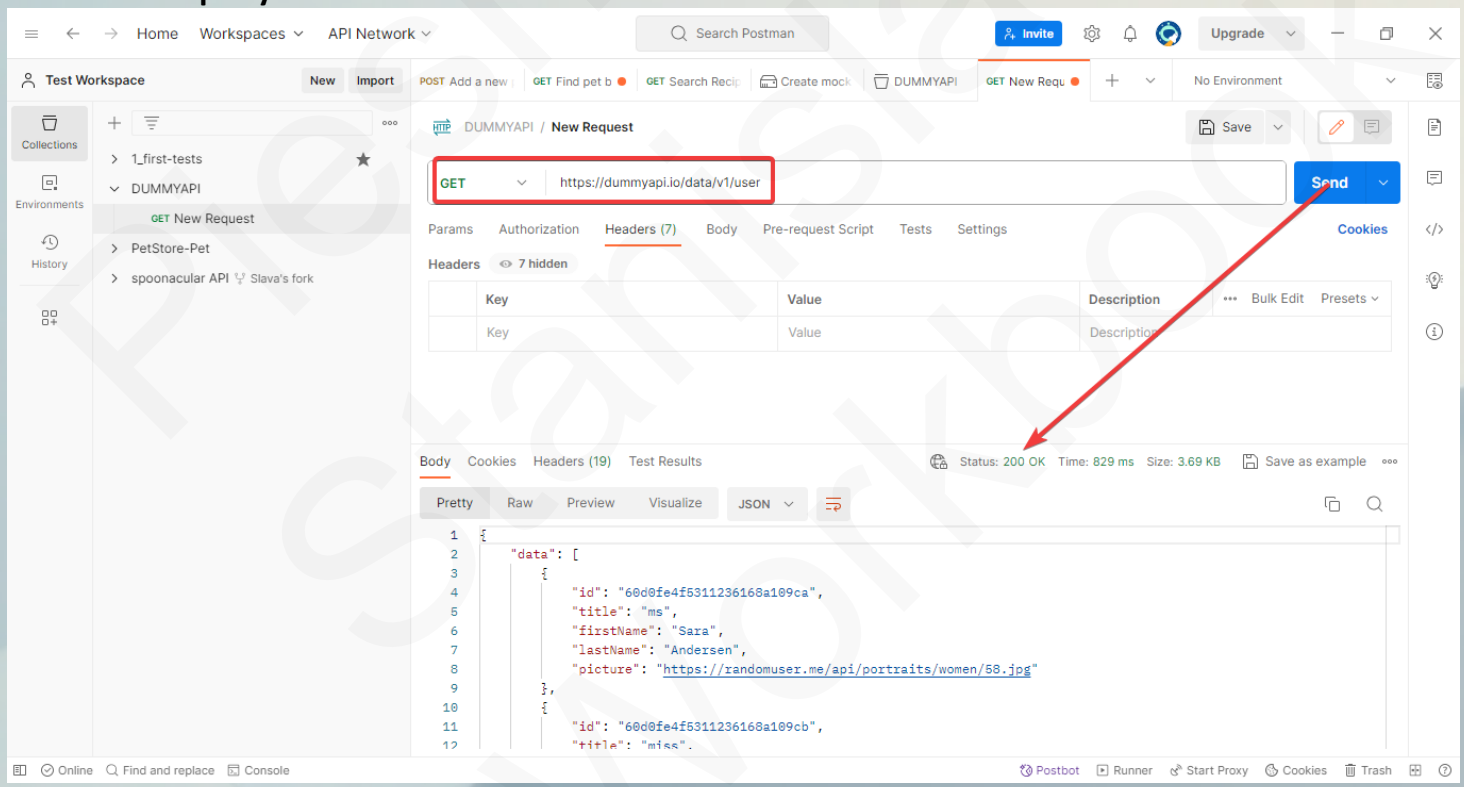
1. Go to https://dummyapi.io/ and click on login. Log in with your Google account and generate an App ID. Create the DUMMYAPI. Go to the DUMMYAPI collection and select the Authorization tab. Select the API Key authorization type. Enter Key and Value. Click Save.



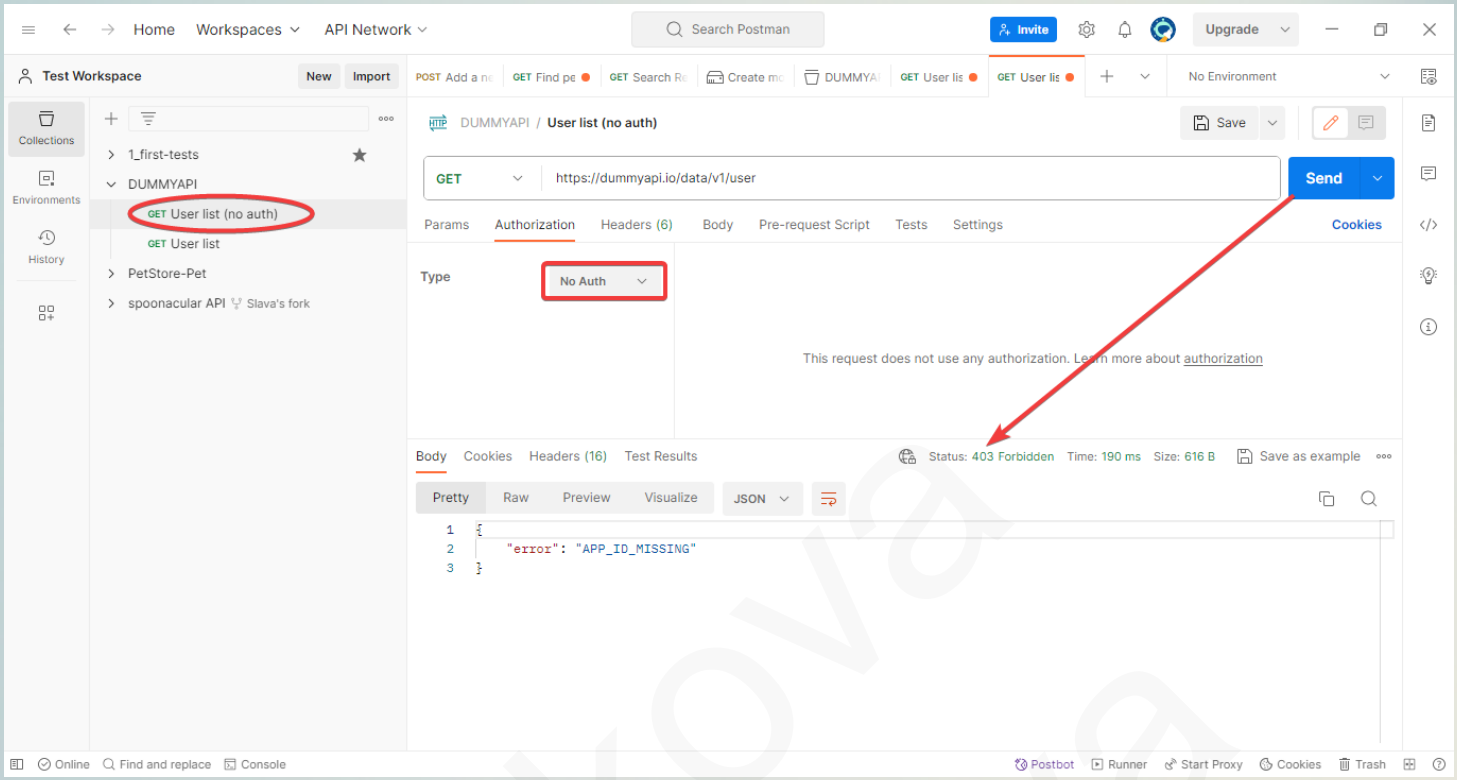
2. We take our base URL (https://dummyapi.io/data/v1/) according to the documentation and paste it into a new request. Choose the GET method. Send the request.



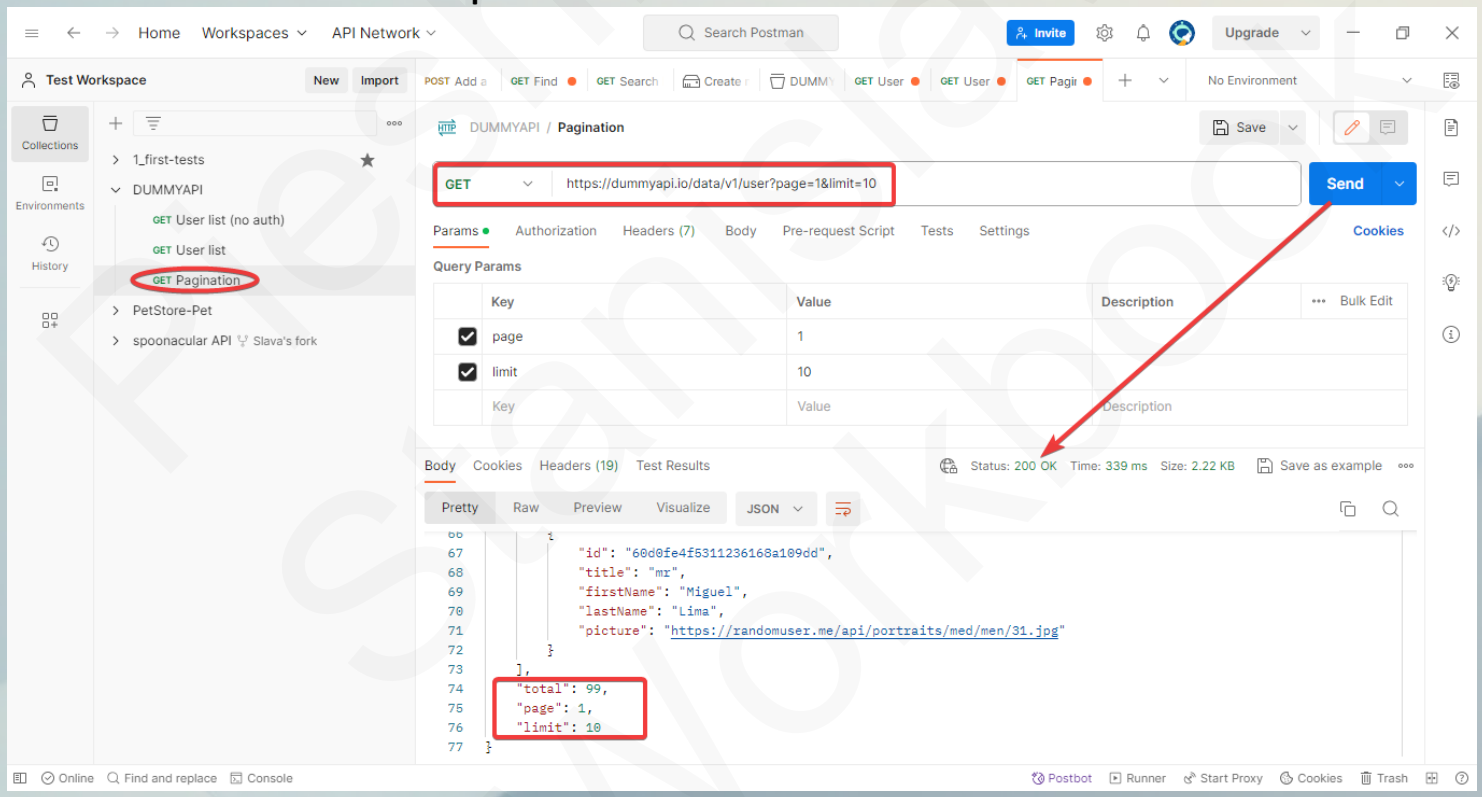
3. We receive a 404 status code in the response. This means that the URL is not recognized and the server cannot find the requested resource. Go to the documentation and fix the error.
To get a list of users, we need to add /user at the end of the URL. We get a successful status code 200. Rename the query to User list.



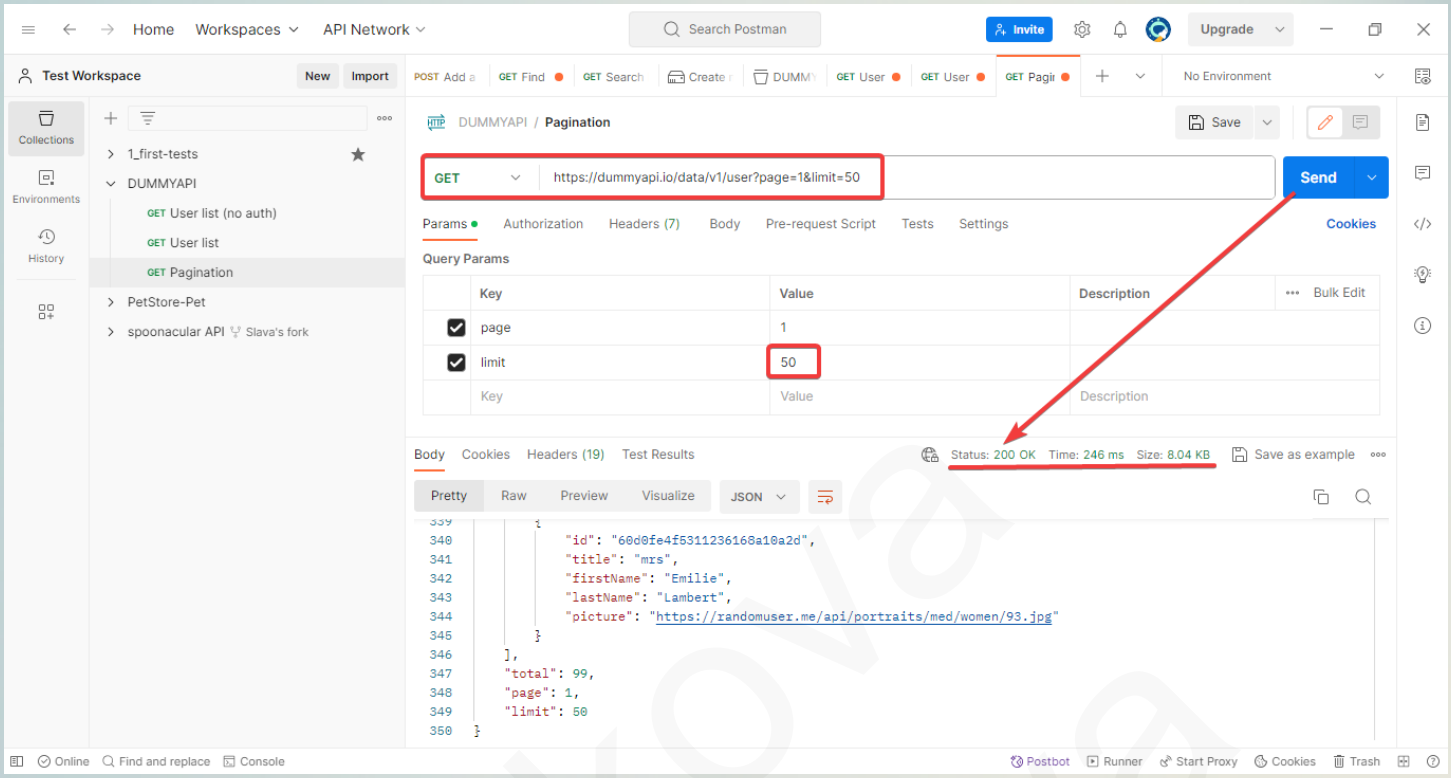
4. Create a GET request without authorization. The name of the new request is User list (no auth). Paste the URL:
https://dummyapi.io/data/v1/user. Click the Send button. We receive a 403 status error code - the client is not authorized.



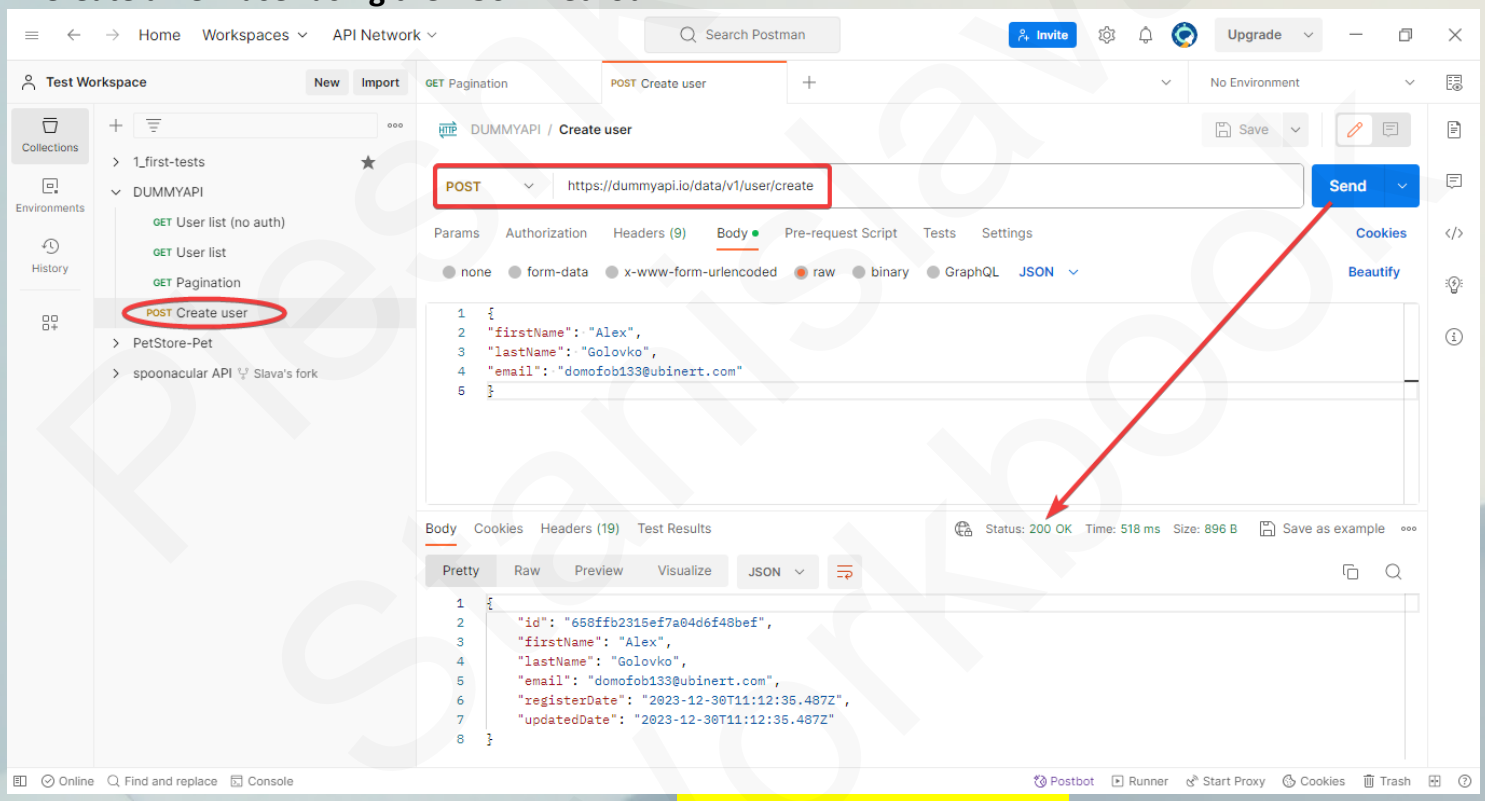
5. Check the pagination conditions. Create a new query and name it Pagination. We use the GET method. Paste the URL from the example https://dummyapi.io/data/v1/user?page=1&limit=10 in the documentation and send the request.



6. Check the upper limit. According to the documentation, this value is 50. I get a 200 status code. The server response time and weight have changed as expected.

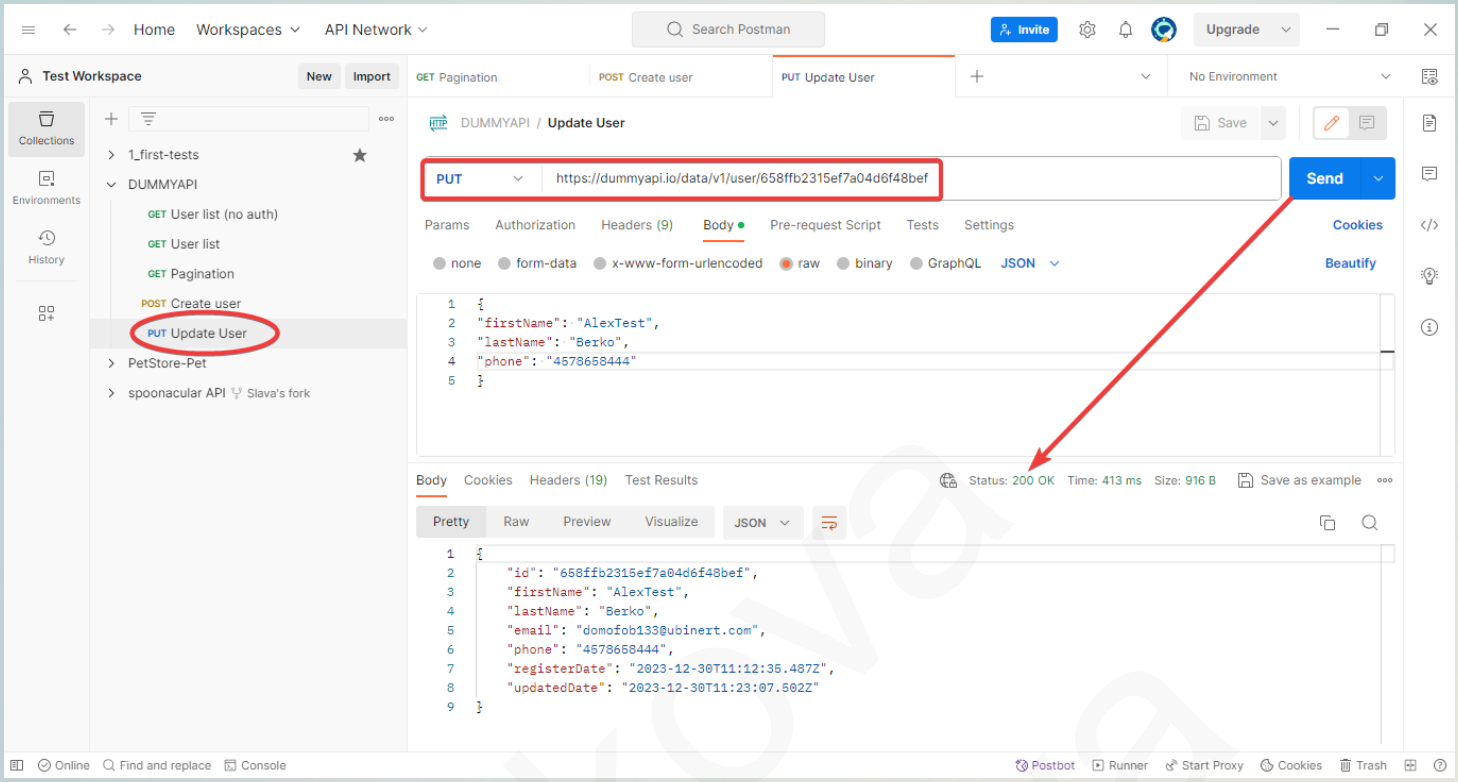


7. Create a new user using the POST method.

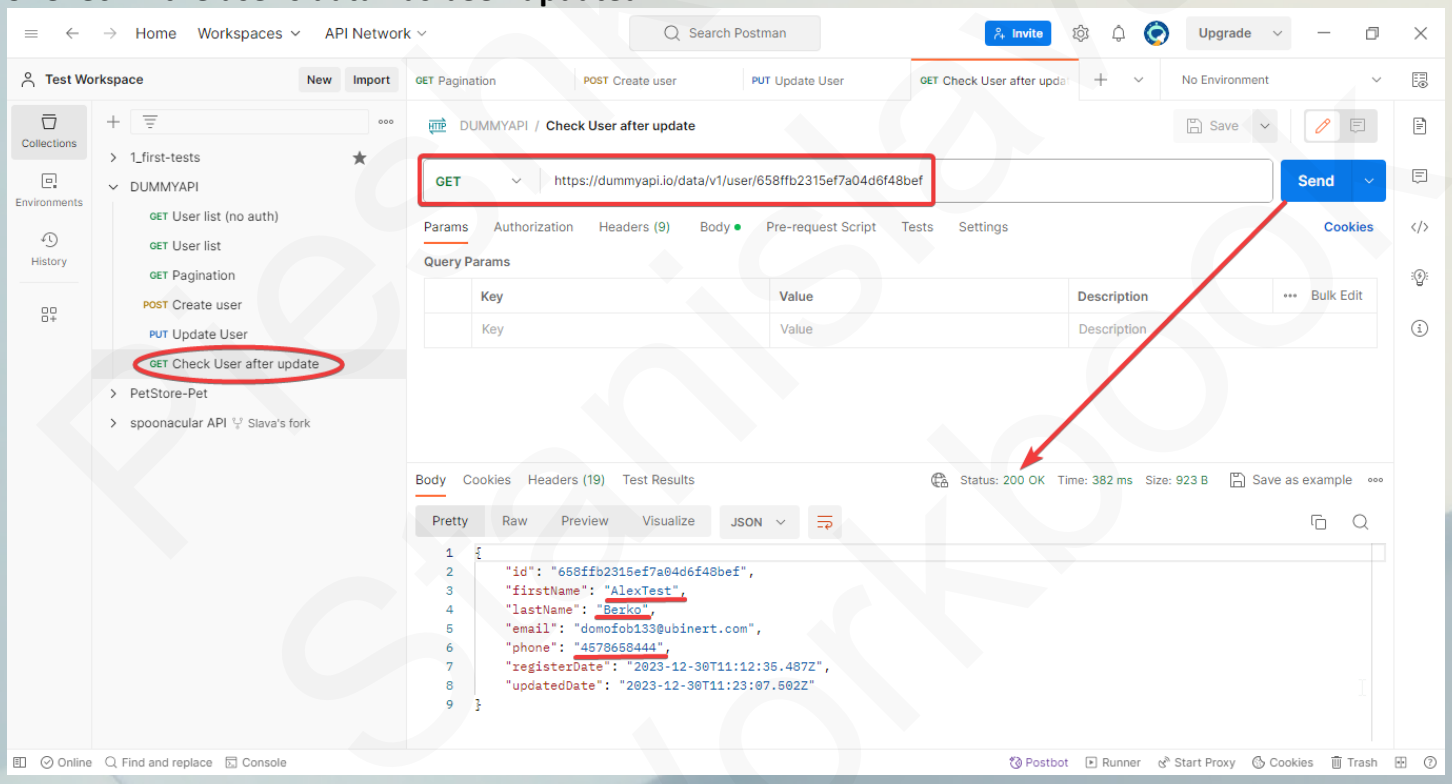


The user has been successfully created. His Id: 658ffb2315ef7a04d6f48bef

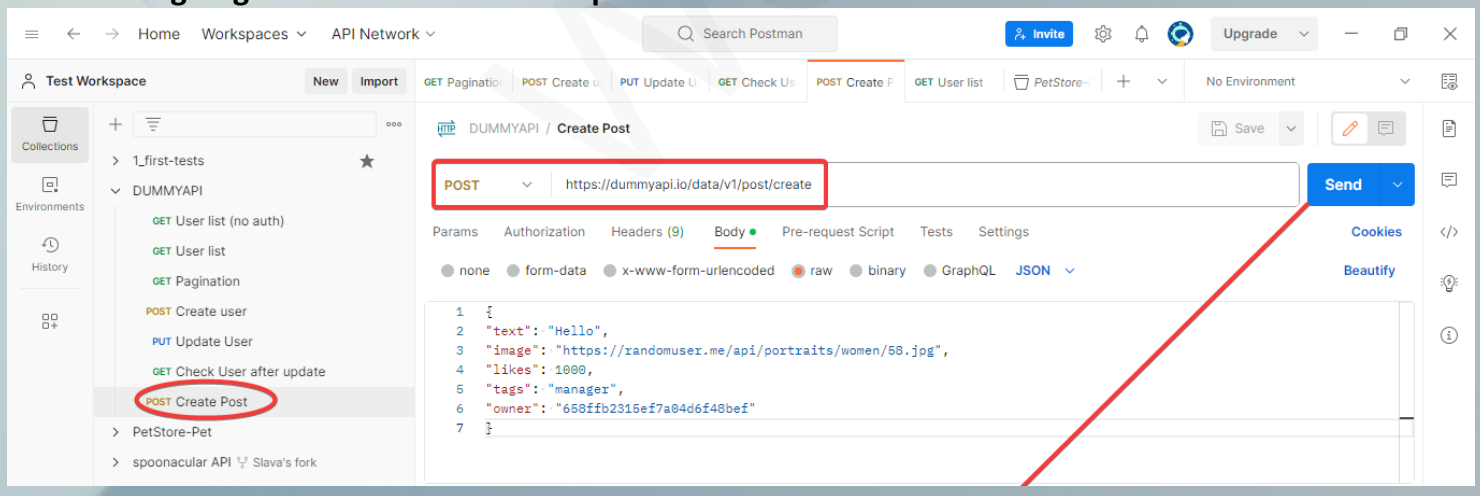
8. Modify a newly created user. Change all the data. The email remains the same.

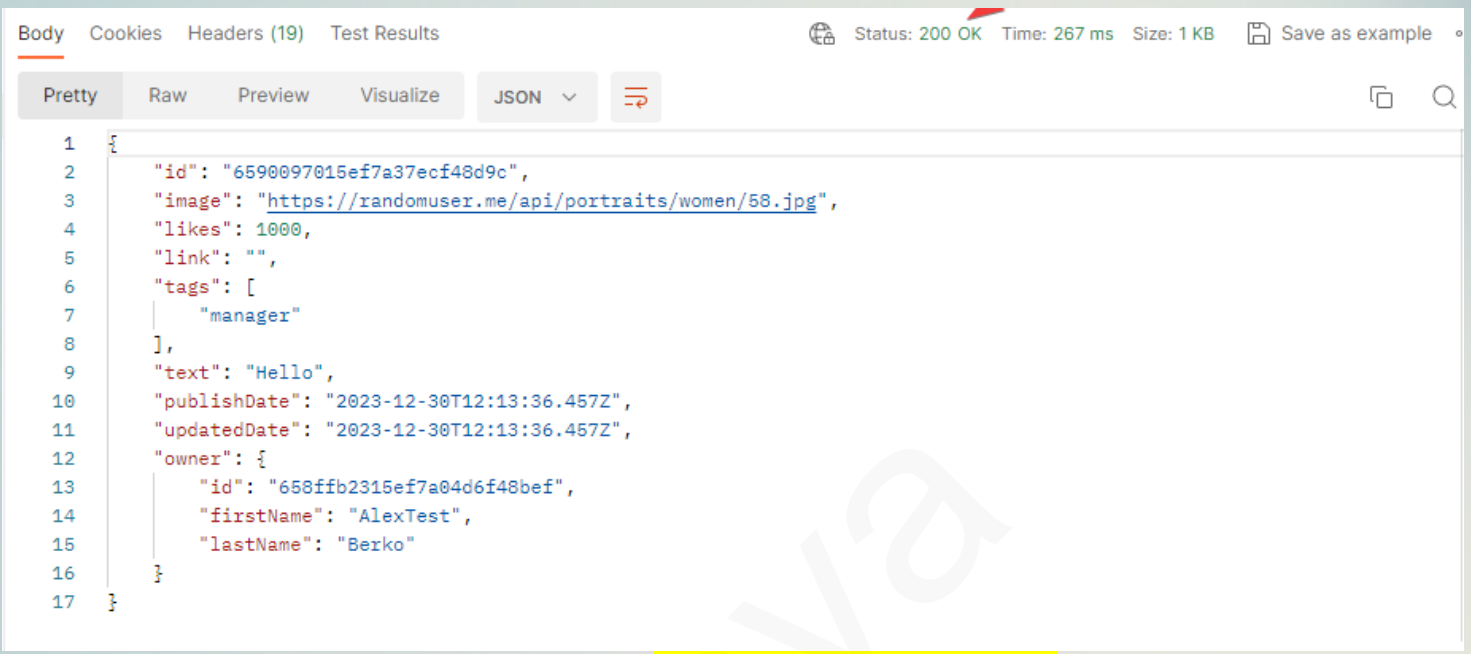


9. Check if the user's data has been updated.



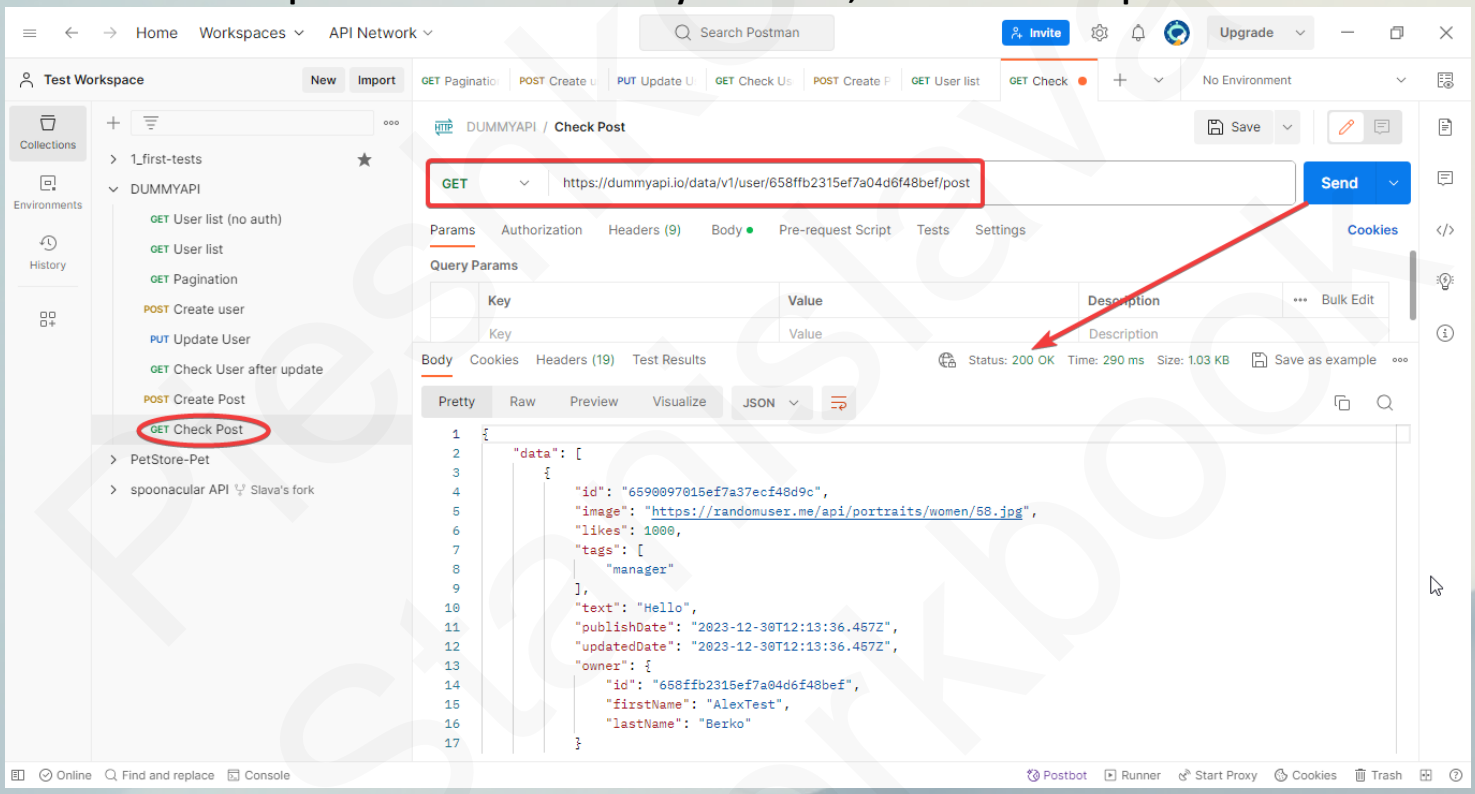
10. We are going to undo the user's first post.



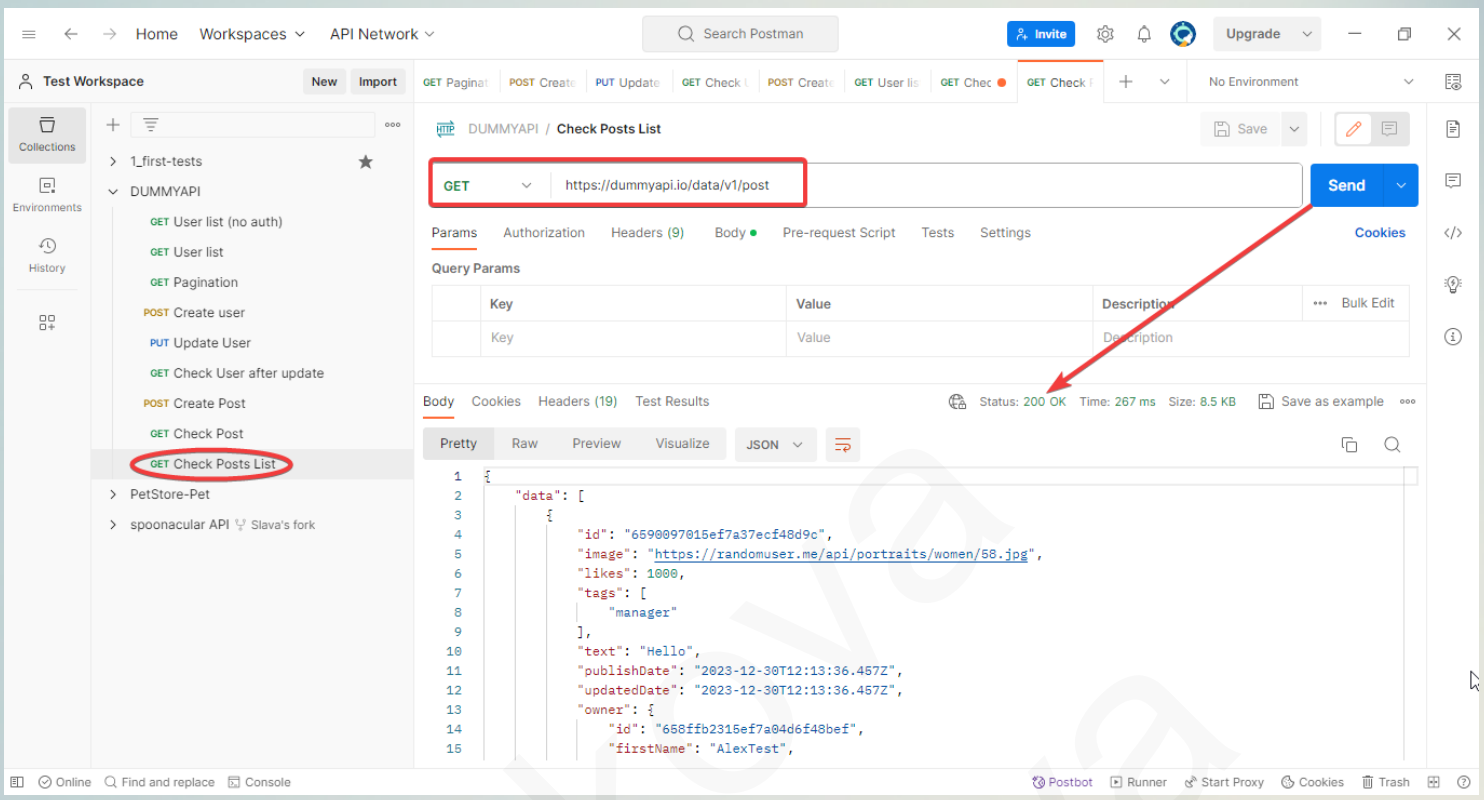


The post was successfully published. Post ID: 6590097015ef7a37ecf48d9c

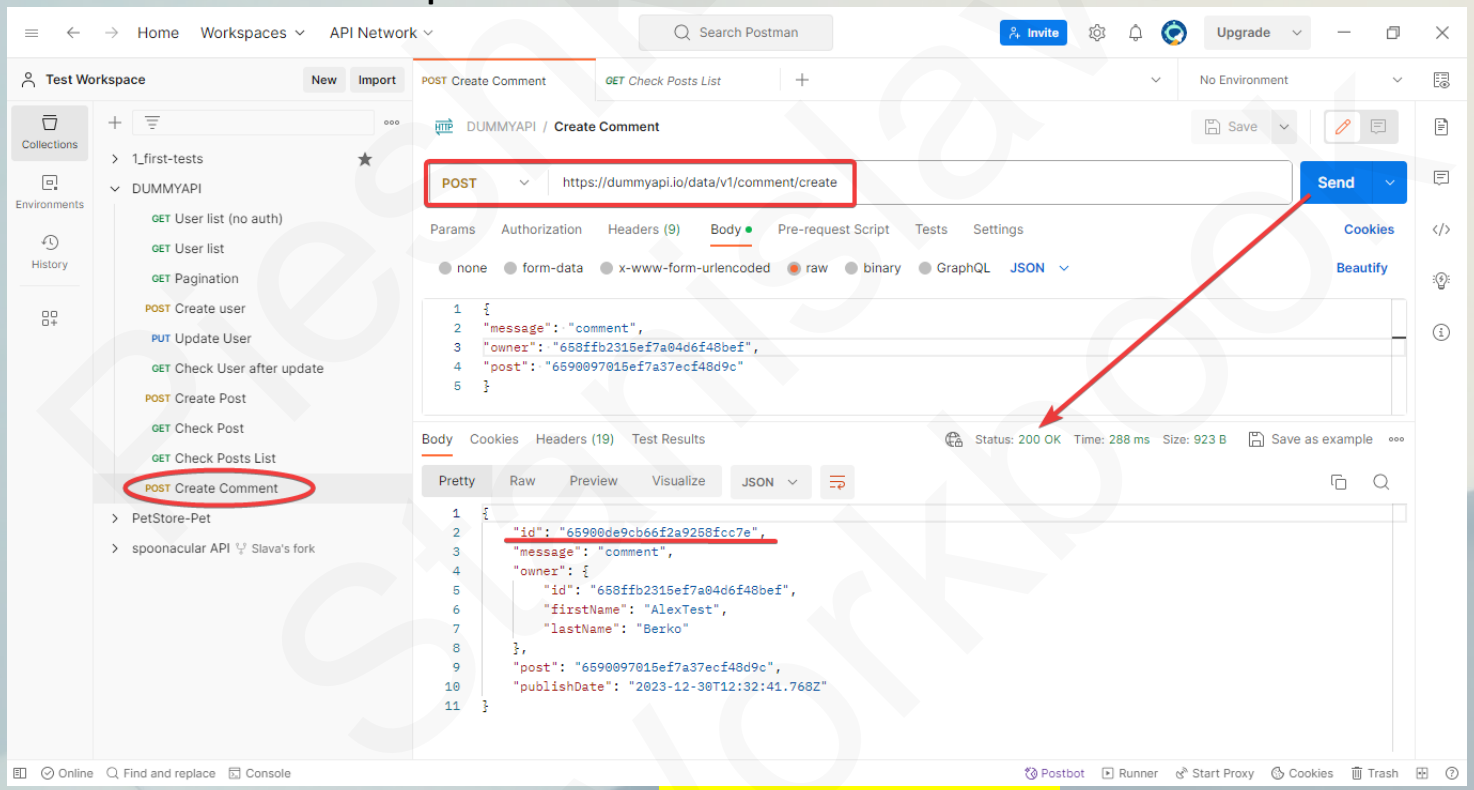
10. We check if our post was created accurately. To do this, we look at all the posts of this user.



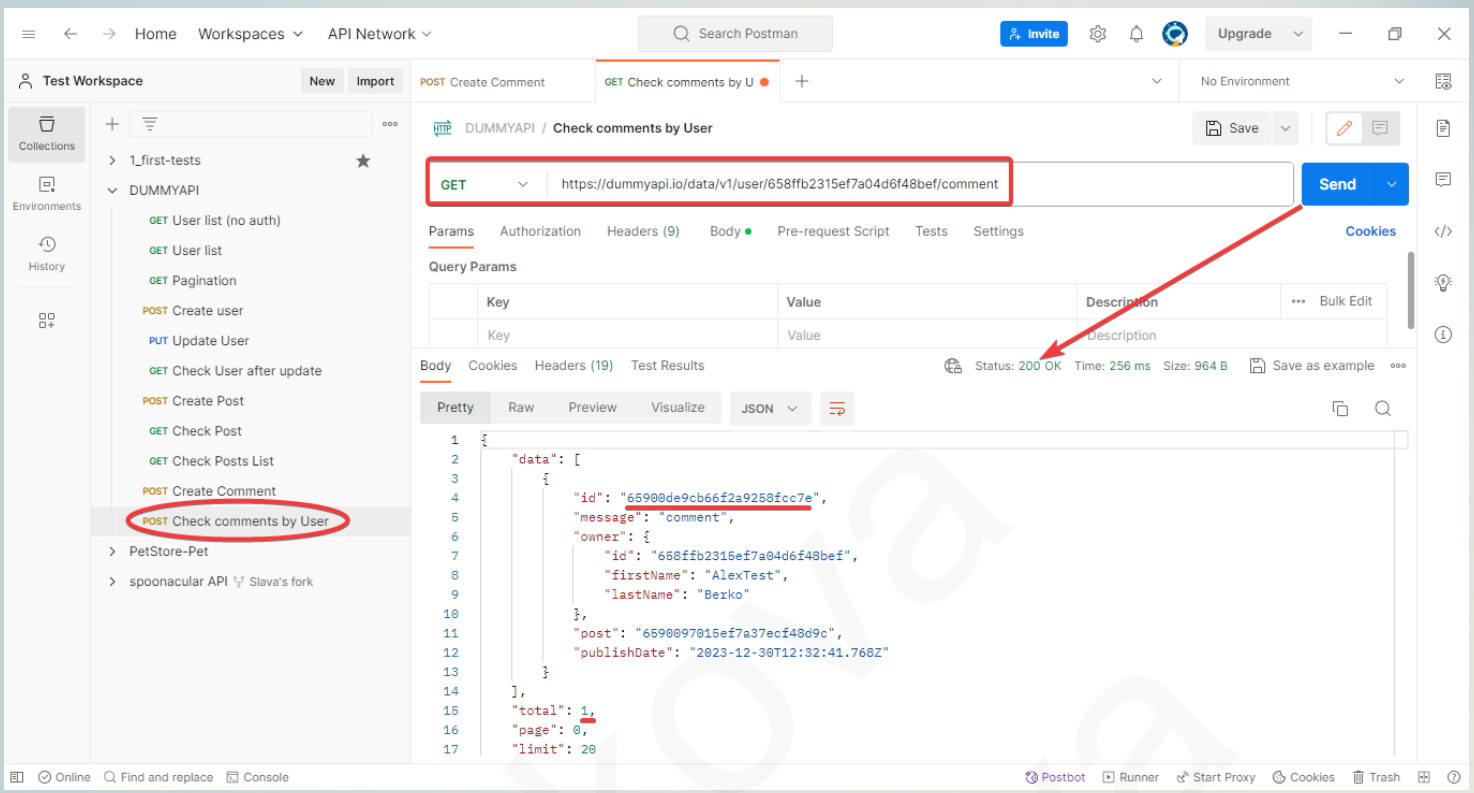
11. Get list of posts sorted by creation date.



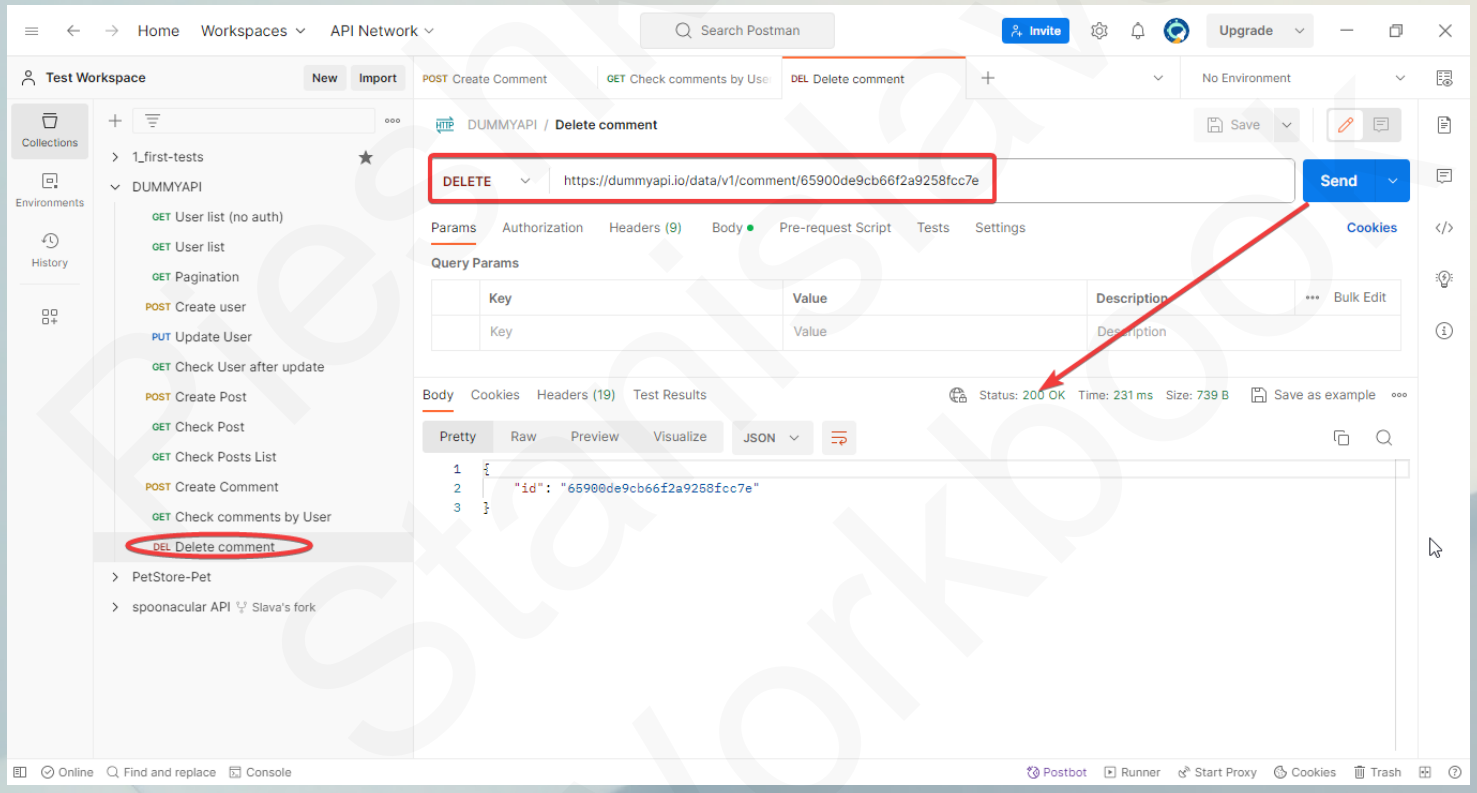
12. Create a comment to the post.



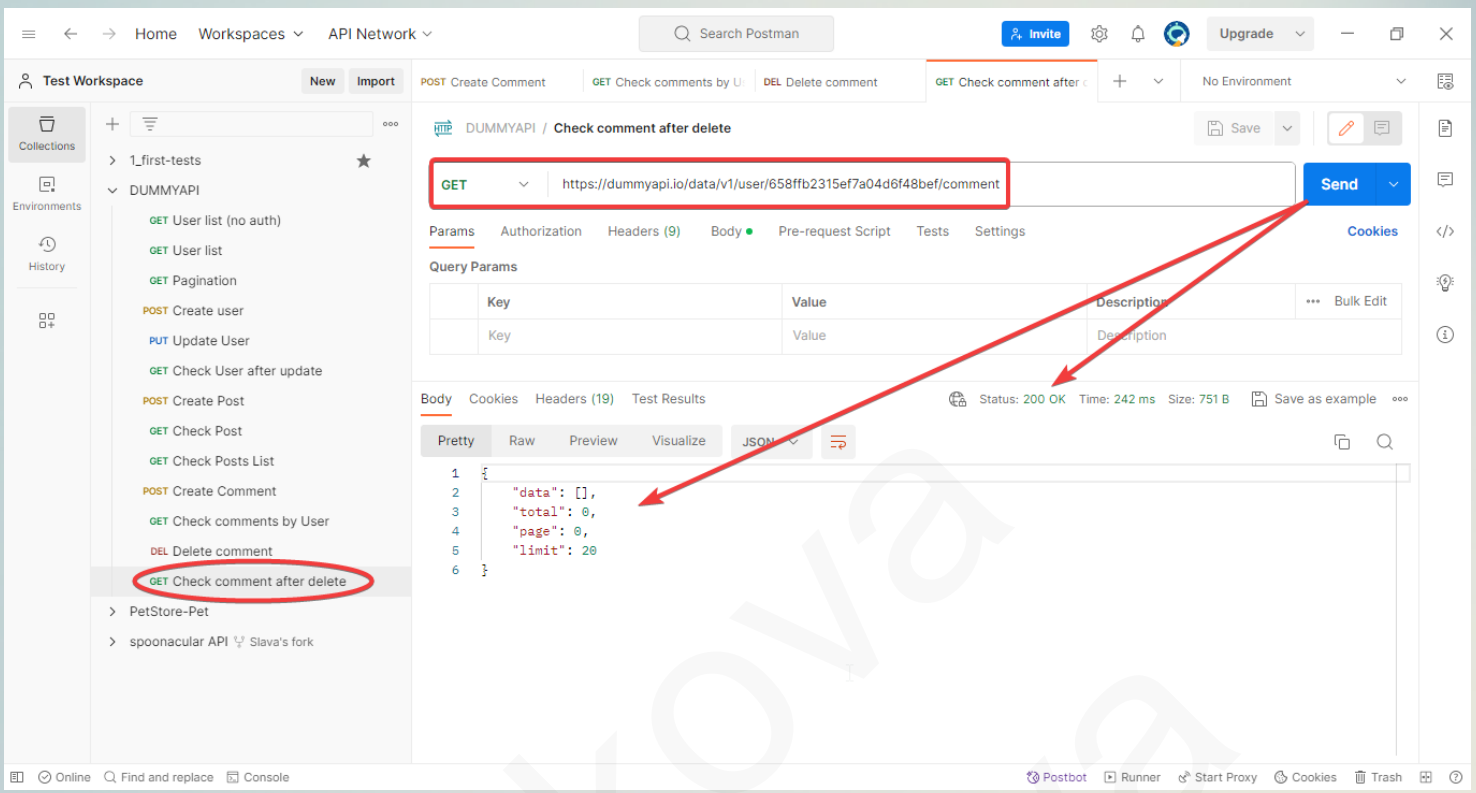
Comment successfully created. Comment ID: 65900de9cb66f2a9258fcc7e



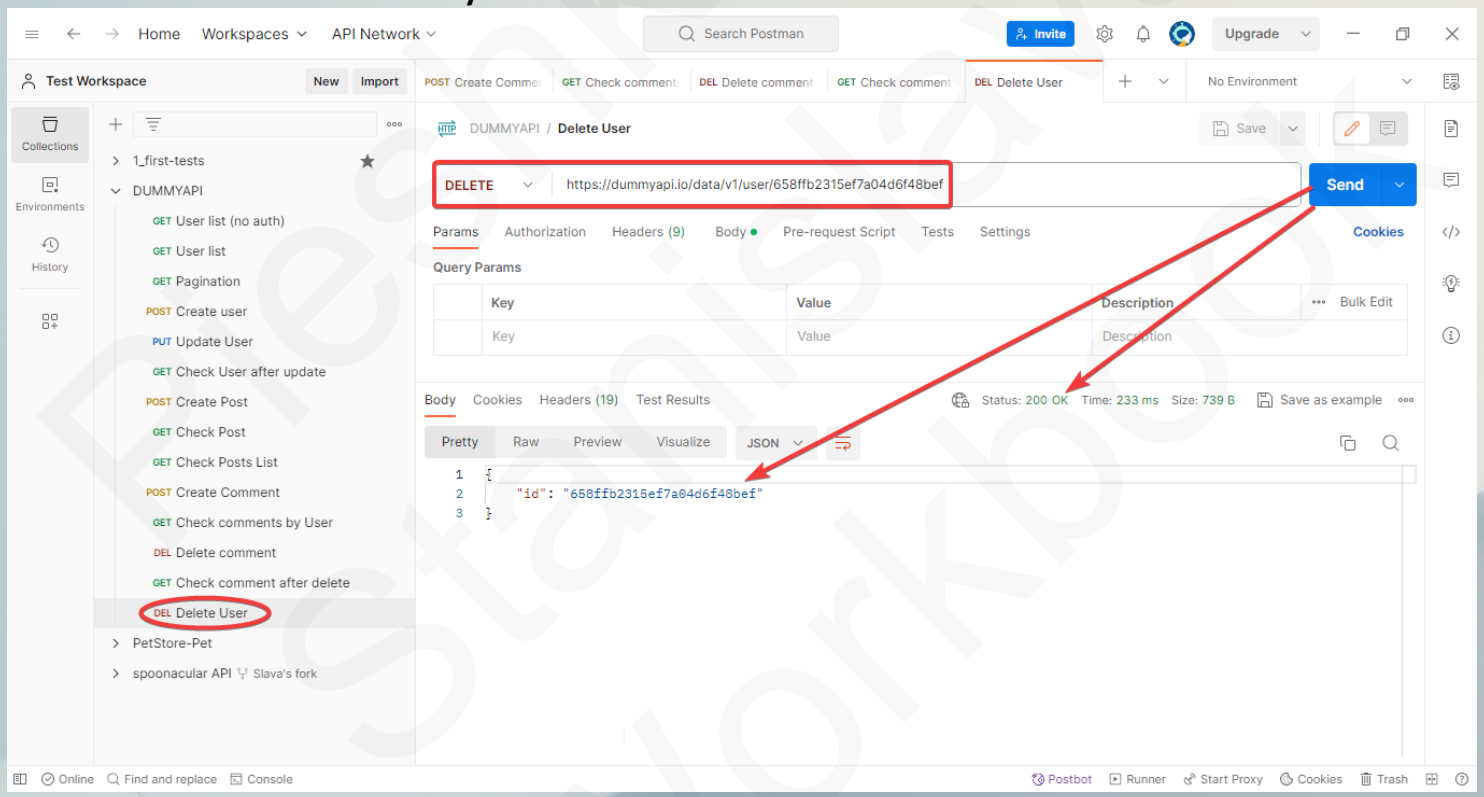
13. Delete the user's comment.



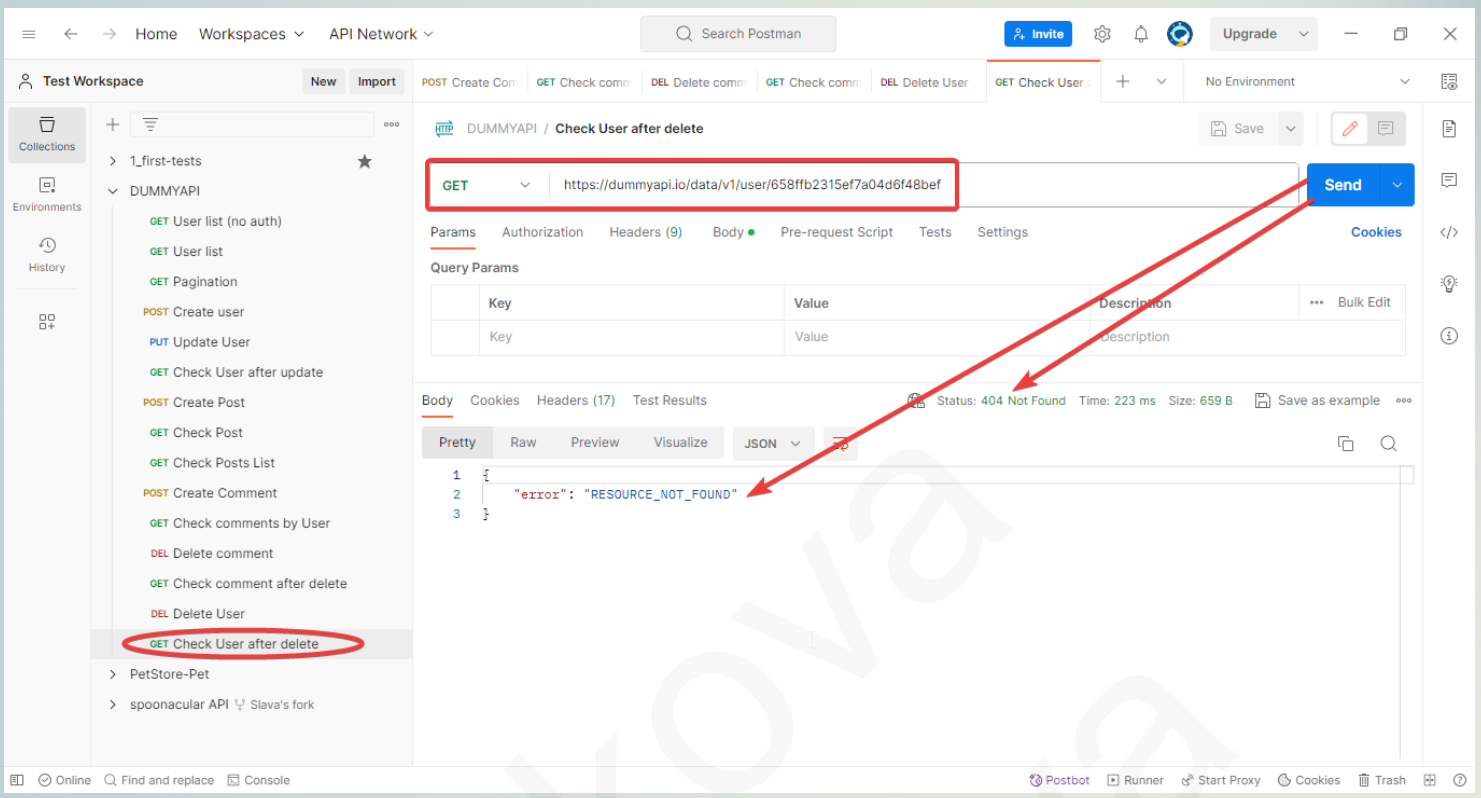
14. Check if the comment was really deleted.



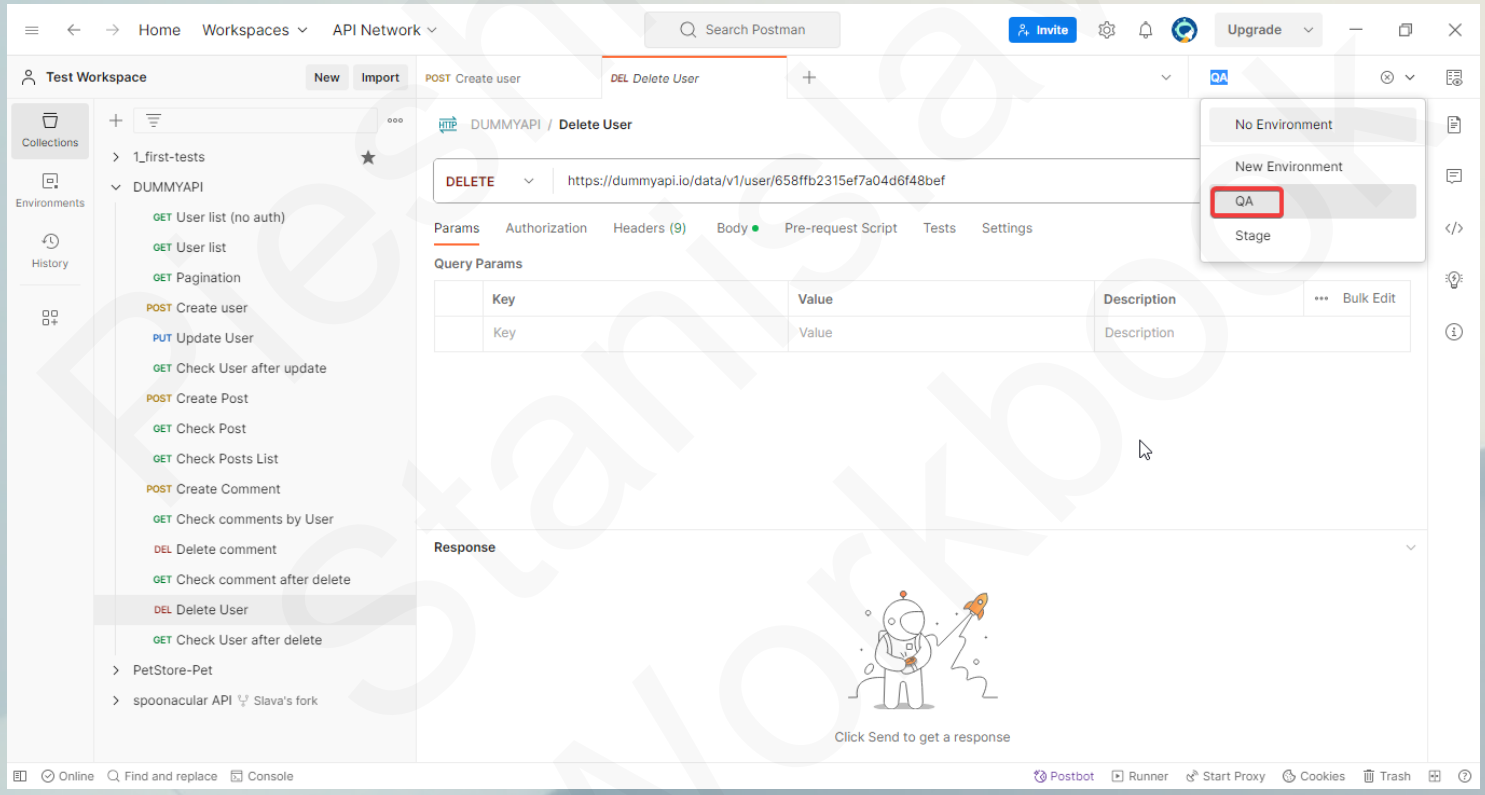
15. Remove the User from the system.



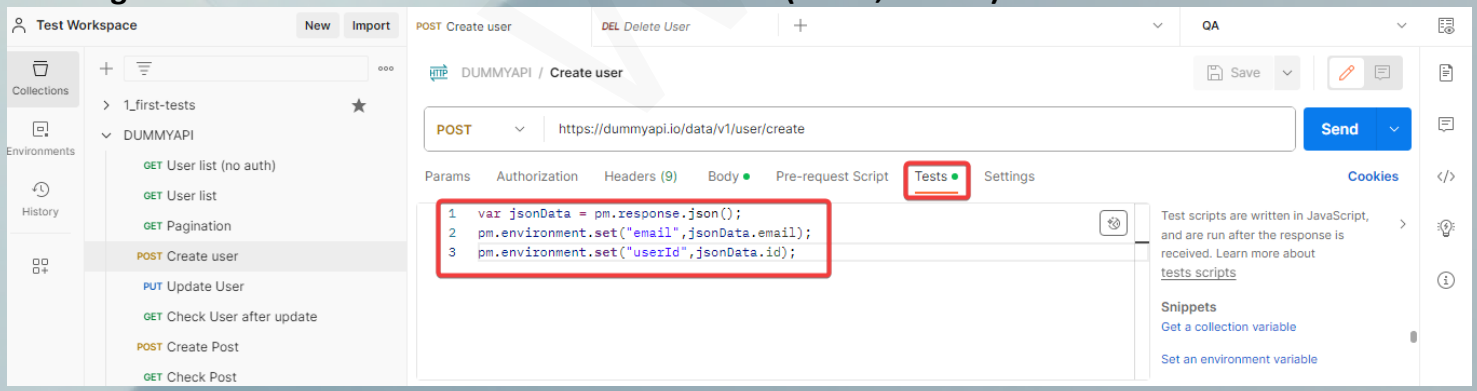
16. Check if the user has really deleted.



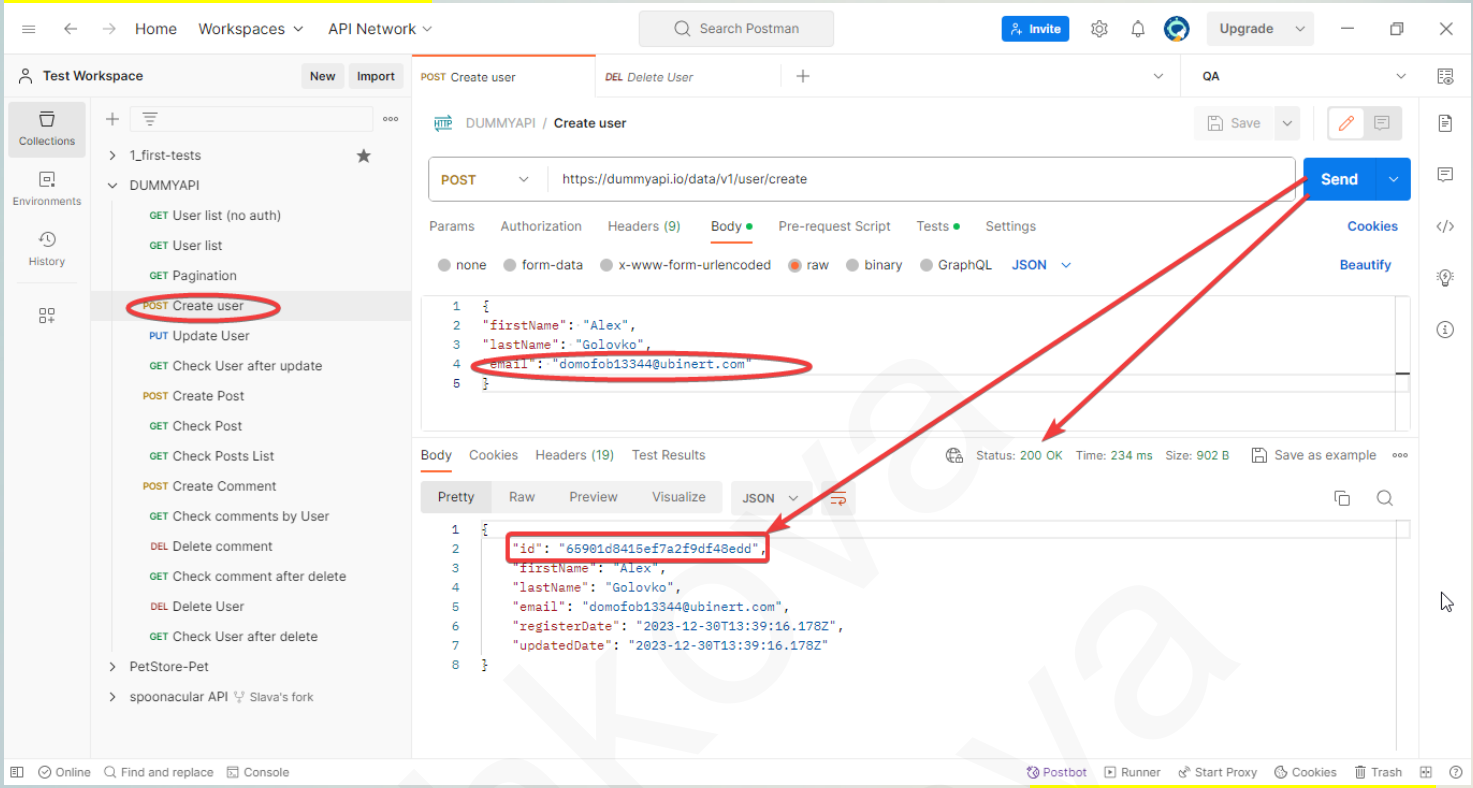
17. Create a new environment for QA.



18. Drag the data that we will work with in the future (email, user Id) into the environment.

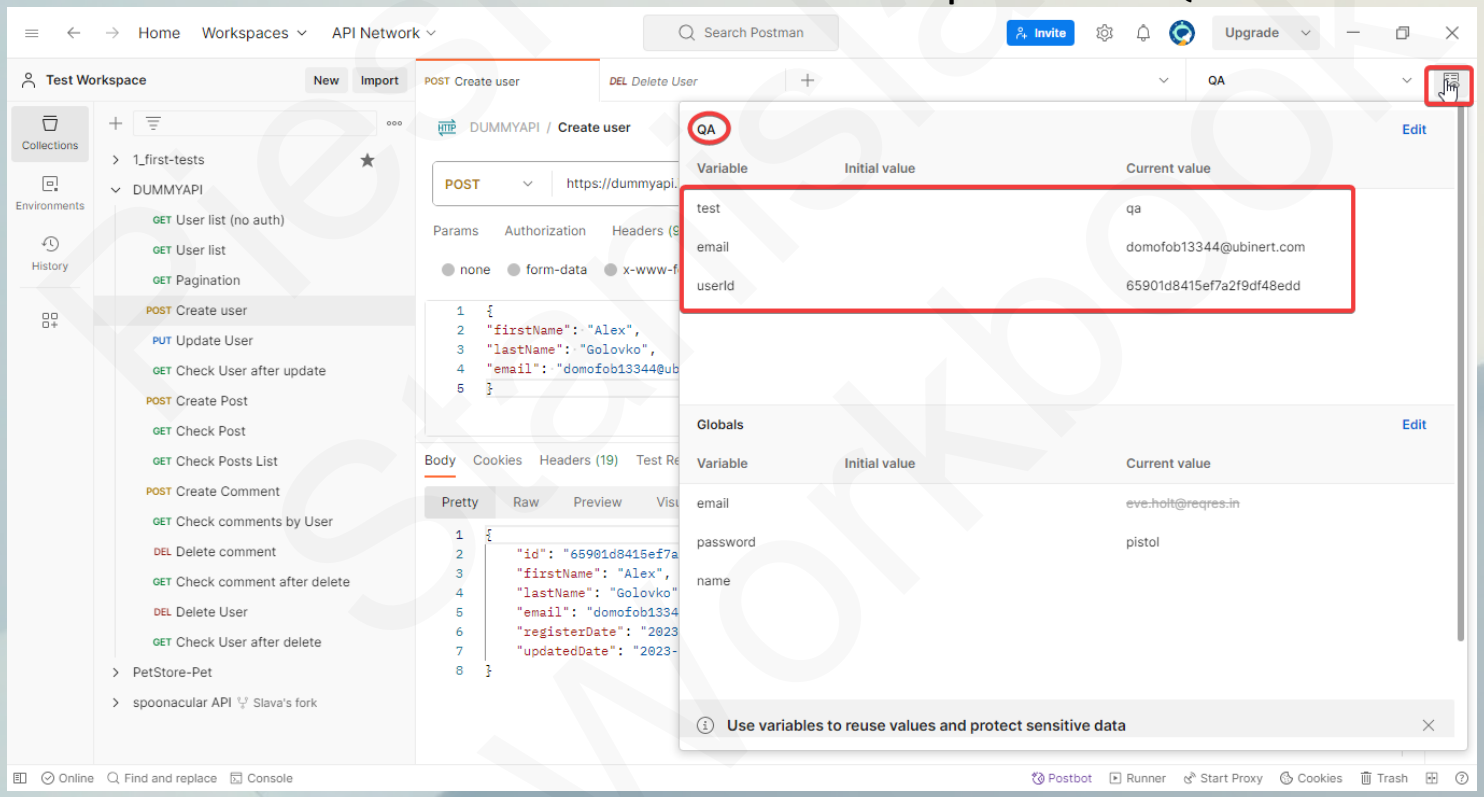


19. To see if our snippet is working correctly, go to the Body section and enter a new email domofob13344@ubinert.com



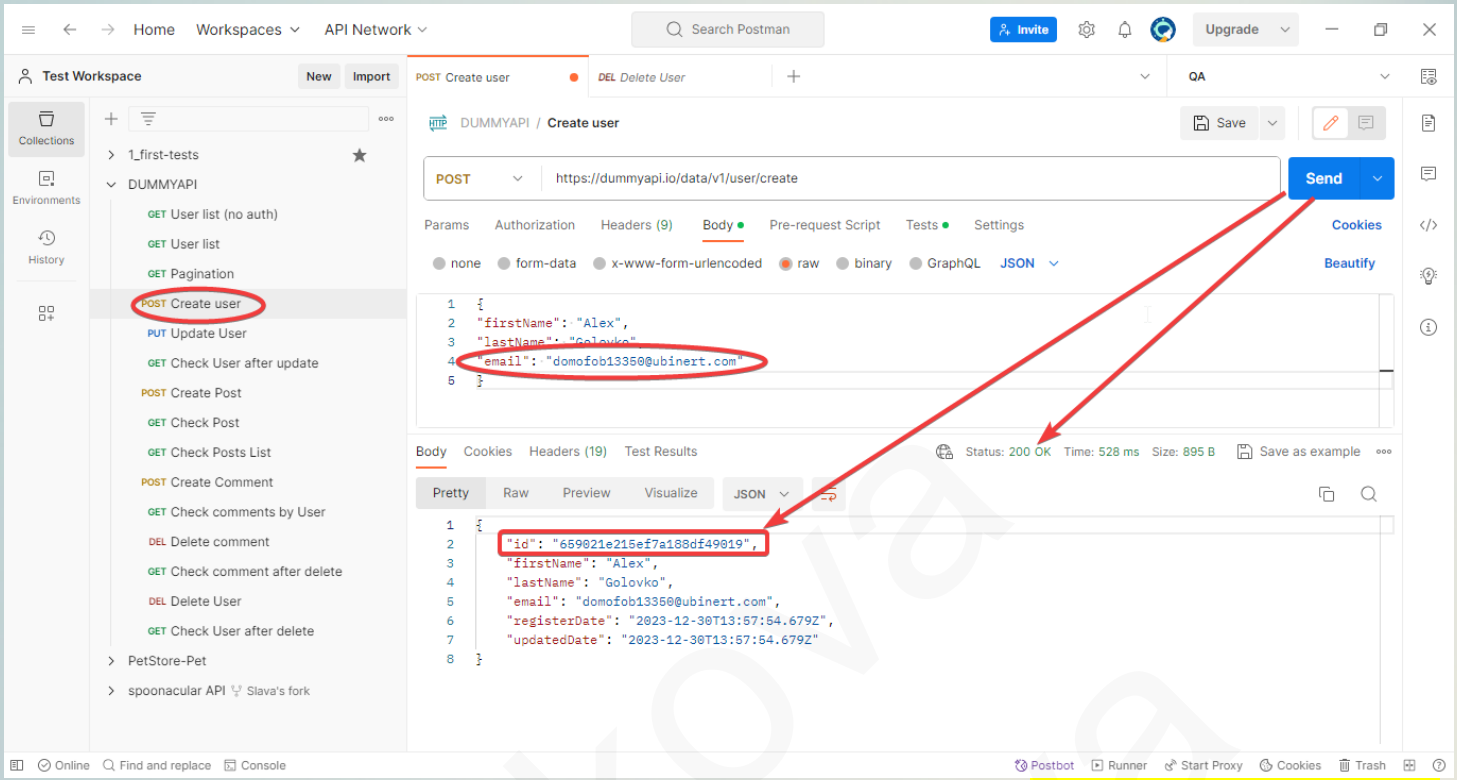
User with new email successfully created. A new id has been provided: 65901d8415ef7a2f9df48edd

20. Go to Environment and check if the new user's data has been pulled into the QA environment.

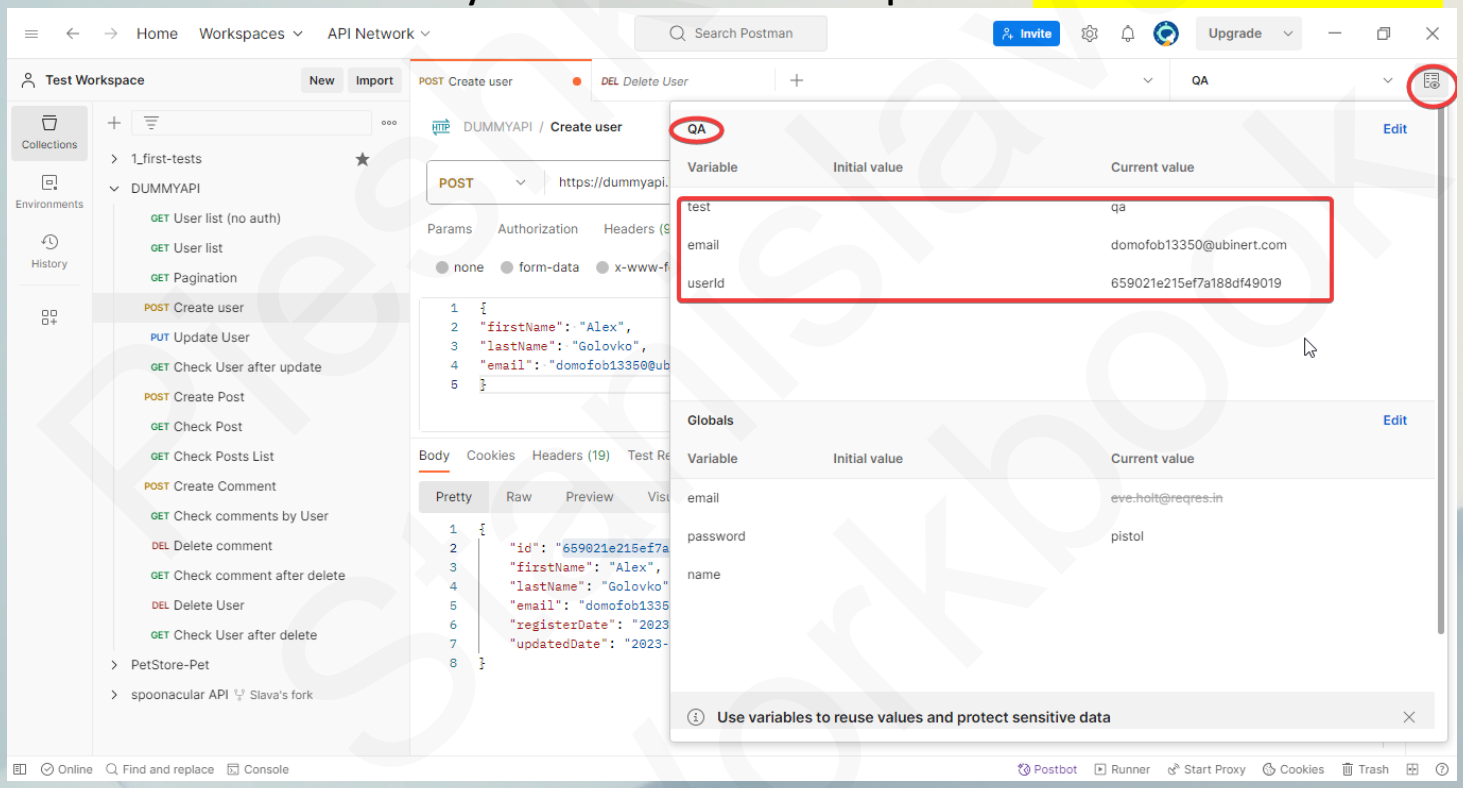


The data has been successfully created.

21. Let's try changing the email to domofob13350@ubinert.com again and make sure that the environment data is updated as well.



User with new email successfully created. A new id has been provided: 659021e215ef7a188df49019



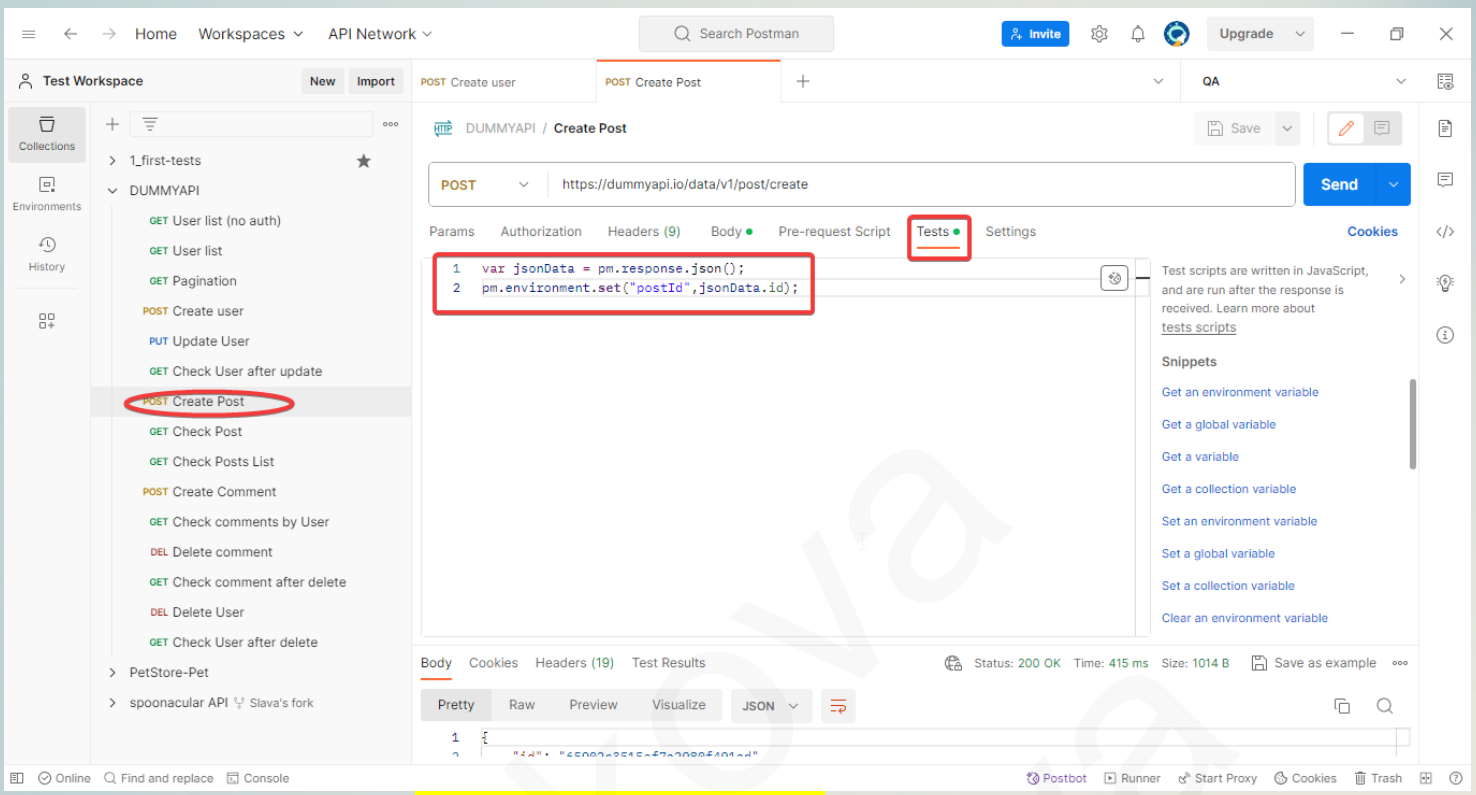
The new email and Id have been successfully updated in the QA environment.

22. The same variables need to be set in two more requests:

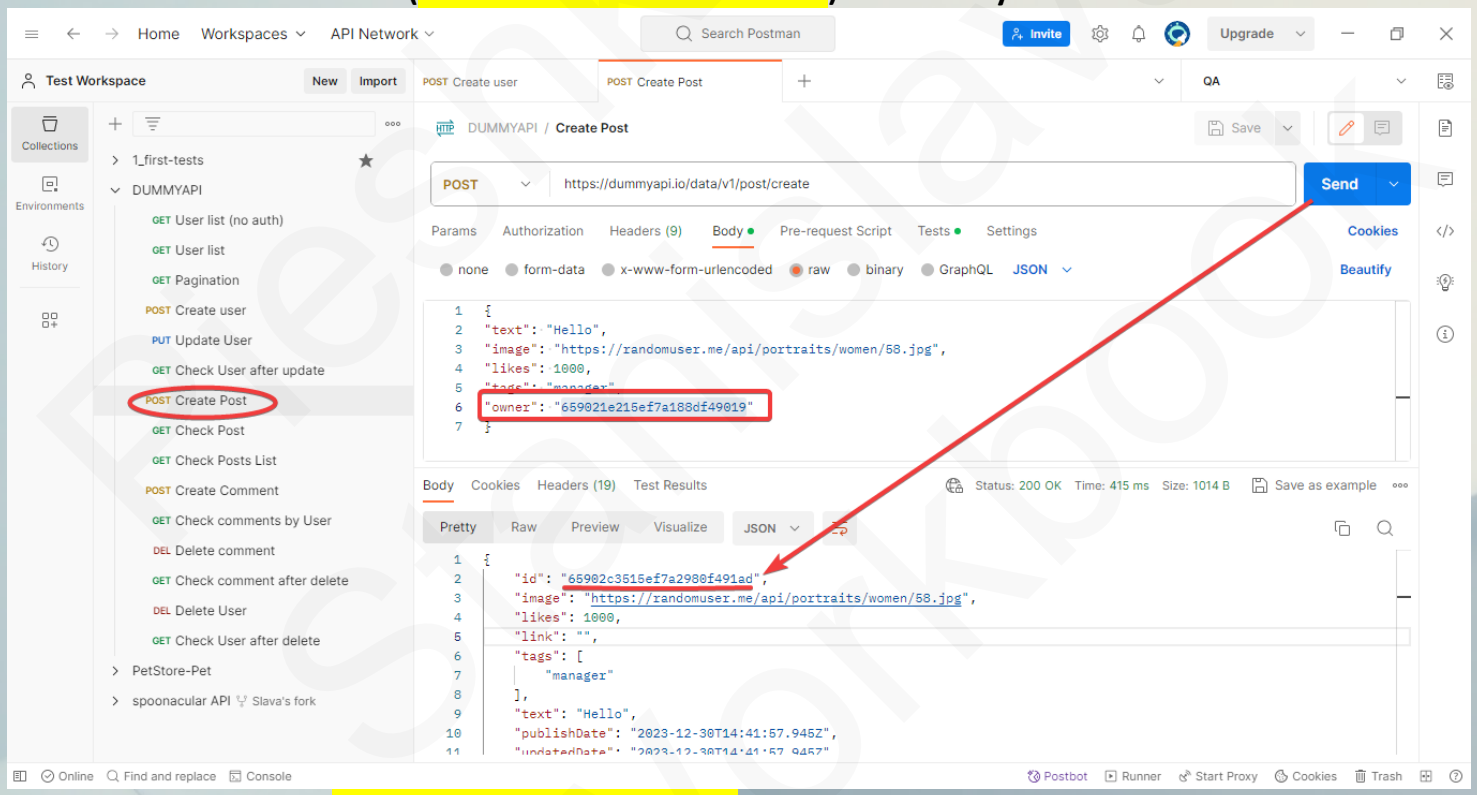
- Create Post

- Create Comment

22.1. Create variables for a new post.

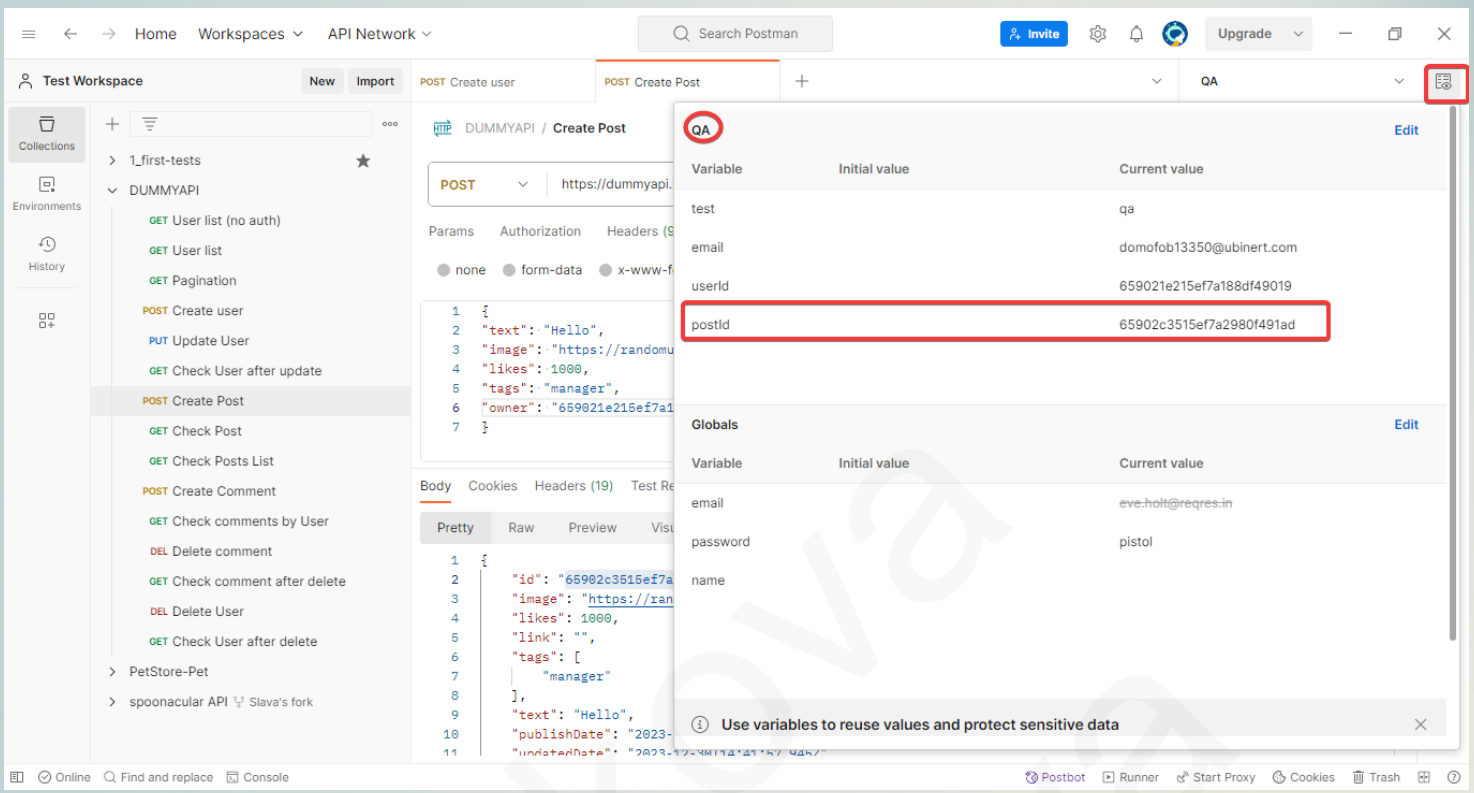


22.2. Paste the new user ID (659021e215ef7a188df49019) in the Body section and click Send.



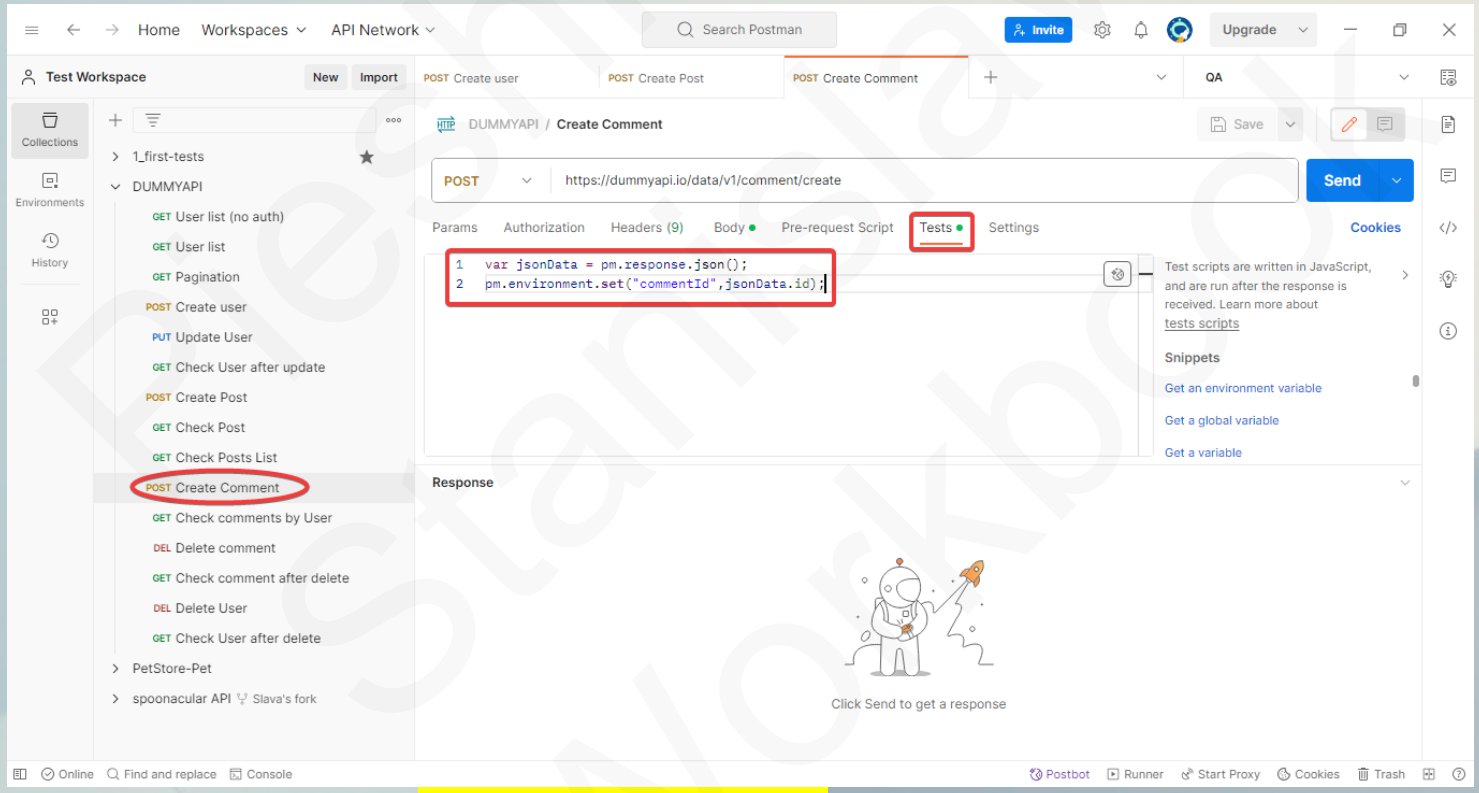
We get a new post ID: 65902c3515ef7a2980f491ad

22.3. Check if a new post ID has been added to the QA environment.

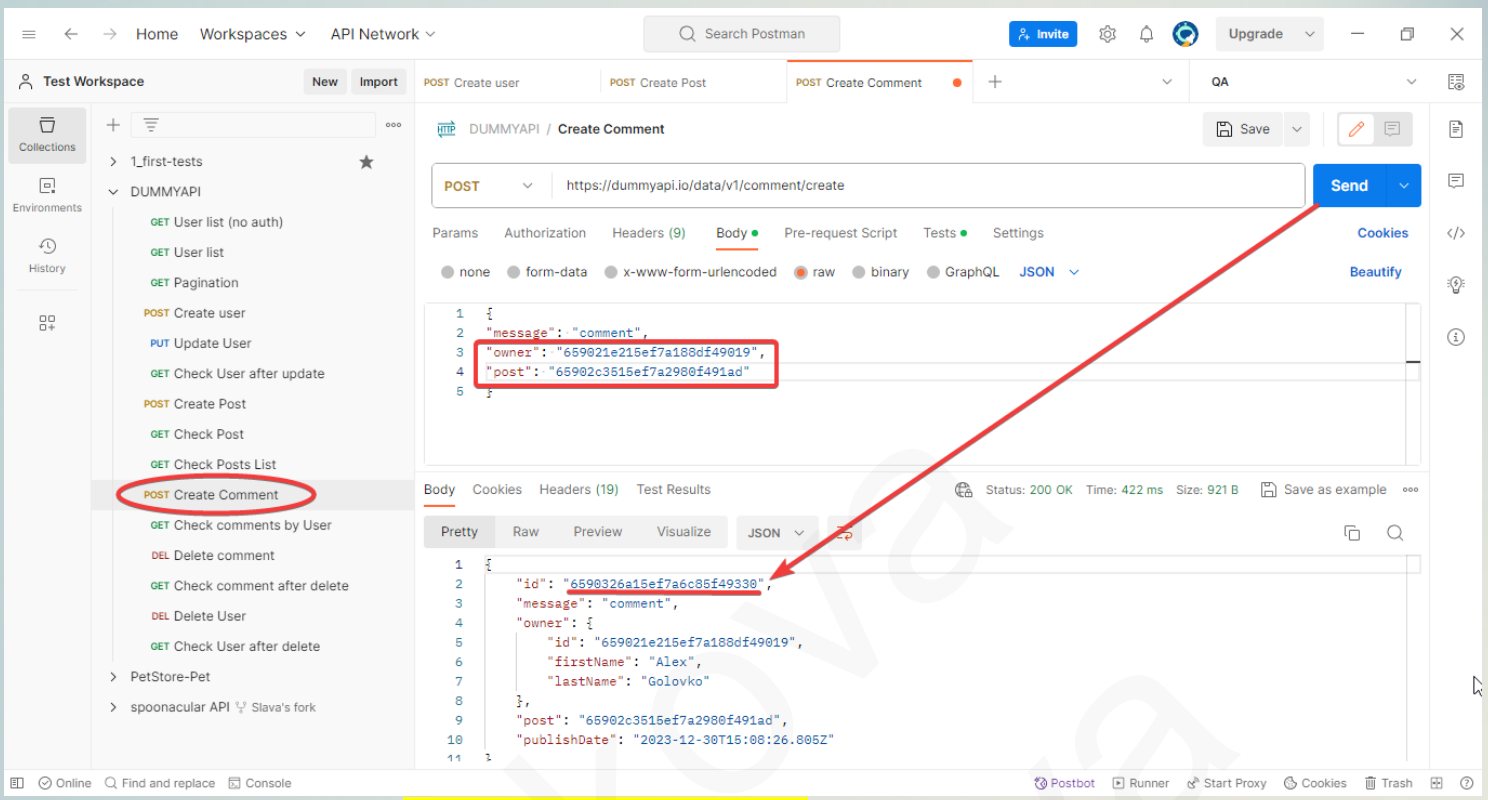


The new postId was successfully updated in the QA environment.

22.4. Create variables for a new comment.

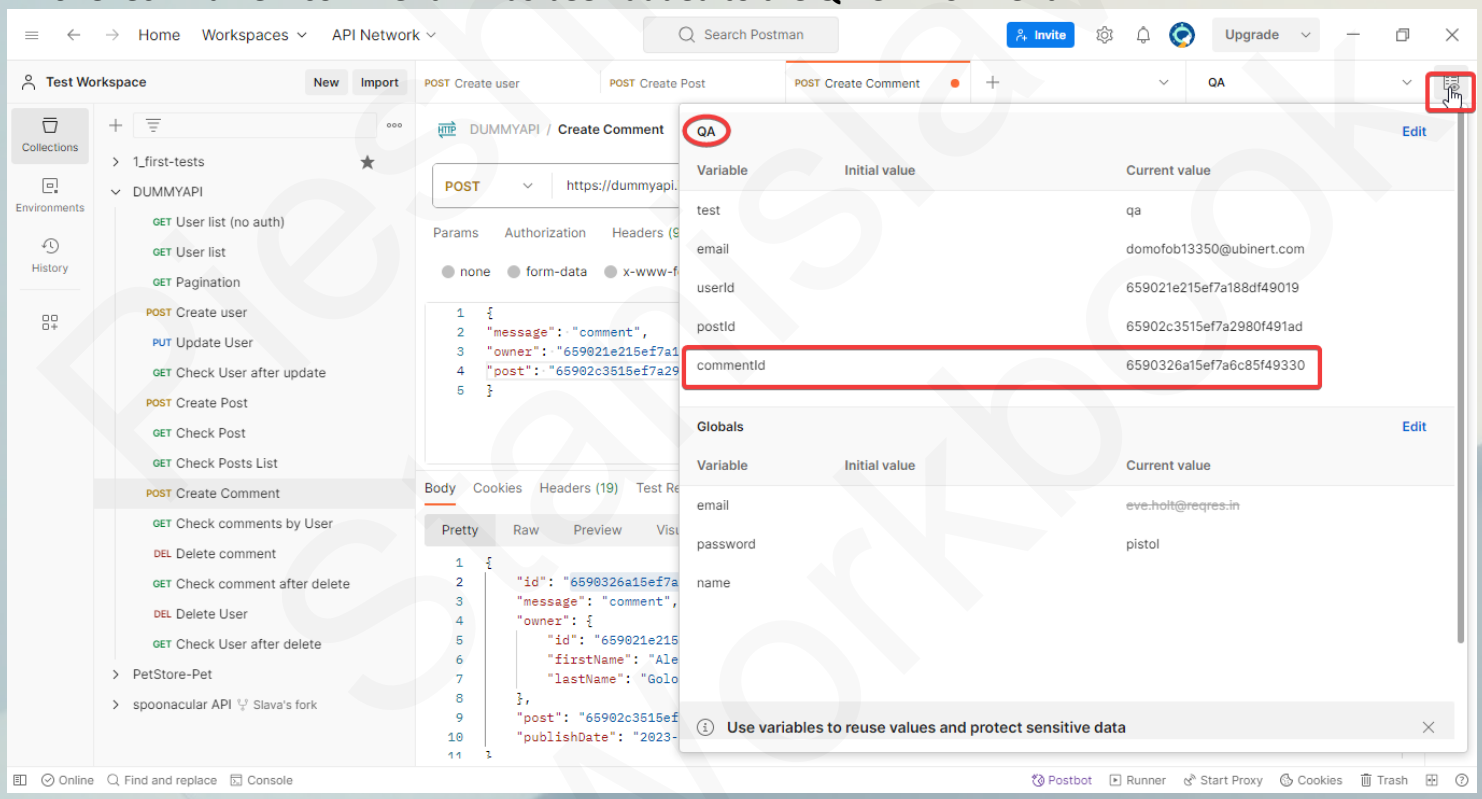


22.5. Paste the new user ID (659021e215ef7a188df49019) and new post ID (65902c3515ef7a2980f491ad) in the Body section and click Send.



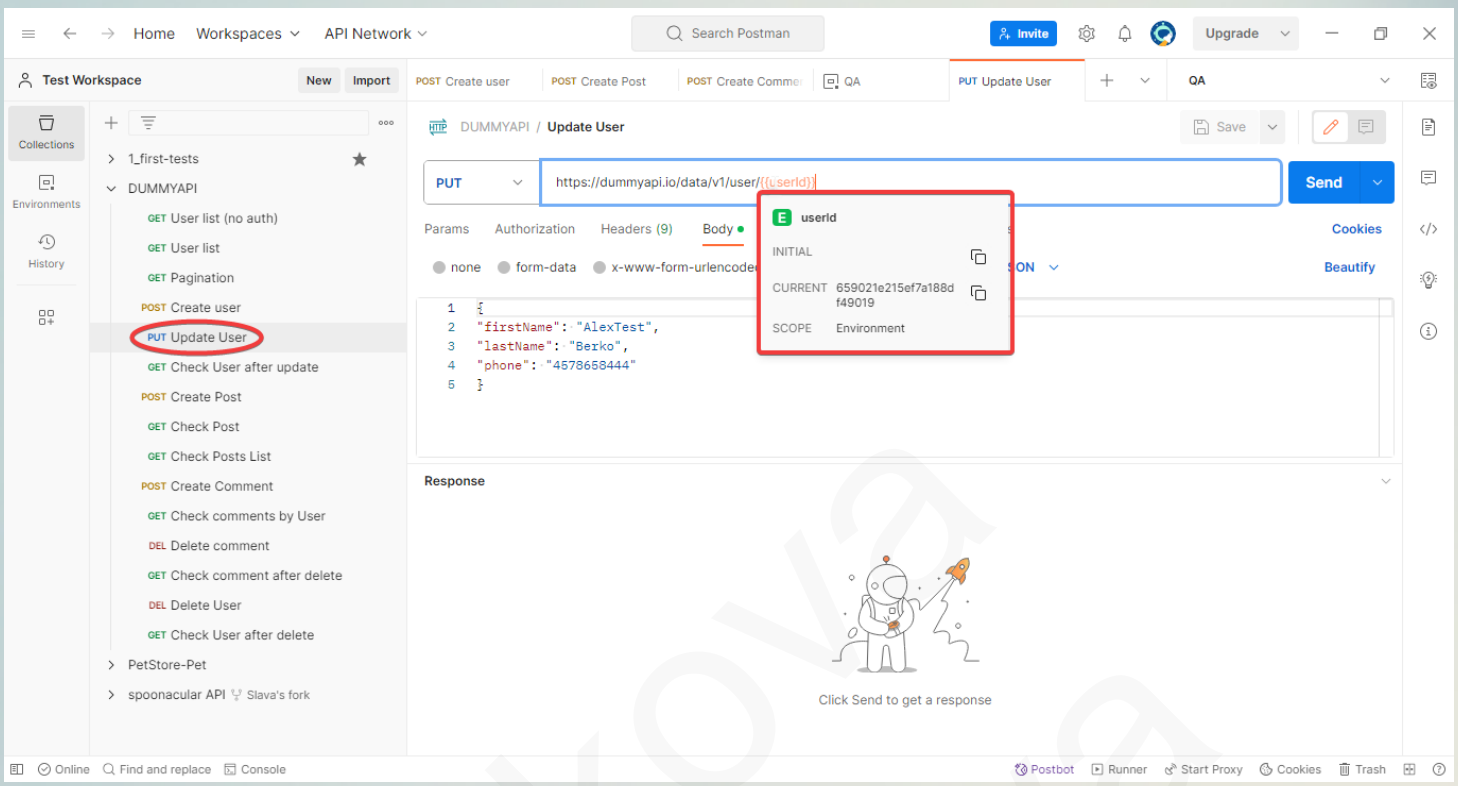
We get a new comment ID: 6590326a15ef7a6c85f49330

22.6. Check if a new comment ID has been added to the QA environment.

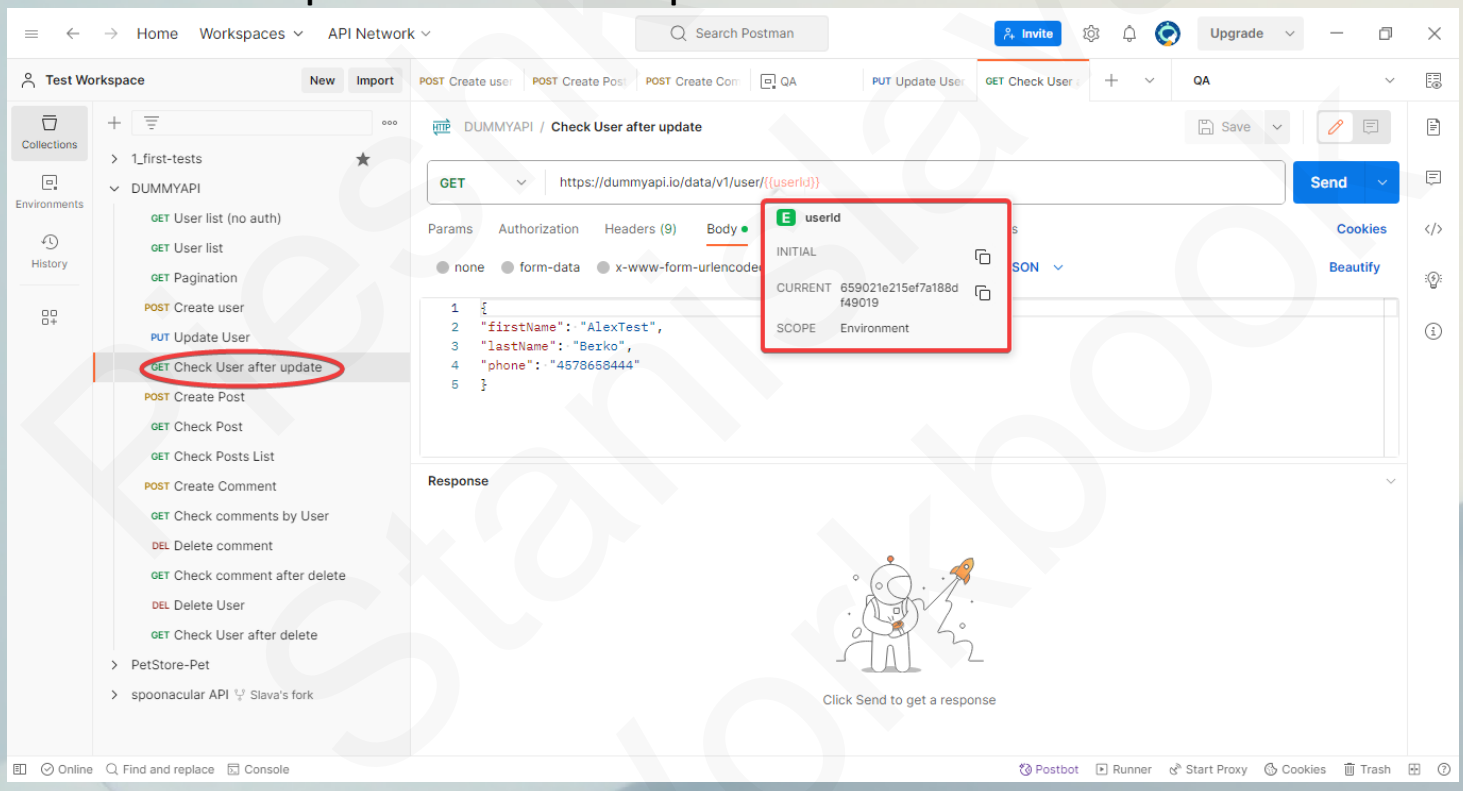


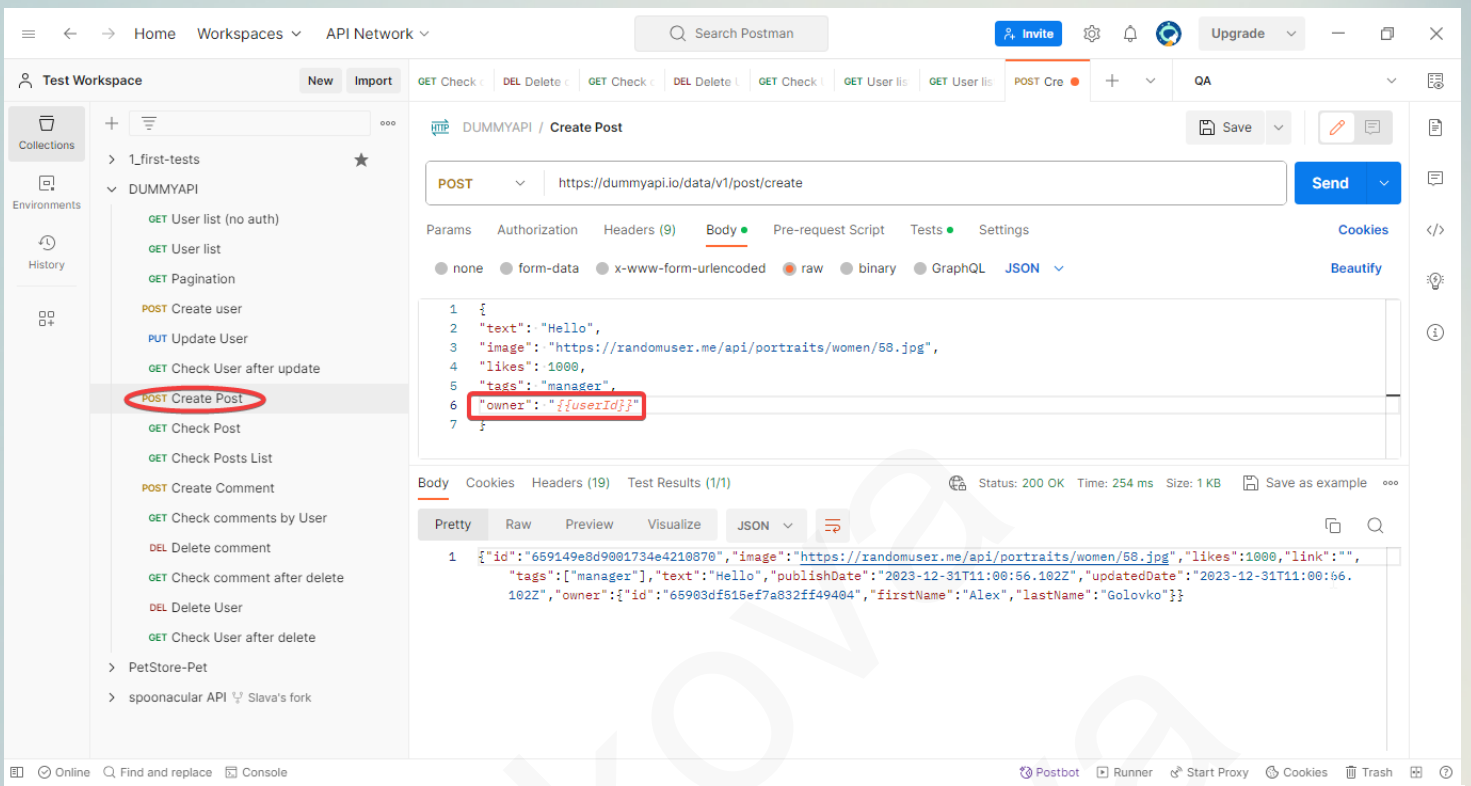
23. We created a collection with variables in the QA environment. Now the main task will be to go through the collection of requests and replace them with a variable format that will allow us to automate tests in the future. In order to no longer enter them manually in the future.

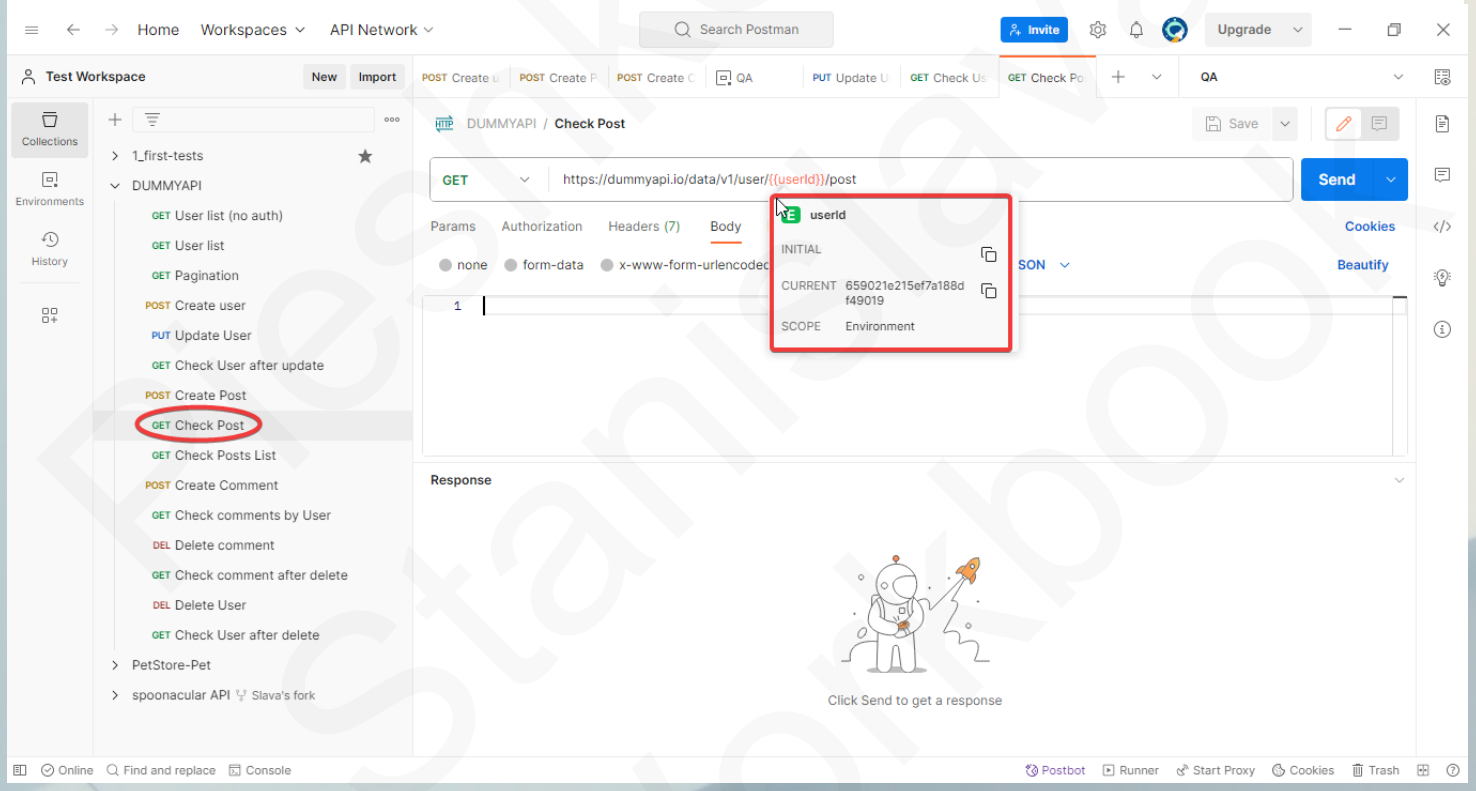
23.1. In the Put method, we entered the User ID manually. We need to change this.

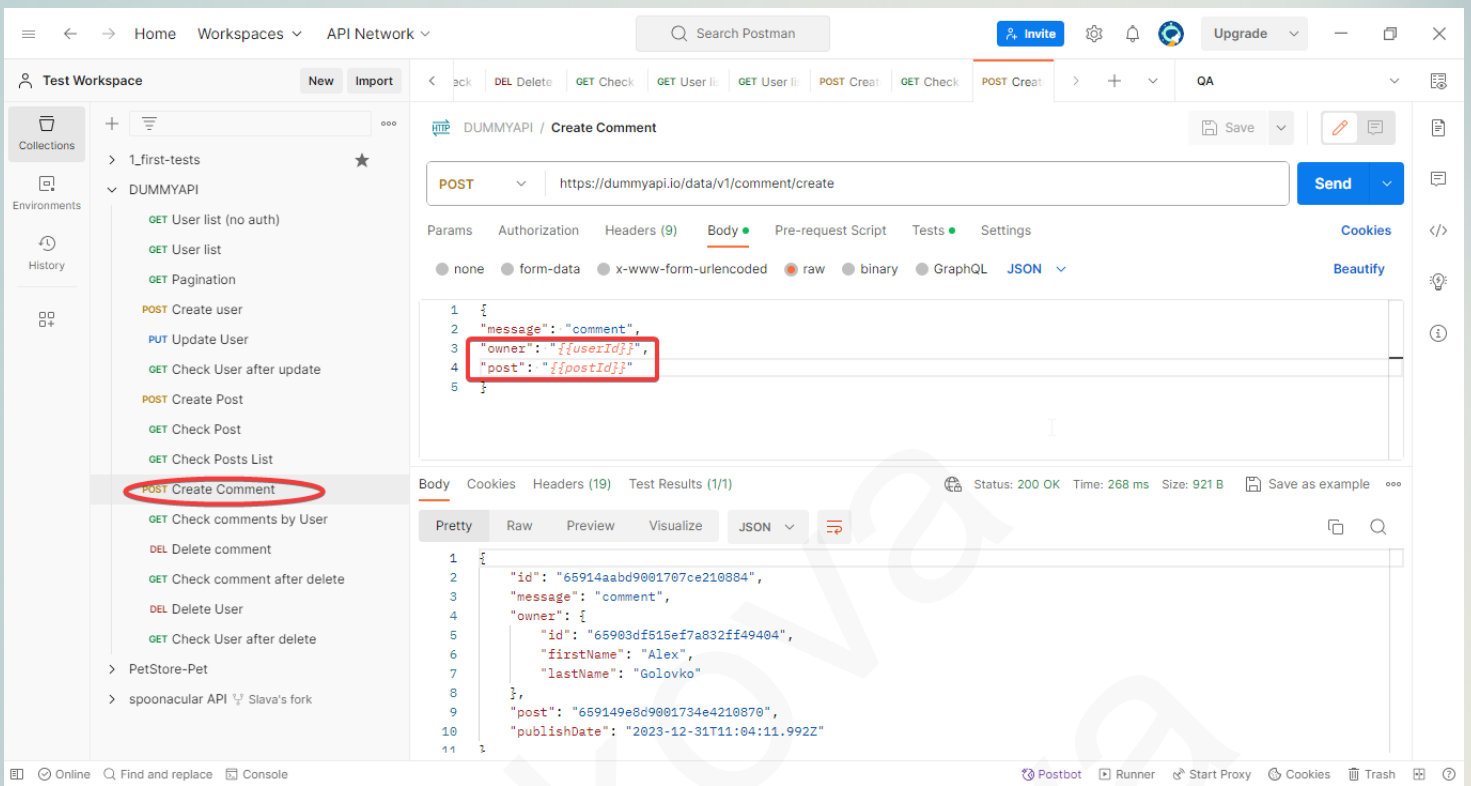


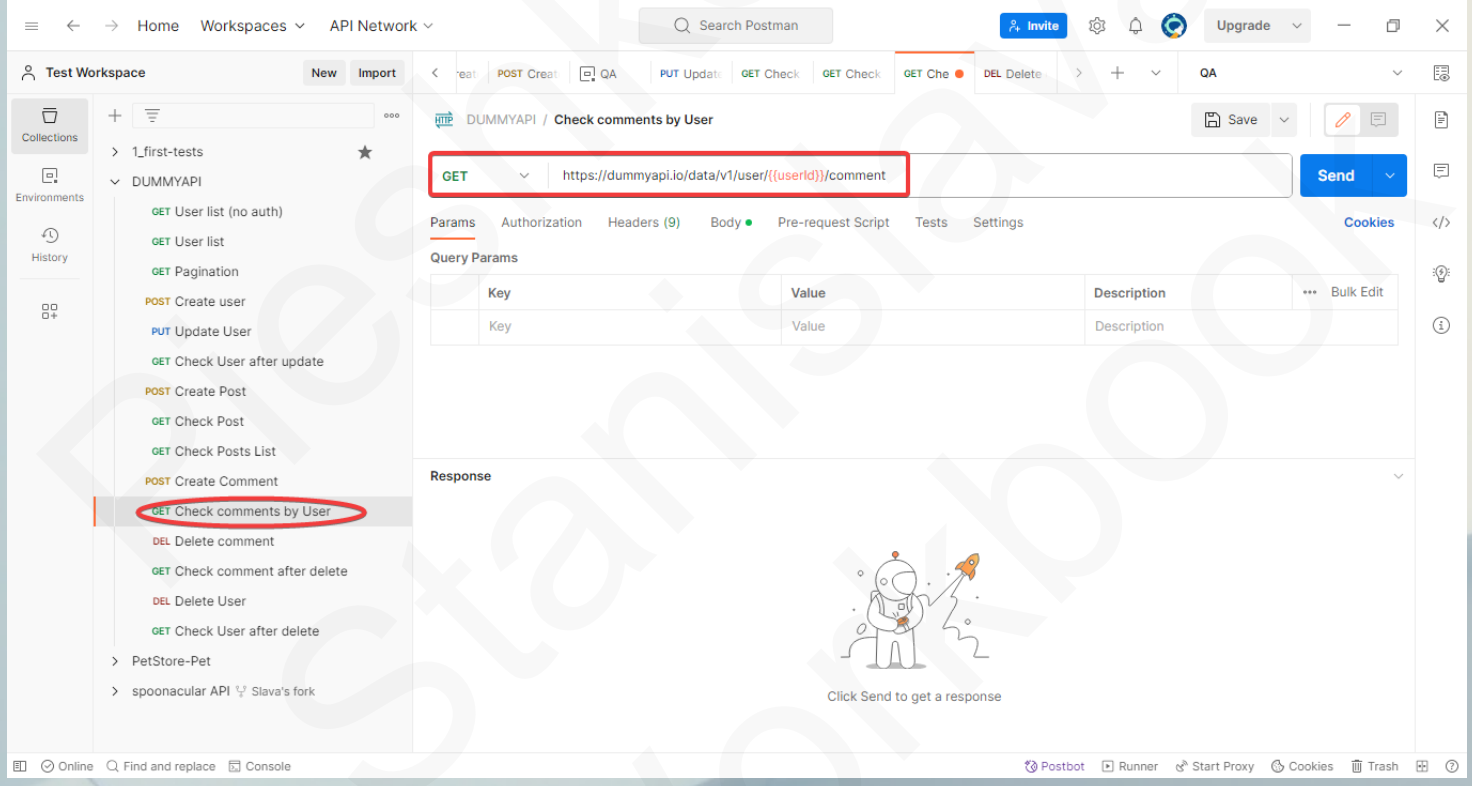
We make the same replacement in all other requests:




Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Test Workspace
New
Import
QA
Collections
Environments
History
1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API
Slava's fork
DUMMYAPI / Create Post
Save
POST
https://dummyapi.io/data/v1/post/create
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"text": "Hello",
"image": "https://randomuser.me/api/portraits/women/58.jpg",
"likes": 1000,
"tags": "manager",
"owner": "{{userId}}"
}
Body
Cookies
Headers (19)
Test Results (1/1)
Status: 200 OK
Time: 254 ms
Size: 1 KB
Save as example
Pretty
Raw
Preview
Visualize
JSON
{"id":"659149e8d9001734e4210870","image":"https://randomuser.me/api/portraits/women/58.jpg","likes":1000,"link":"",
"tags":["manager"],"text":"Hello","publishDate":"2023-12-31T11:00:56.102Z","updatedDate":"2023-12-31T11:00:56.
102Z","owner":{"id":"65903df515ef7a832ff49404","firstName":"Alex","lastName":"Golovko"}}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Test Workspace
New
Import
QA
Collections
Environments
History
1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API
Slava's fork
DUMMYAPI / Check Post
Save
GET
https://dummyapi.io/data/v1/user/{{userId}}/post
Send
Params
Authorization
Headers (7)
Body
Cookies
none
form-data
x-www-form-urlencoded
JSON
Beautify
userId
INITIAL
CURRENT 659021e215ef7a188df49019
SCOPE Environment
Response
Click Send to get a response
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Test Workspace
New
Import
Collections
Environments
History
1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API Slava's fork
QA
DUMMYAPI / Create Comment
Save
POST
https://dummyapi.io/data/v1/comment/create
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"message": "comment",
"owner": "{{userId}}",
"post": "{{postId}}"
}
Body
Cookies
Headers (19)
Test Results (1/1)
Status: 200 OK
Time: 268 ms
Size: 921 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": "65914aabd9001707ce210884",
"message": "comment",
"owner": {
"id": "65903df515ef7a832ff49404",
"firstName": "Alex",
"lastName": "Golovko"
},
"post": "659149e8d9001734e4210870",
"publishDate": "2023-12-31T11:04:11.992Z"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Test Workspace
New
Import
Collections
Environments
History
1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API Slava's fork
QA
DUMMYAPI / Check comments by User
Save
GET
https://dummyapi.io/data/v1/user/{{userId}}/comment
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Response
Click Send to get a response
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

Home Workspaces API Network Search Postman Invite Upgrade

Test Workspace New Import

Collections Environments History

1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API Slava's fork

DUMMYAPI / **Delete comment**

Save

DELETE https://dummyapi.io/data/v1/comment/{{commentId}} Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Response

Click Send to get a response

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

Home Workspaces API Network Search Postman Invite Upgrade

Test Workspace New Import

Collections Environments History

1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API Slava's fork

DUMMYAPI / **Check comment after delete**

Save

GET https://dummyapi.io/data/v1/user/{{userId}}/comment Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

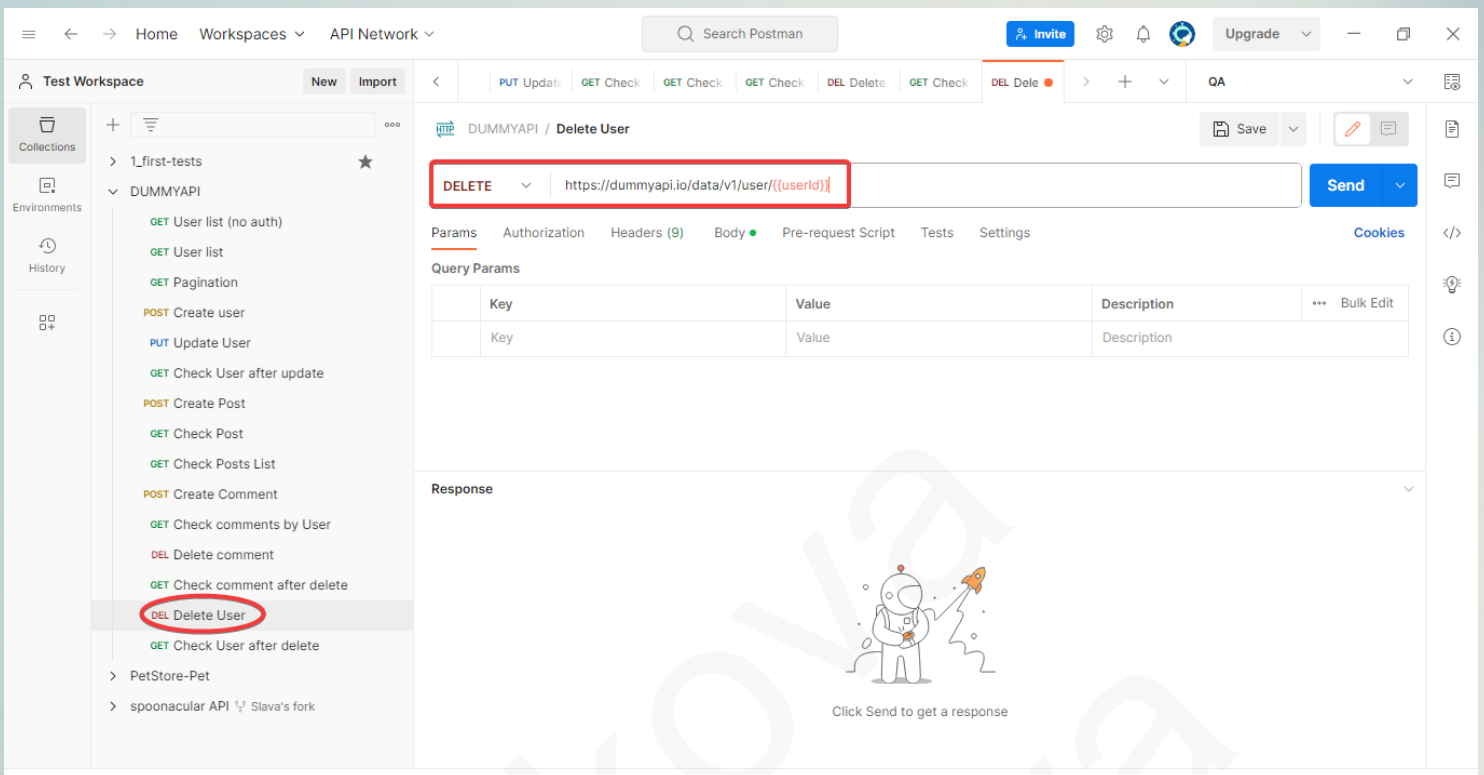
Query Params

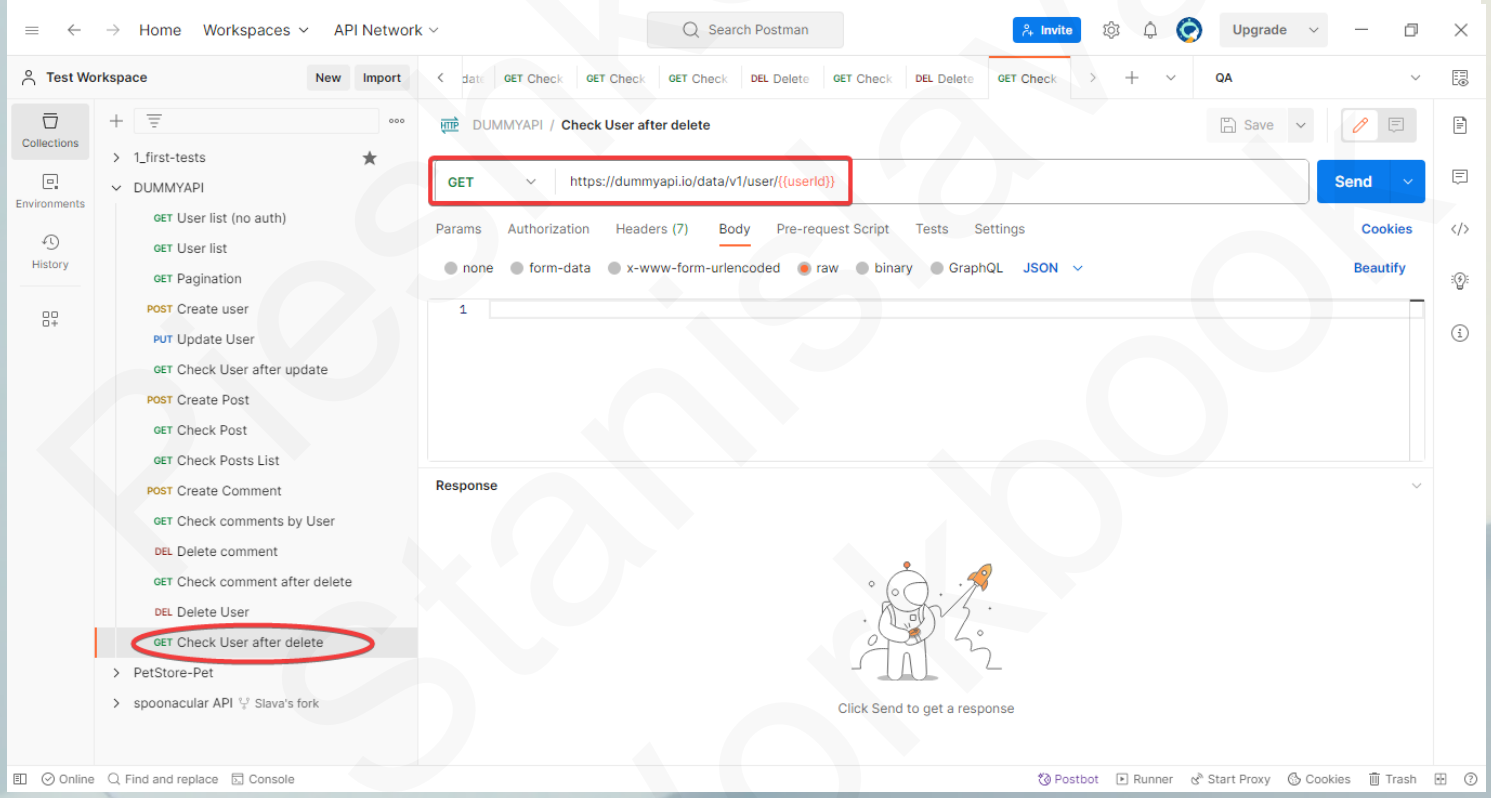
Key	Value	Description	Bulk Edit
Key	Value	Description	

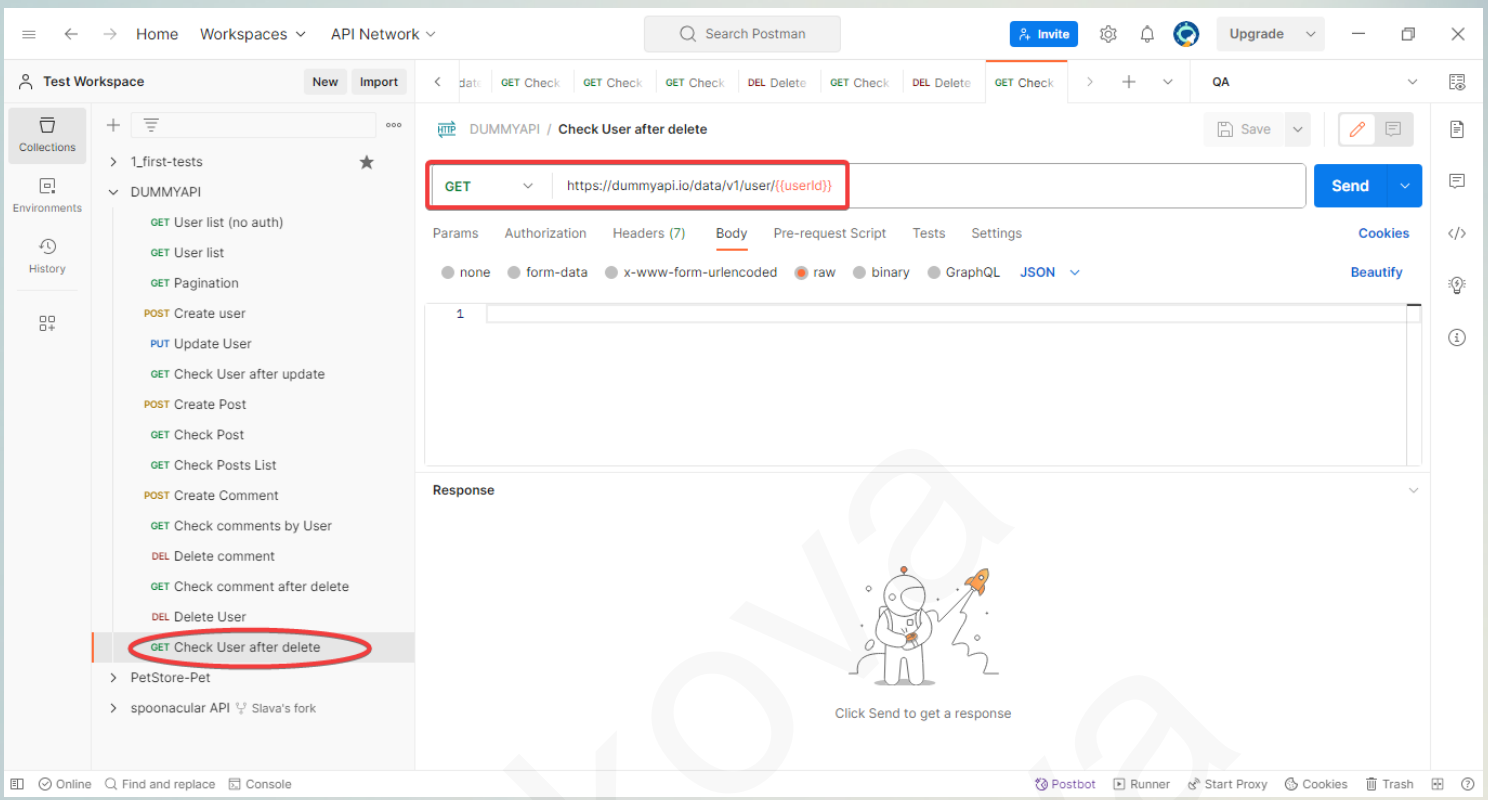
Response

Click Send to get a response

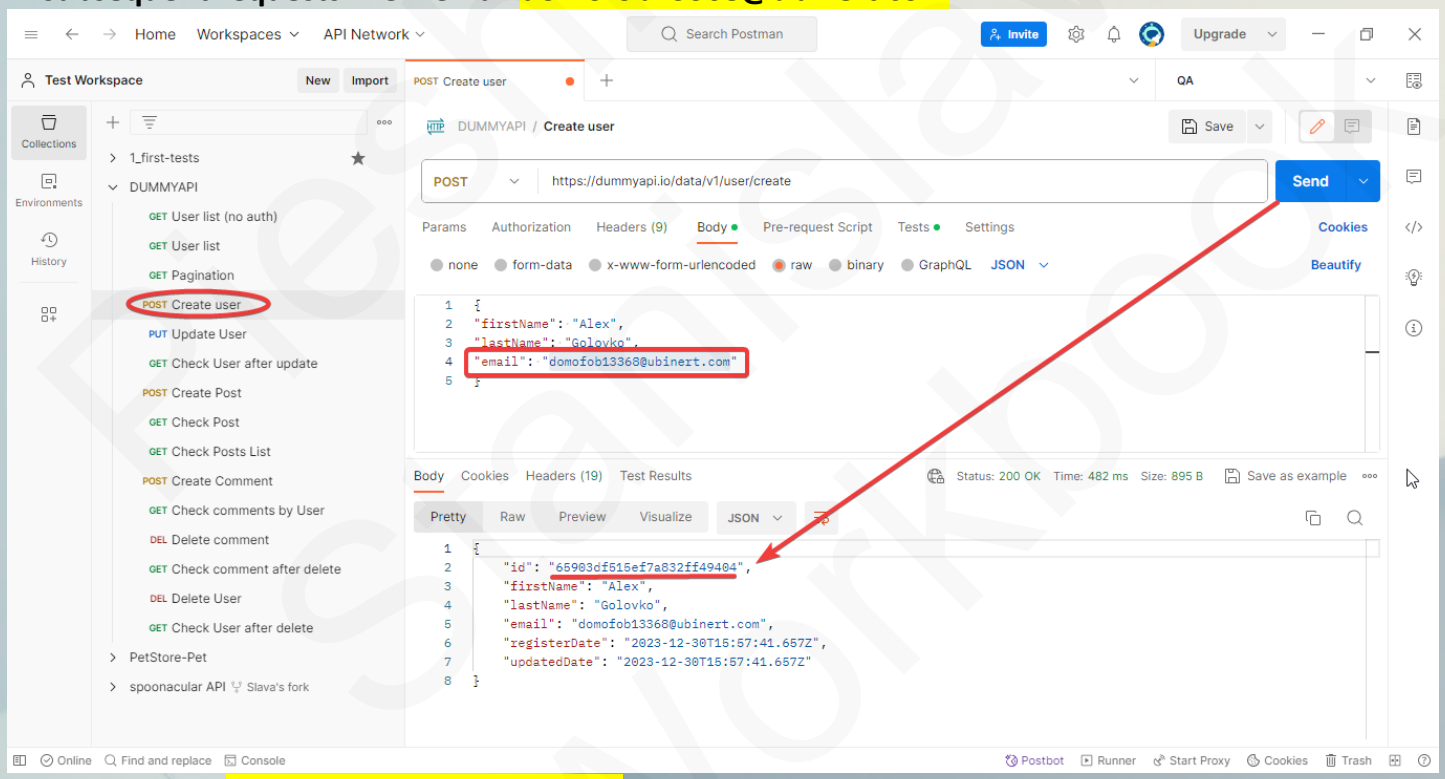
Online Find and replace Console Postbot Runner Start Proxy Cookies Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Test Workspace
New
Import
QA
Collections
Environments
History
1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API Slava's fork
DUMMYAPI / Delete User
Save
DELETE
https://dummyapi.io/data/v1/user/{{userId}}
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Response
Click Send to get a response
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

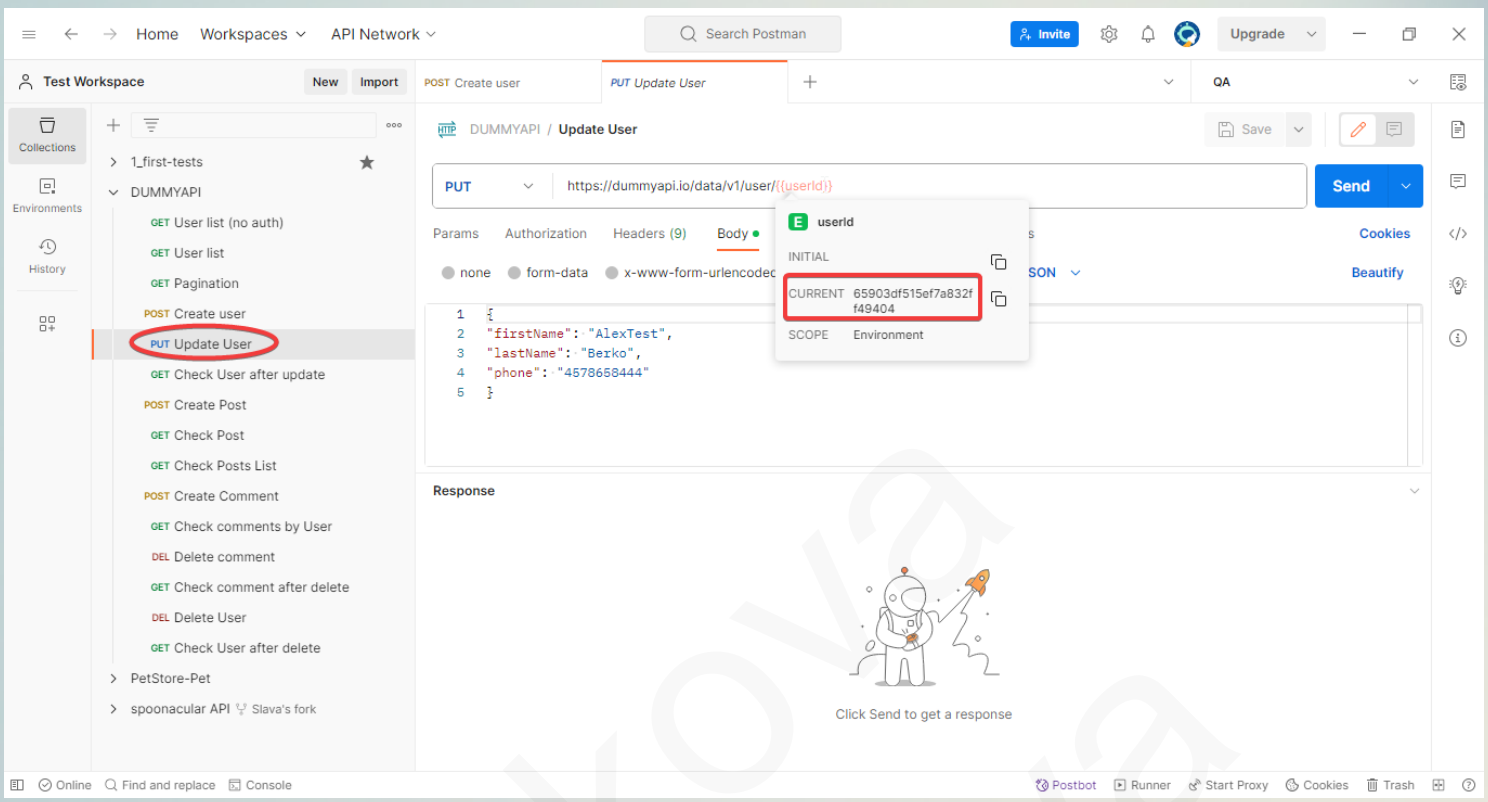

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Test Workspace
New
Import
QA
Collections
Environments
History
1_first-tests
DUMMYAPI
GET User list (no auth)
GET User list
GET Pagination
POST Create user
PUT Update User
GET Check User after update
POST Create Post
GET Check Post
GET Check Posts List
POST Create Comment
GET Check comments by User
DEL Delete comment
GET Check comment after delete
DEL Delete User
GET Check User after delete
PetStore-Pet
spoonacular API Slava's fork
DUMMYAPI / Check User after delete
Save
GET
https://dummyapi.io/data/v1/user/{{userId}}
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Response
Click Send to get a response
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash



24. Now we check whether, for example, when creating a new User, its new ID automatically appears in subsequent requests. New email domofob13368@ubinert.com



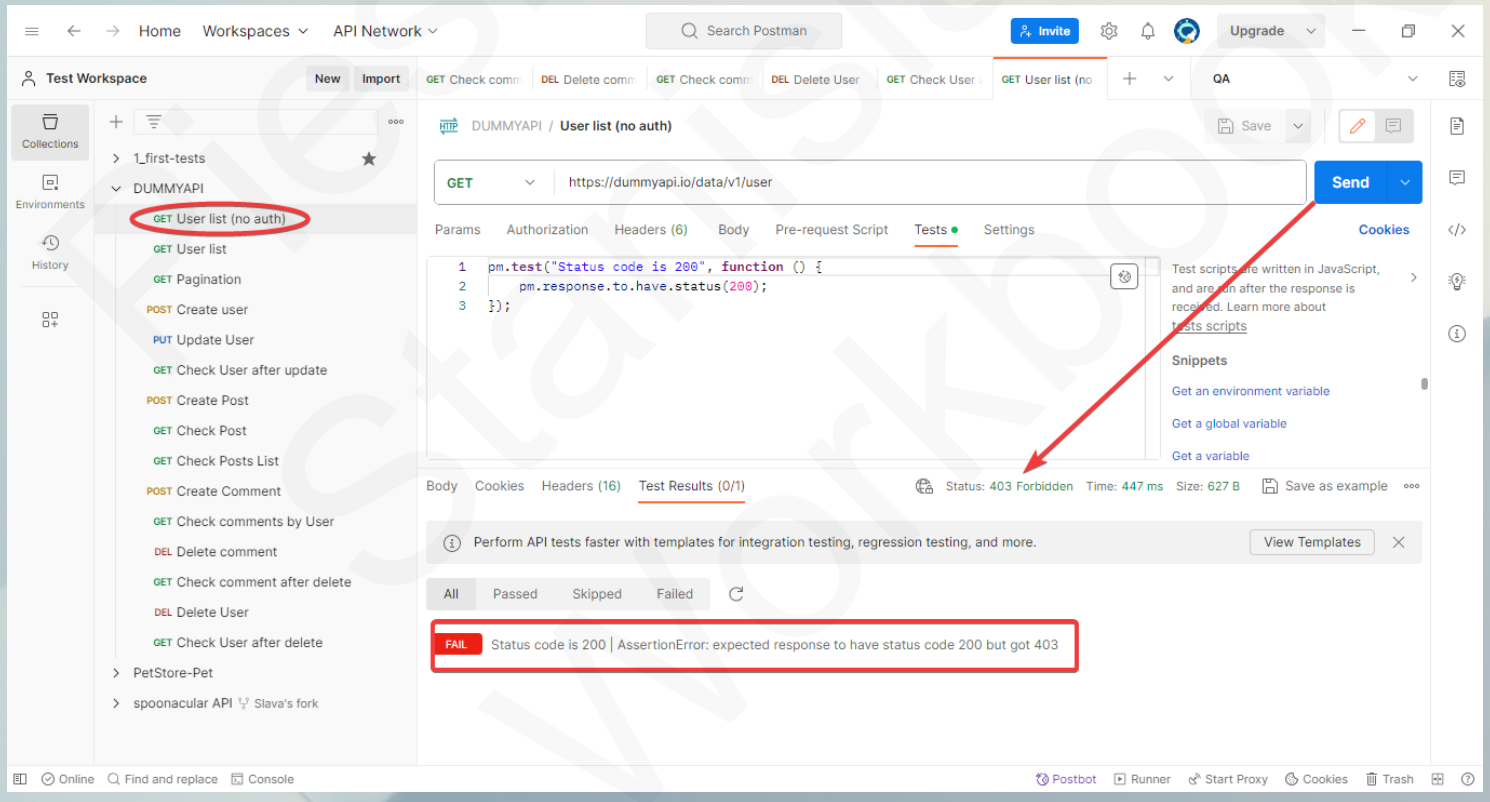
A truly new Id: 65903df515ef7a832ff49404 was automatically pulled up in the next request:



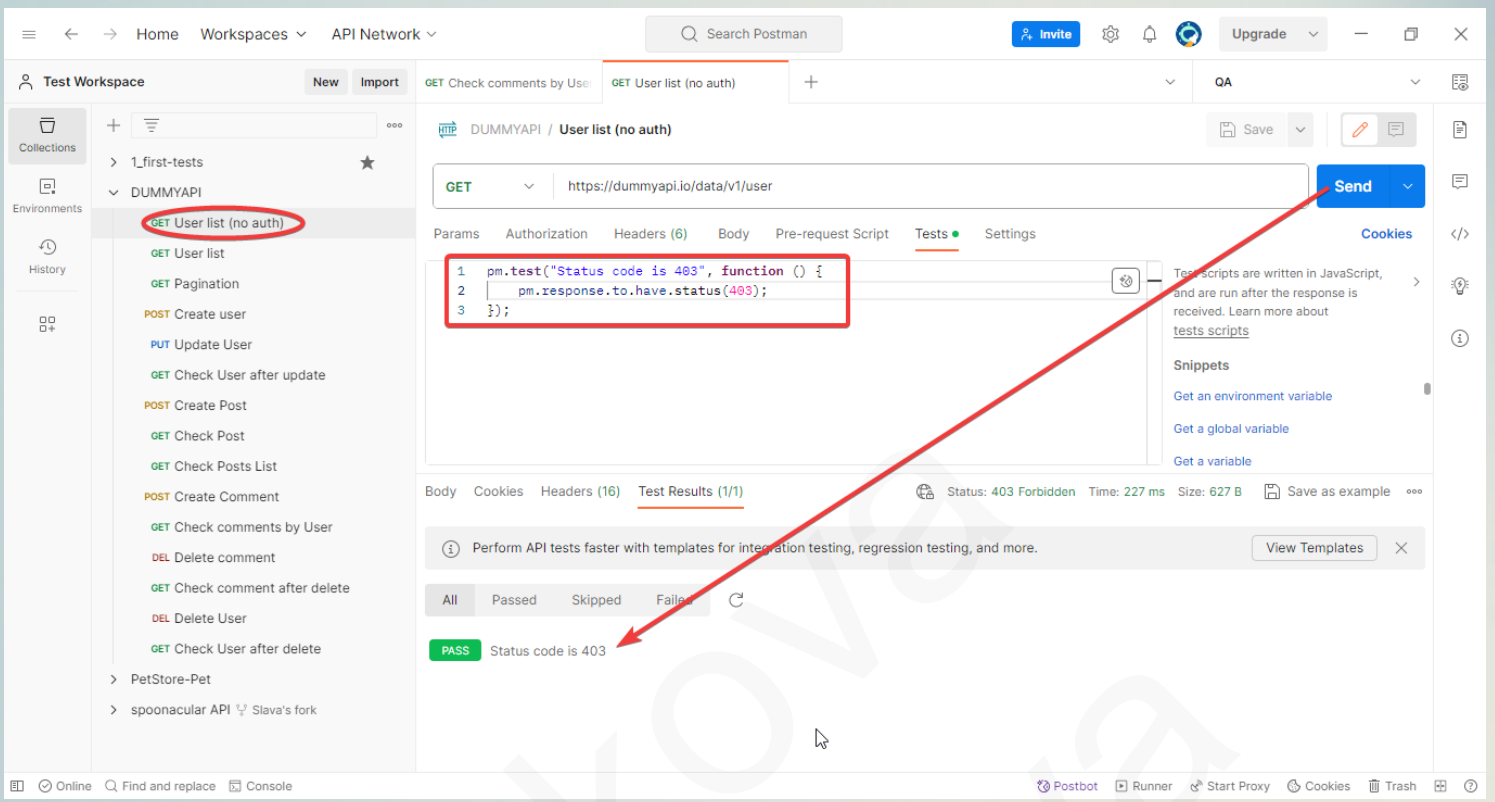
25. Create tests for these requests.

25.1. The first test is that the expected status code is 200. Applies to all tests.

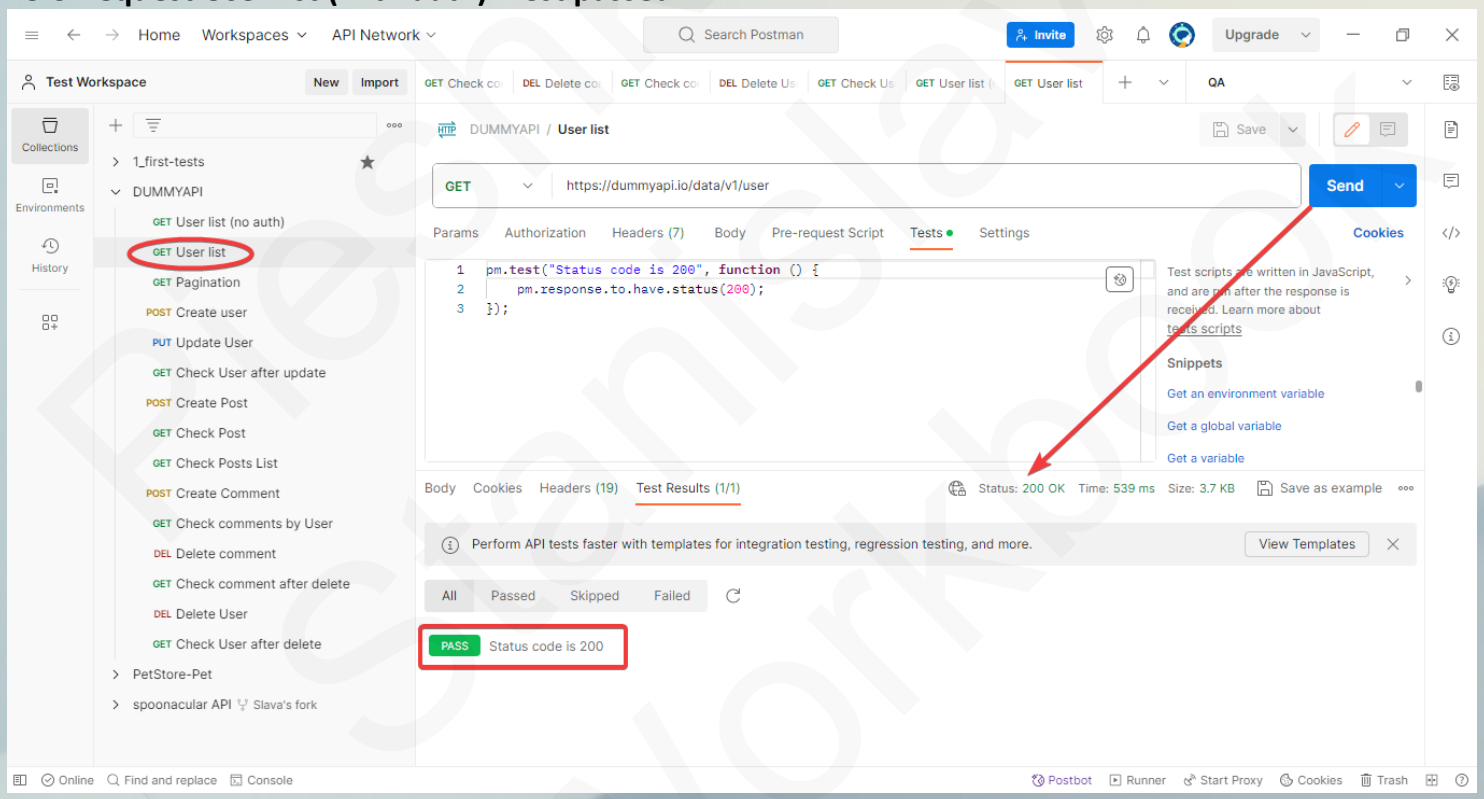
25.2.1. Request User list (no auth). The first test failed. It is expected that the status code is 403, because the user is not authorized.



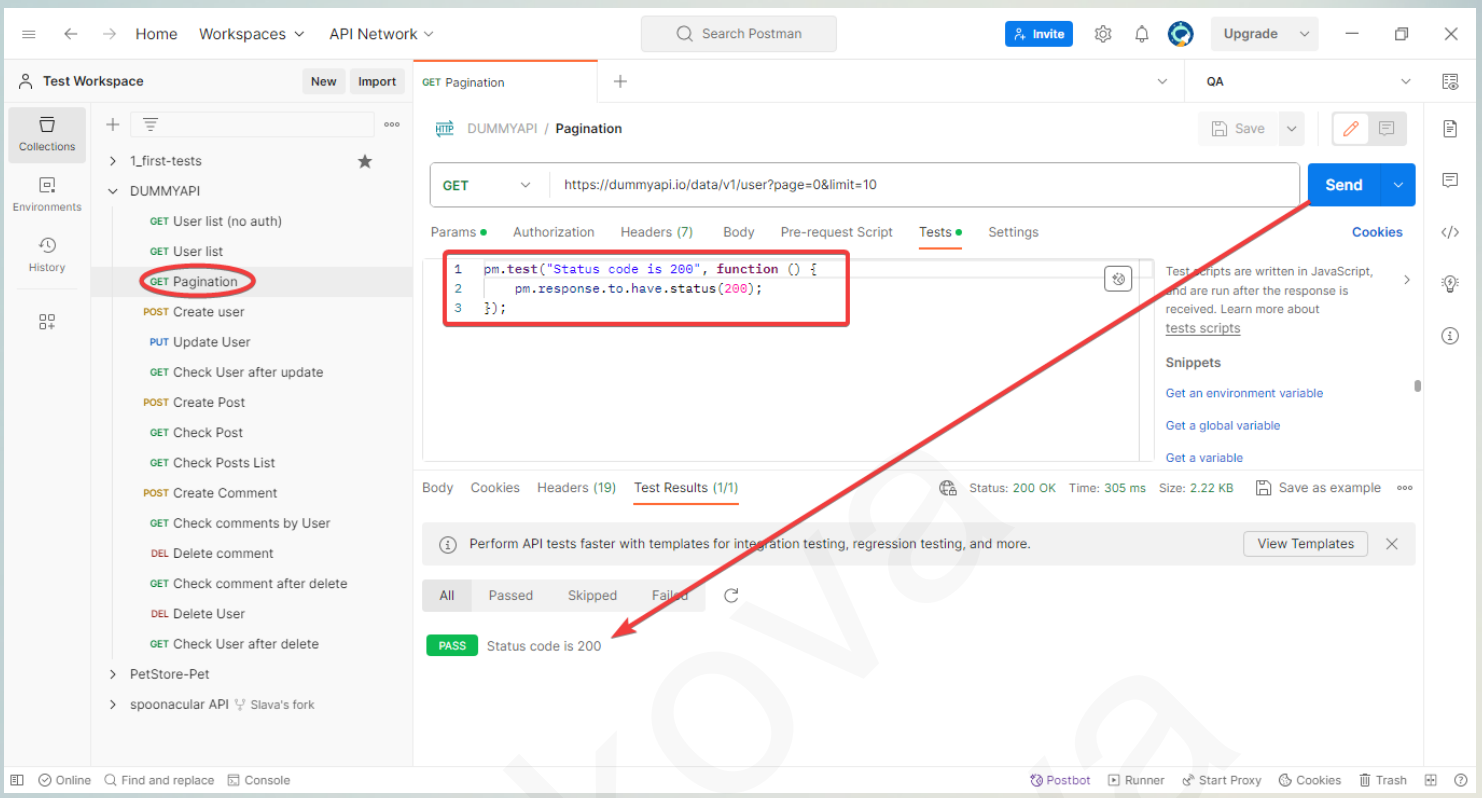
25.2.2. Request User list (no auth). The second test passed. It is expected that the status code is 403, because the user is not authorized. We expect the status code to be 403 and get the same response.



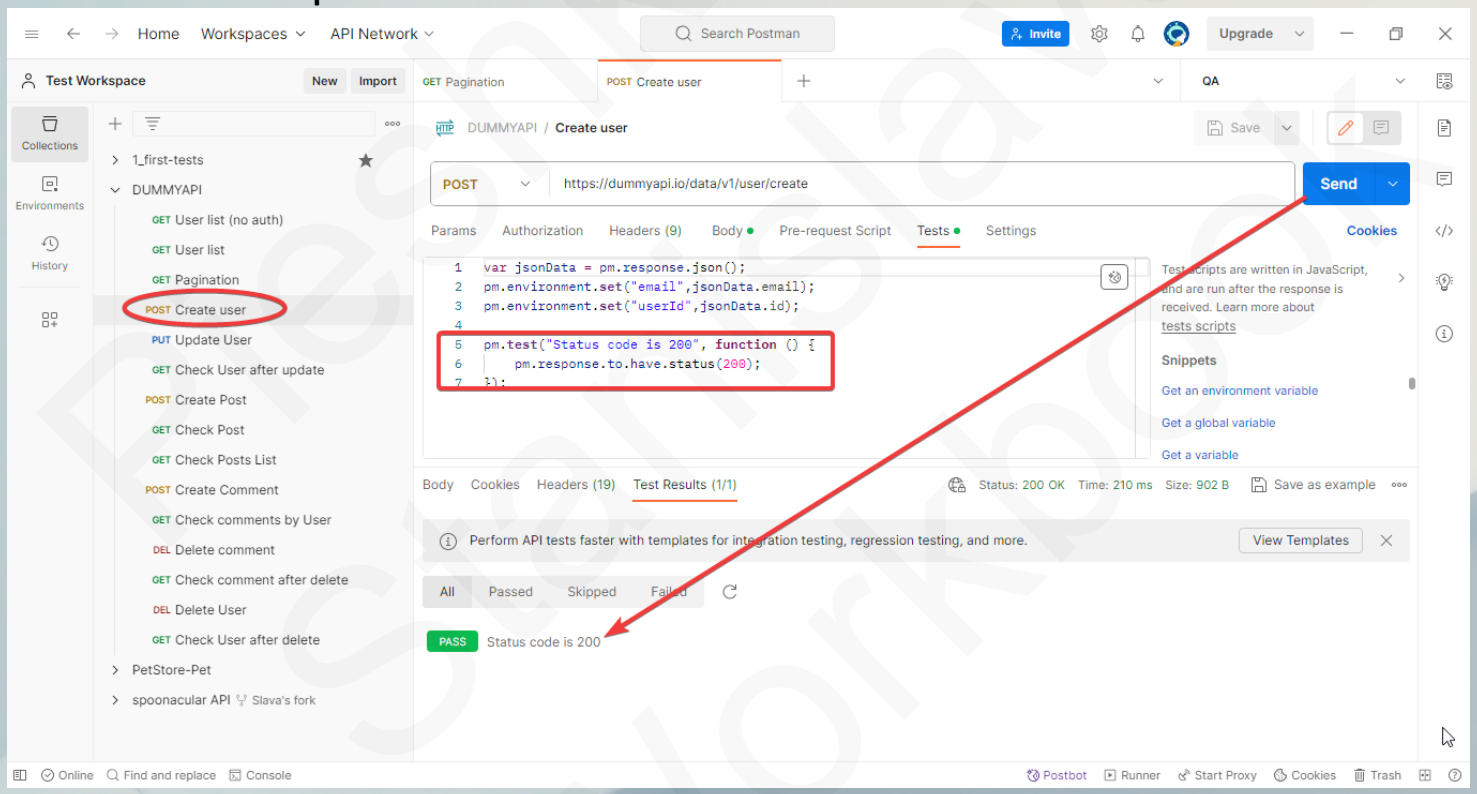
25.3 Request User list (with auth). Test passed.



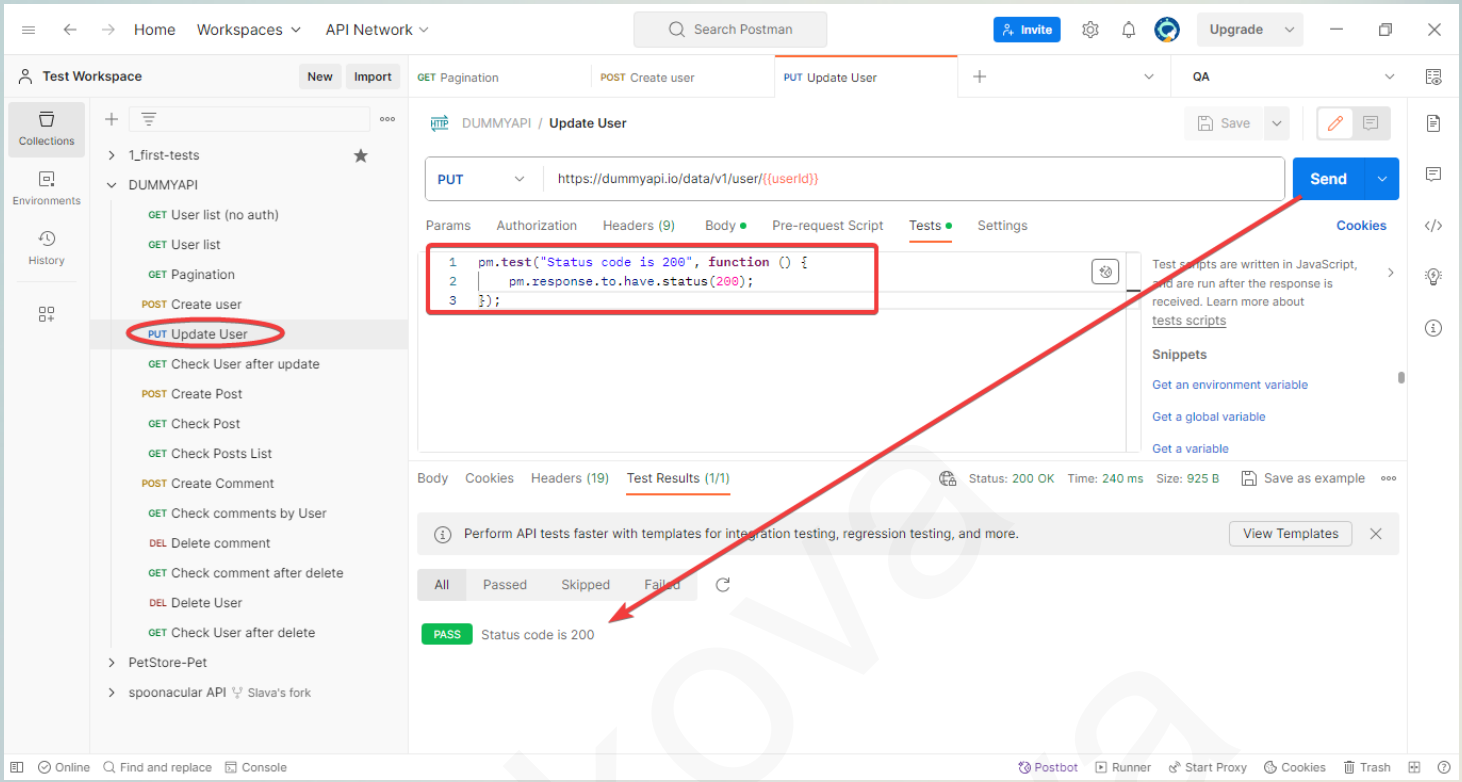
25.4 Pagination test passed.



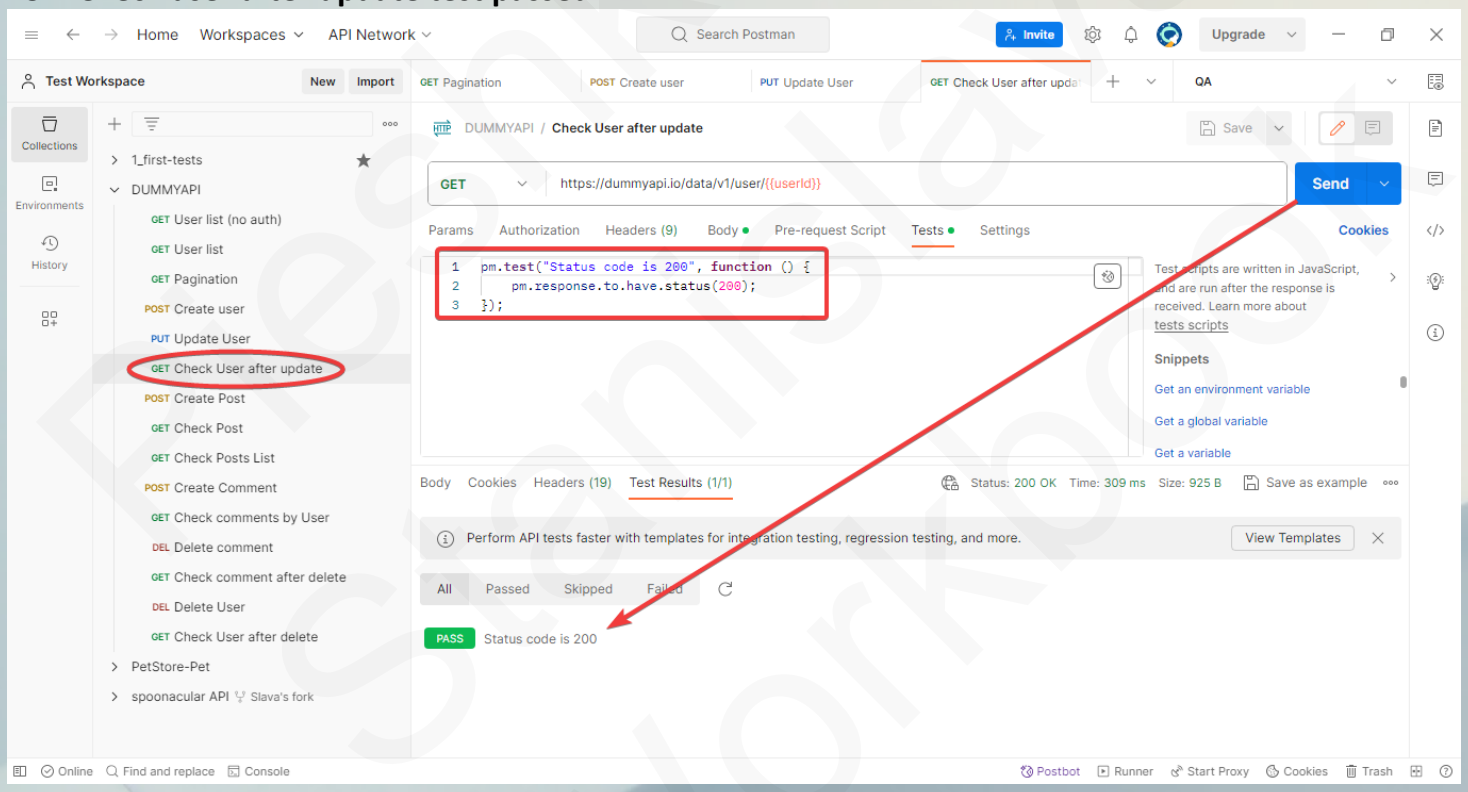
25.5 Create user test passed.



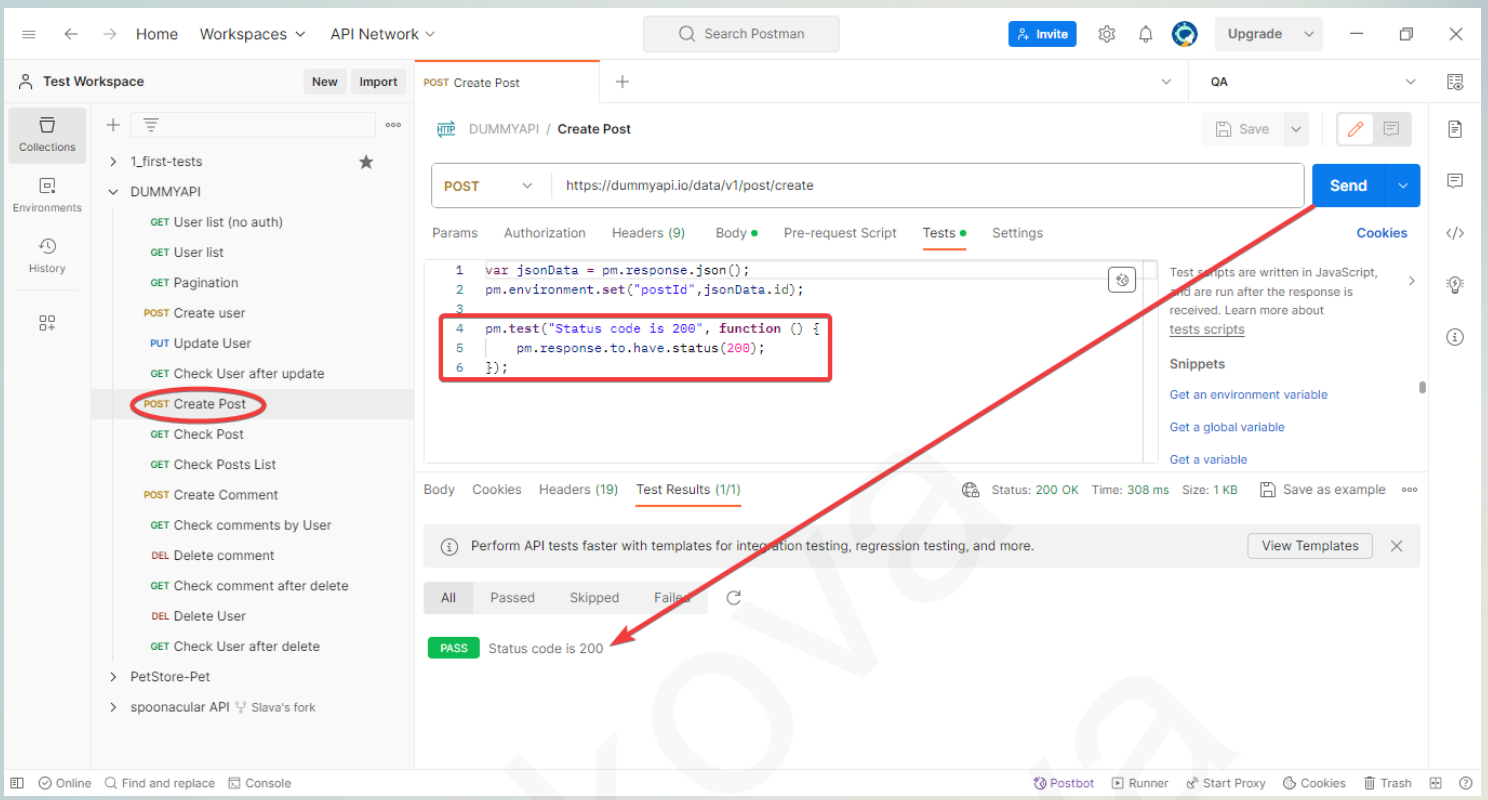
25.6 Update user test passed.



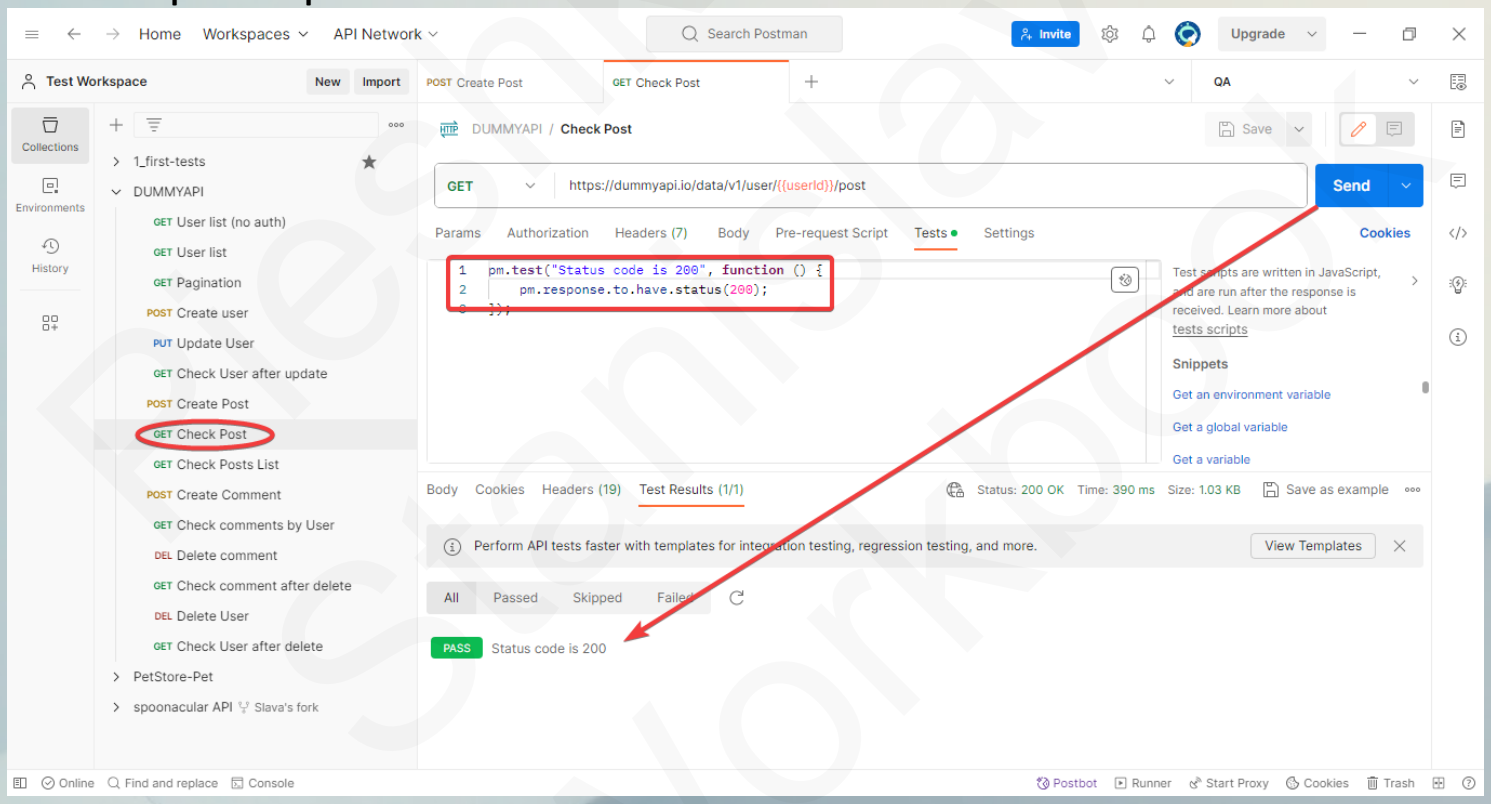
25.7 Check user after update test passed.



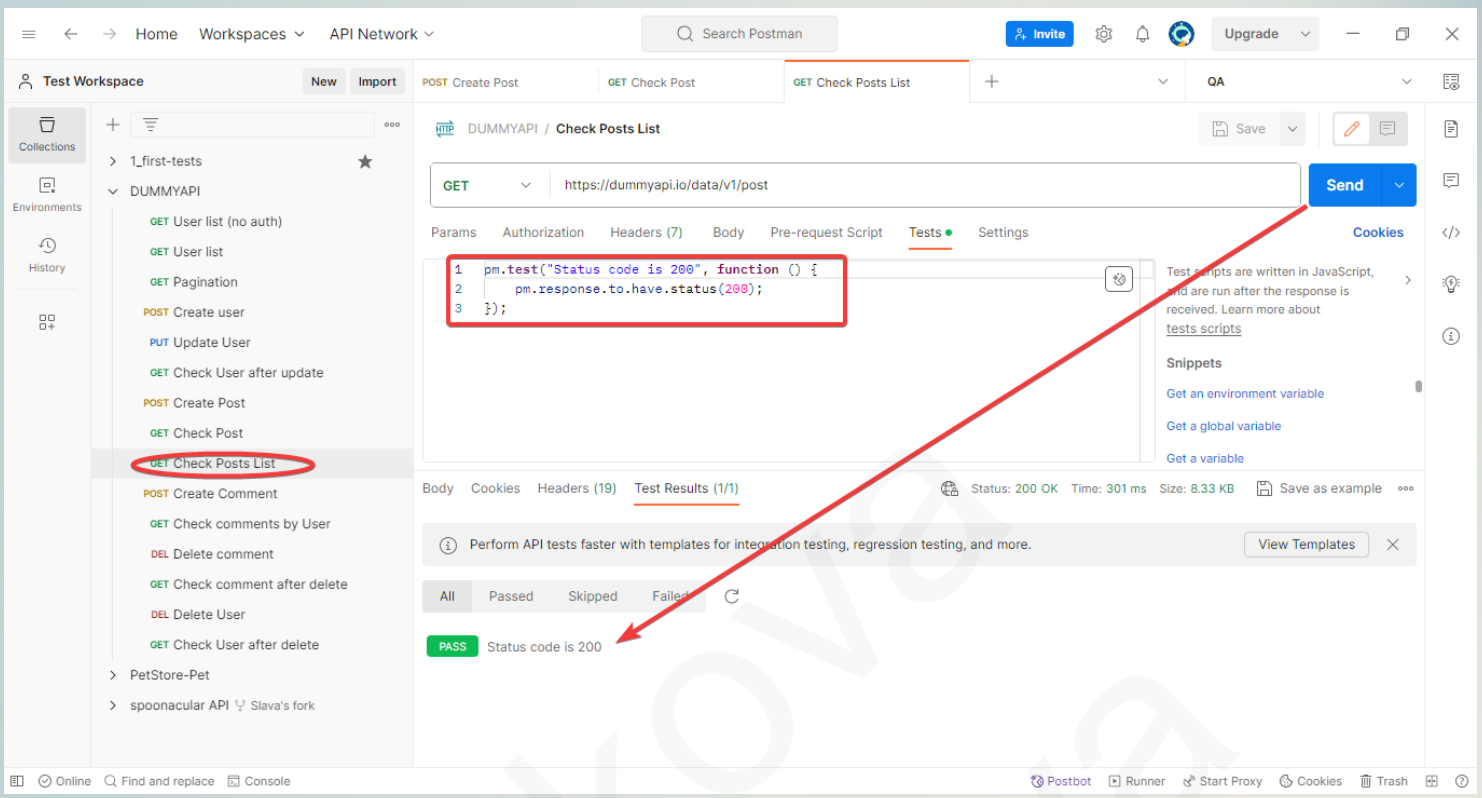
25.8 Create post test passed.



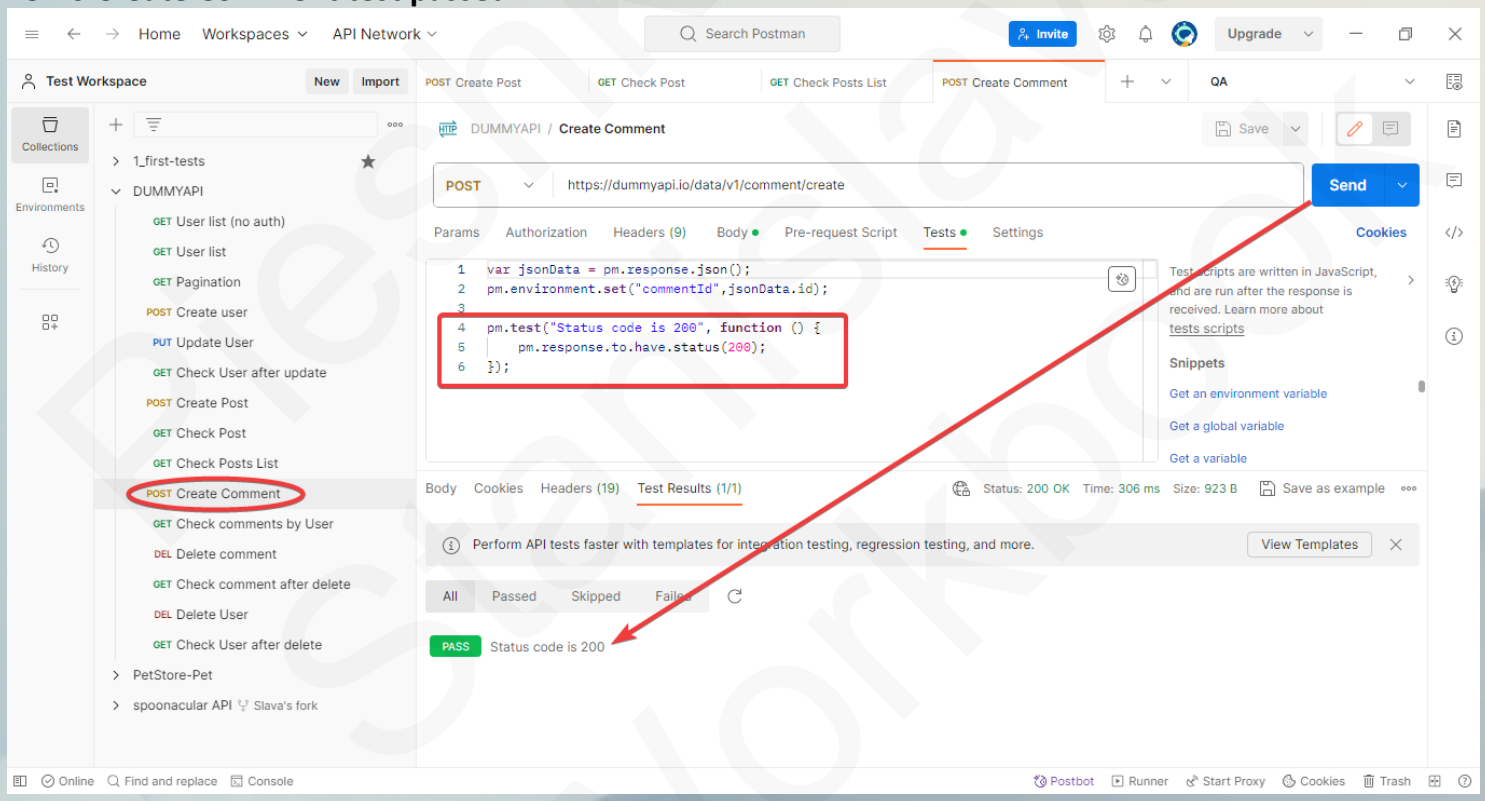
25.8 Check post test passed.



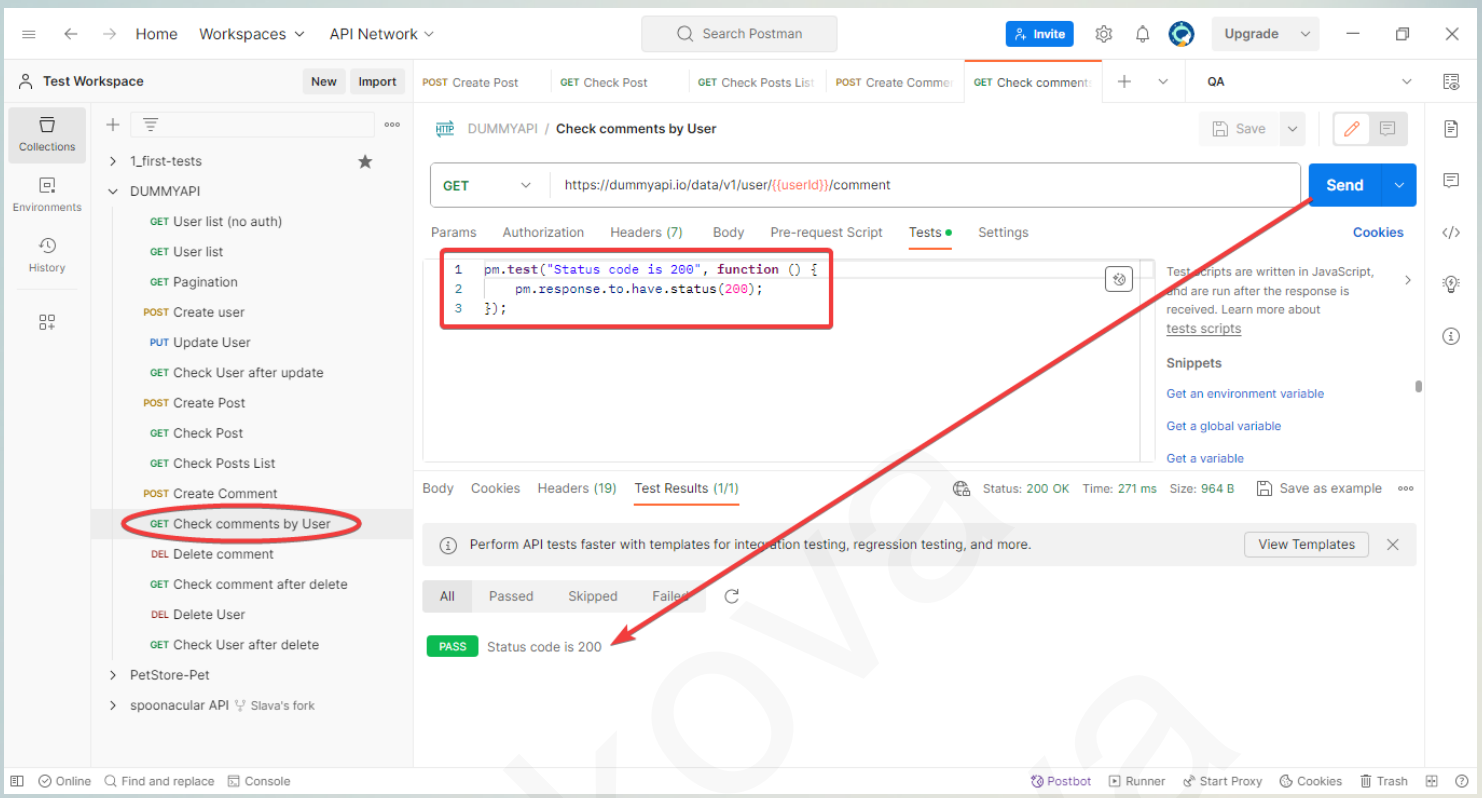
25.9 Check posts list test passed.



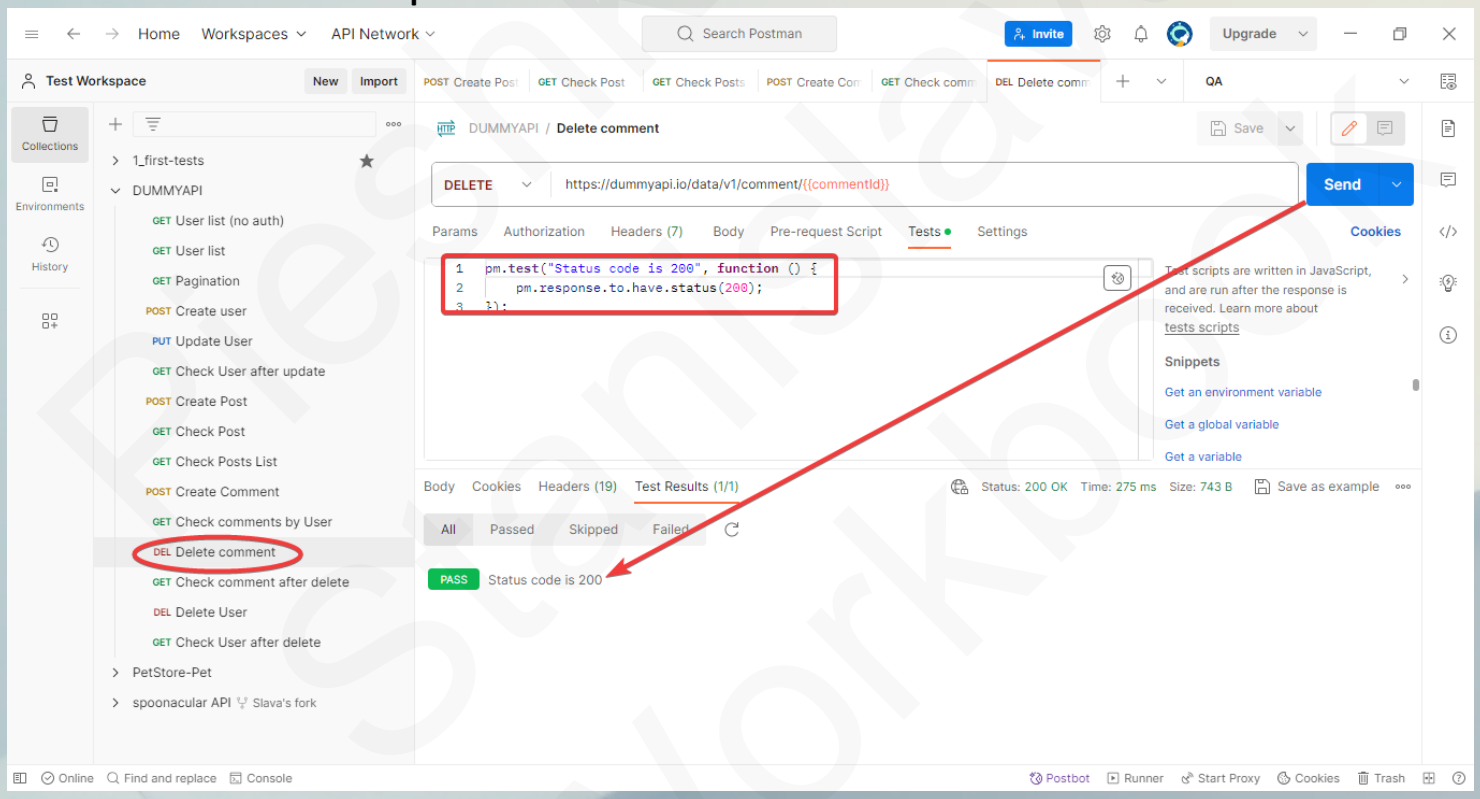
25.10 Create Comment test passed.



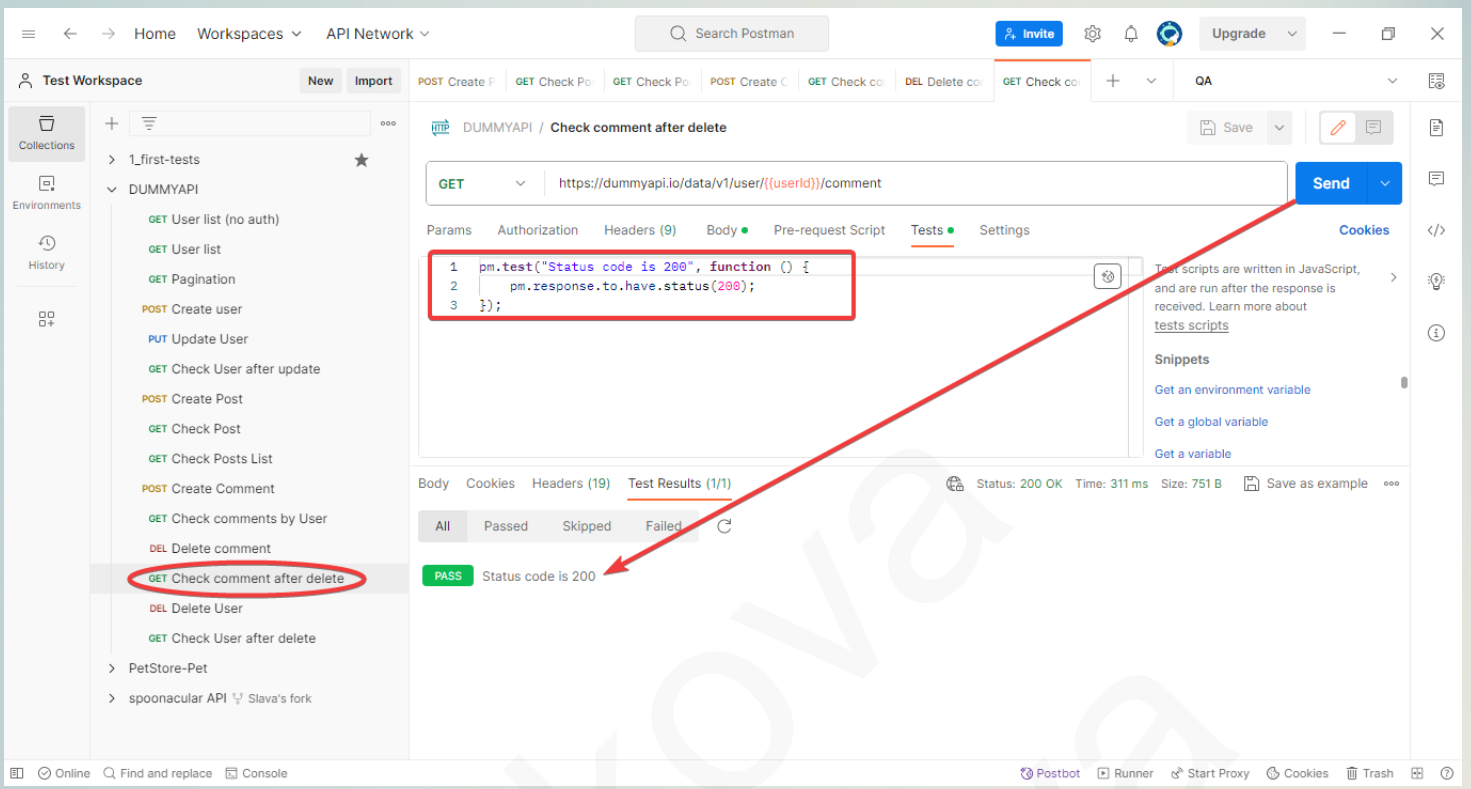
25.11 Check Comments by user test passed.



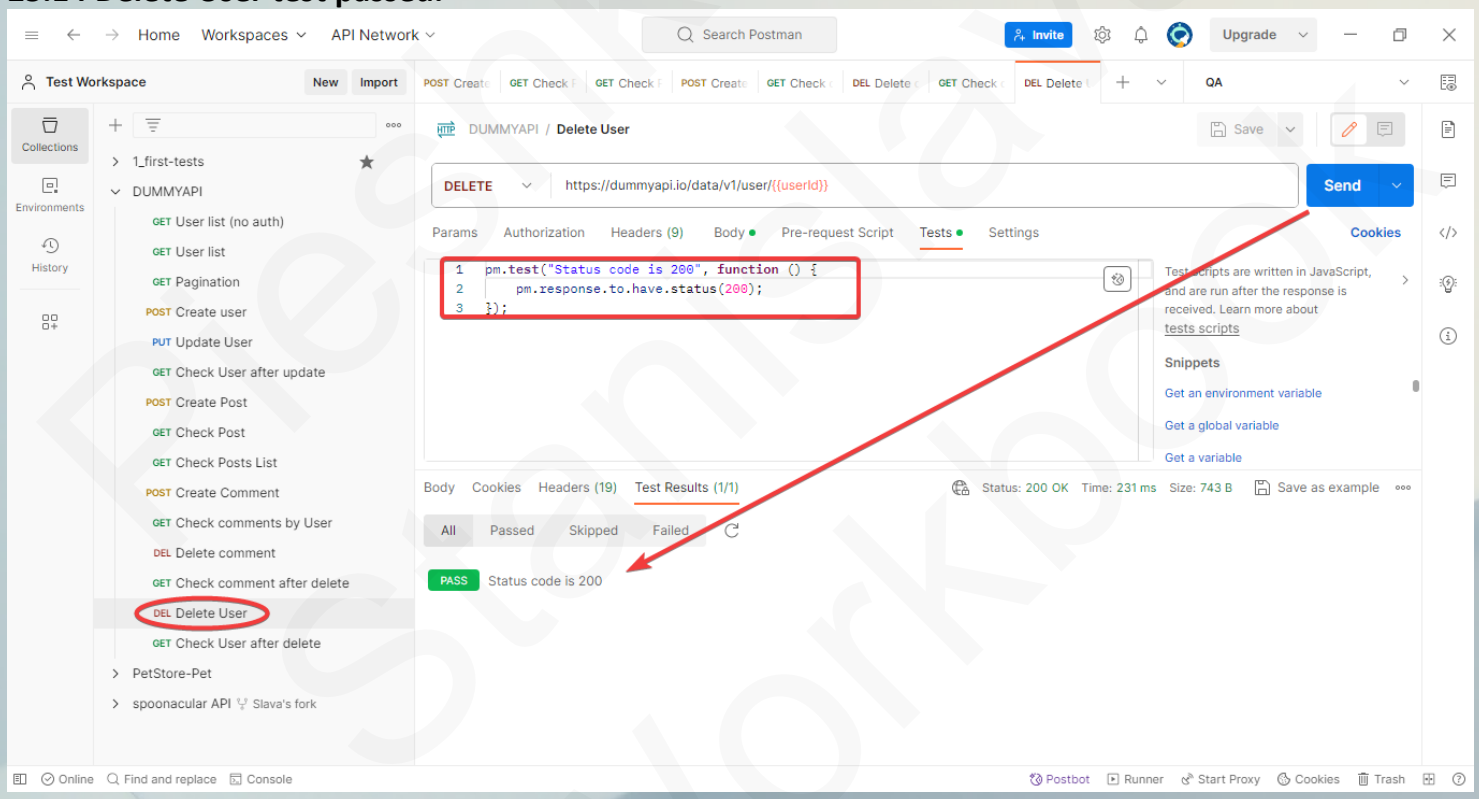
25.12 Delete Comment test passed.



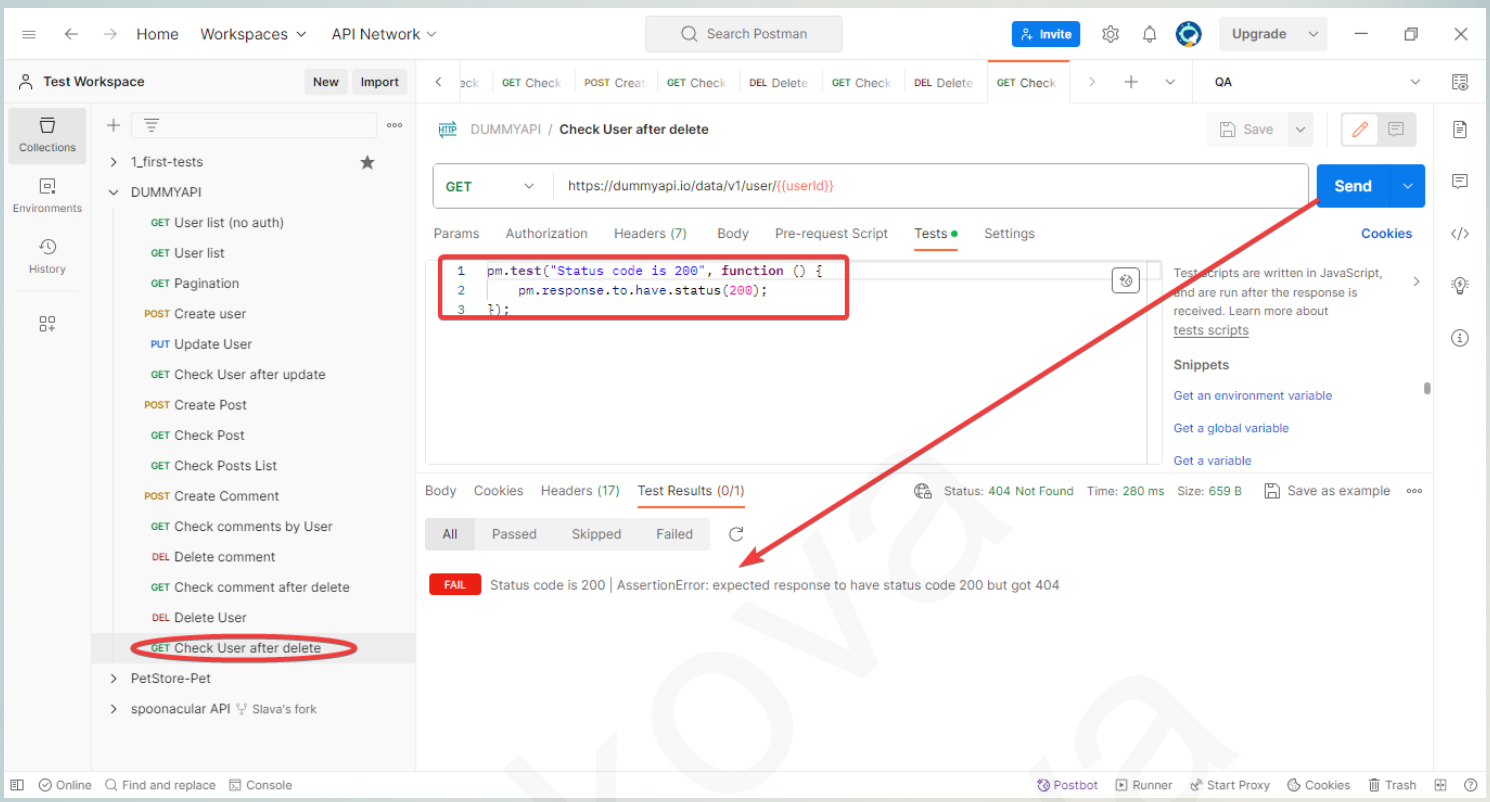
25.13 Check Comment after delete test passed.



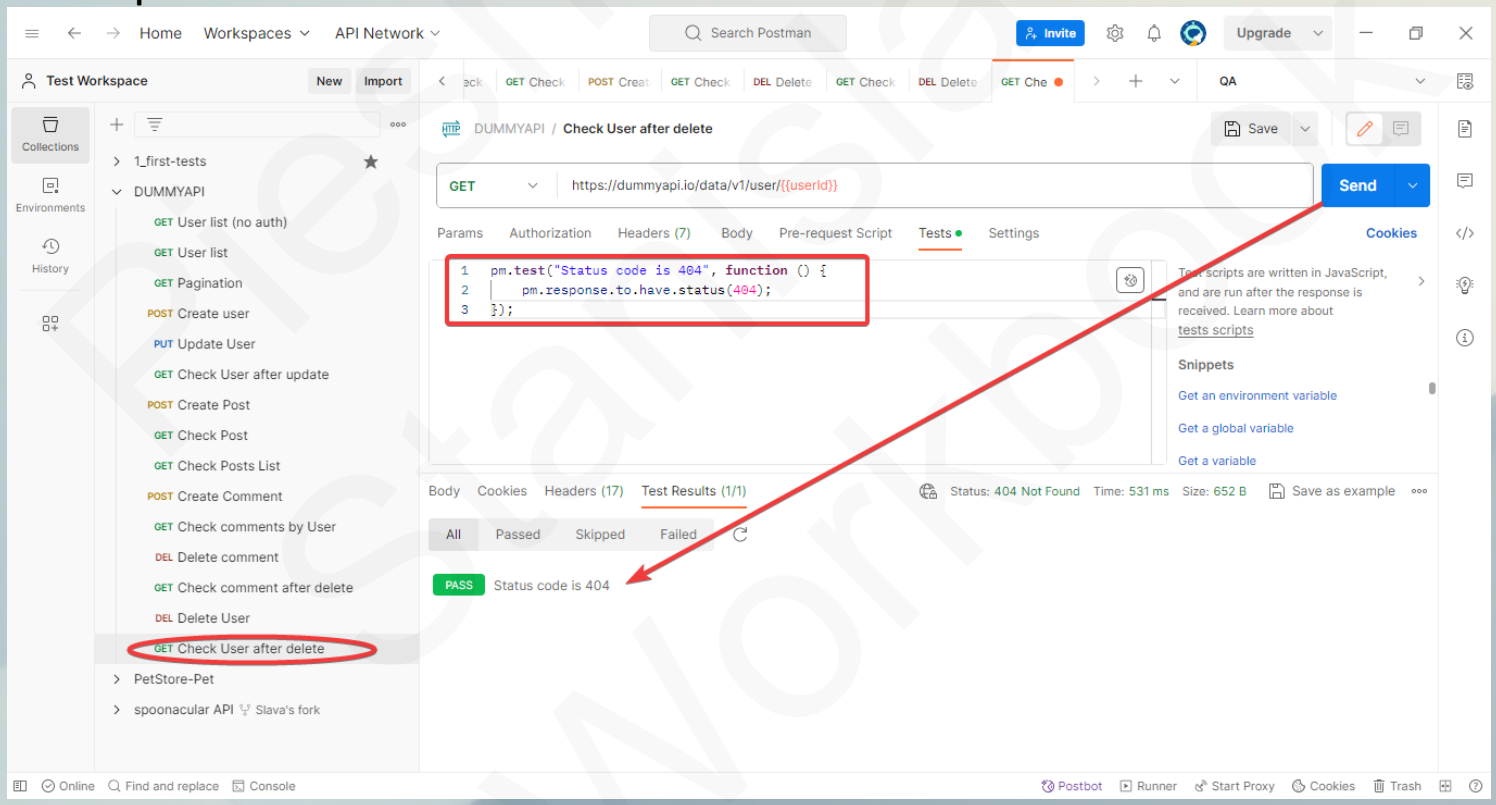
25.14 Delete User test passed.



25.14.1 Check User after delete request. The first test failed. It is expected that the status code is 404, because the user no longer exists.

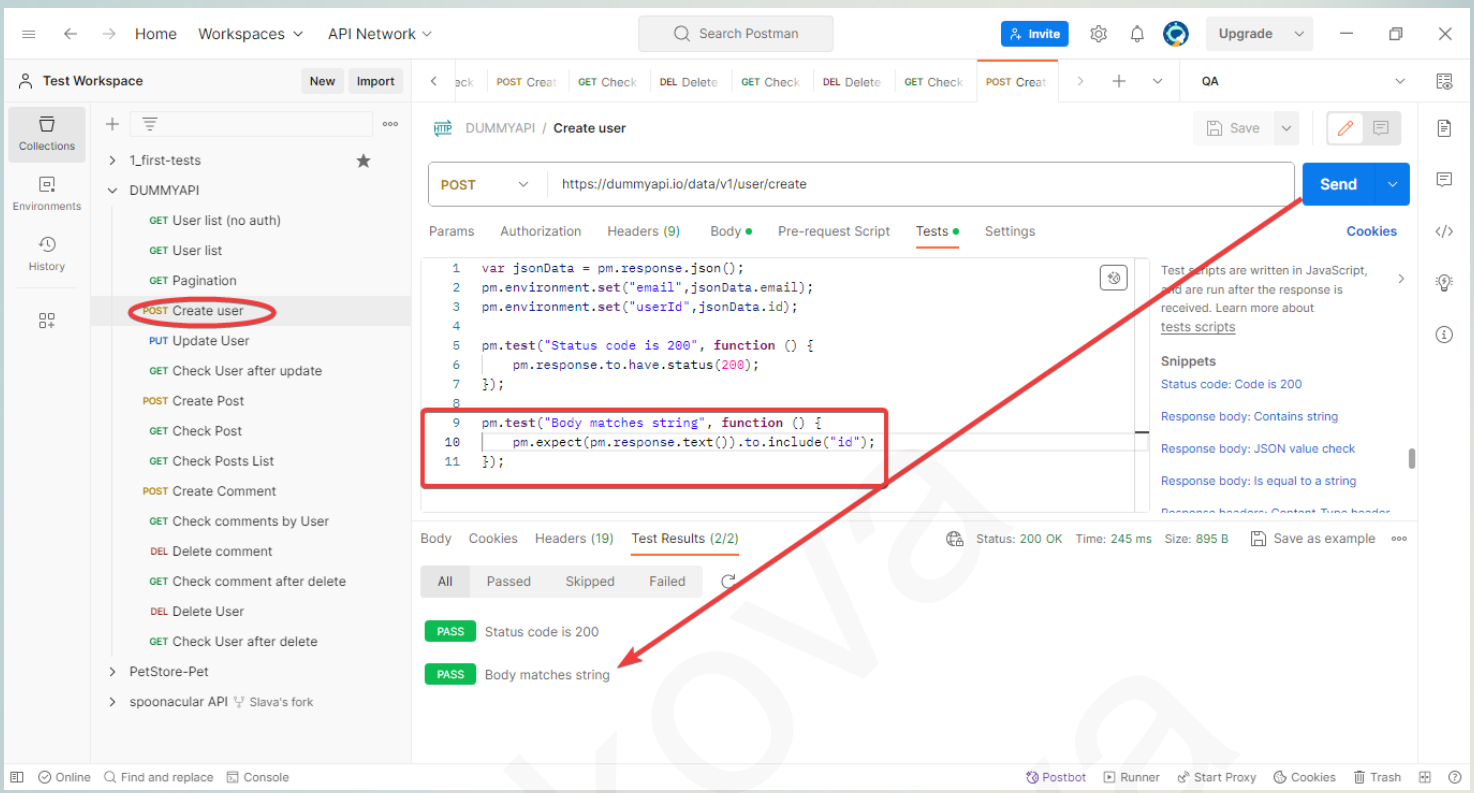


25.14.2 Check User after delete request. The second test passed. It is expected that the status code is 404, because the requested resource does not exist. We expect the status code to be 404 and get the same response.

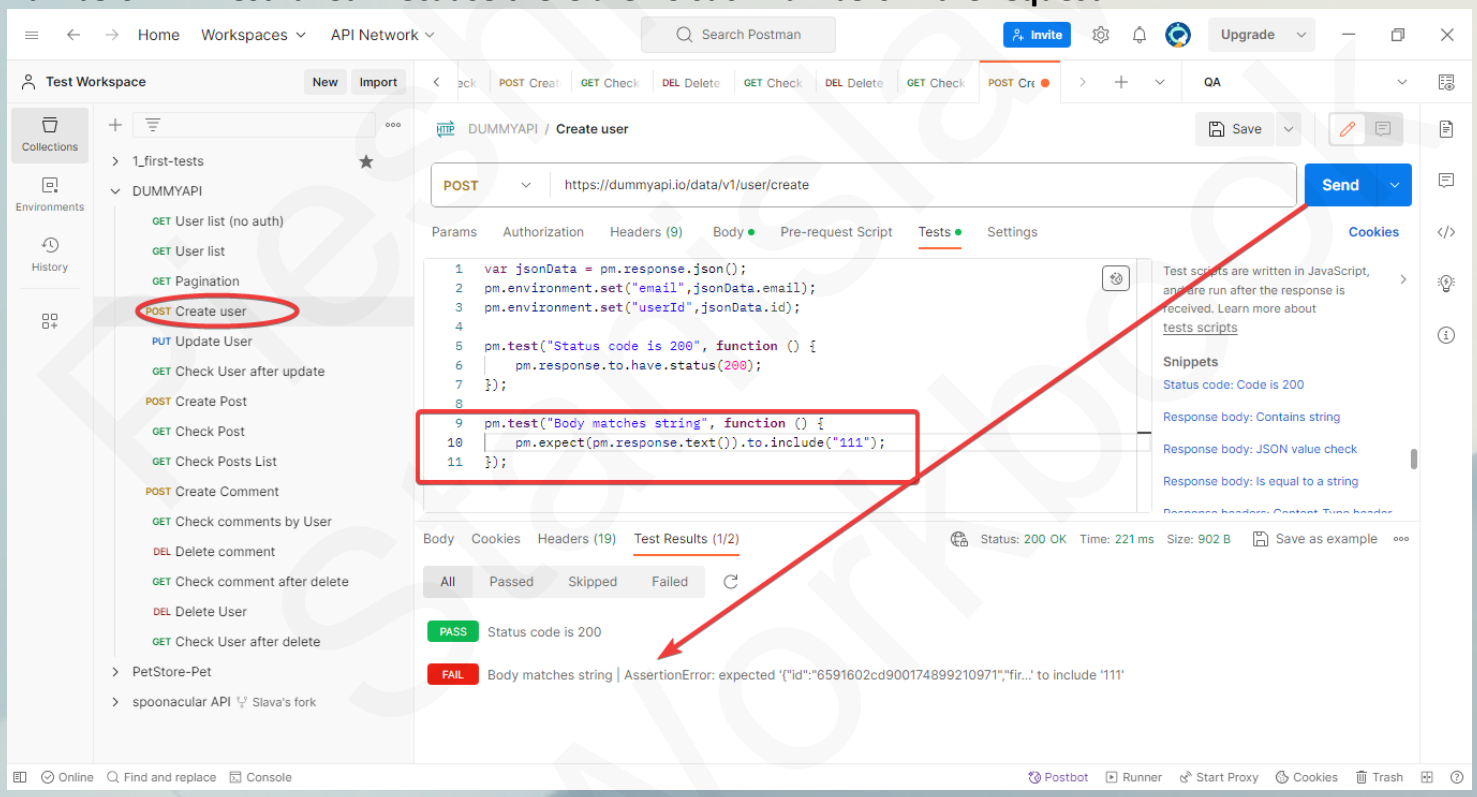


26. Check for the presence of a string in a response with a specific content.

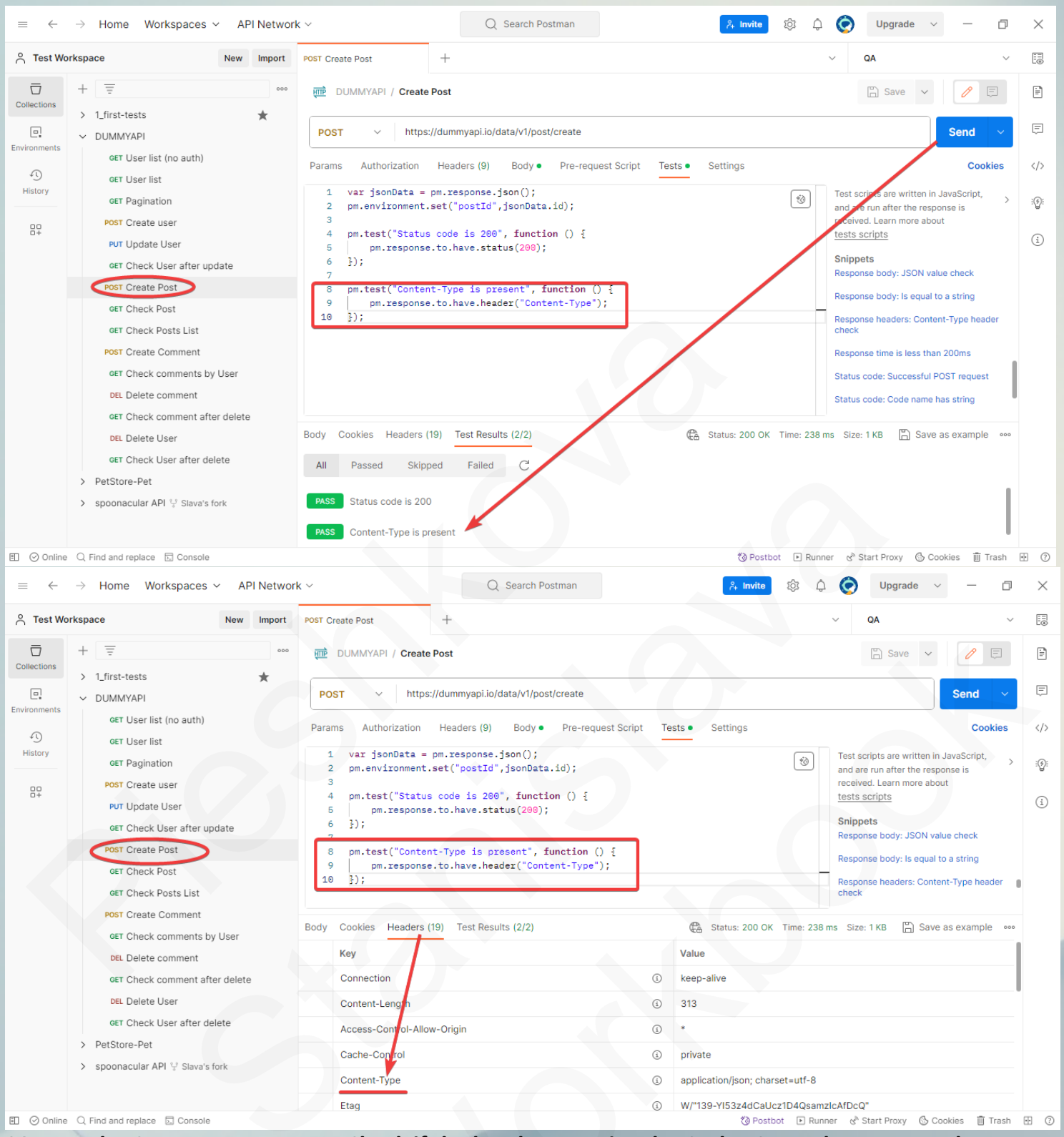
26.1 Test the Create user request. Check that the response contains a string with user's Id. Test passed.



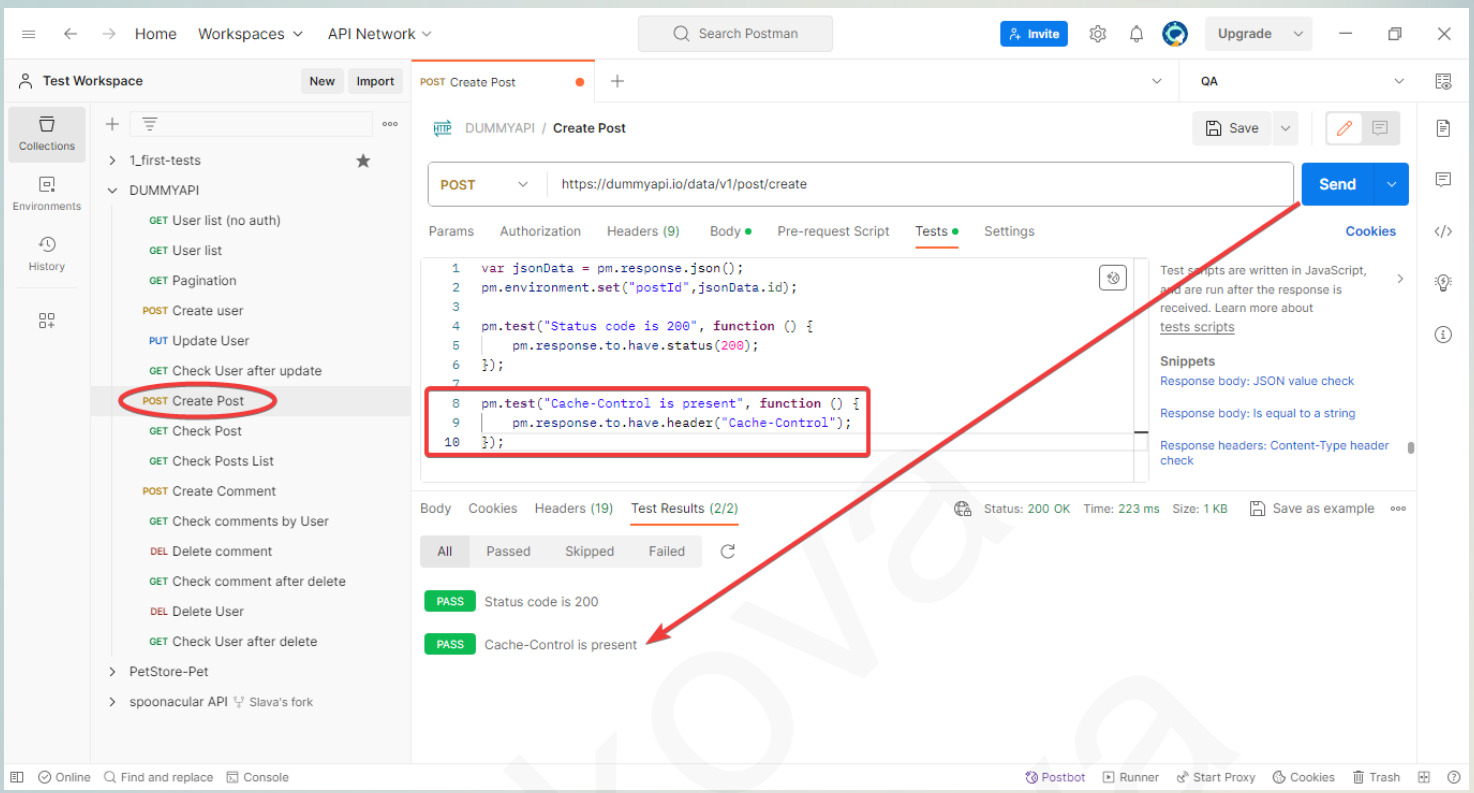
26.2 Test the Create user request. Check that the response contains a string with with arbitrary numbers 111. Test failed. Because there are no such numbers in the request.



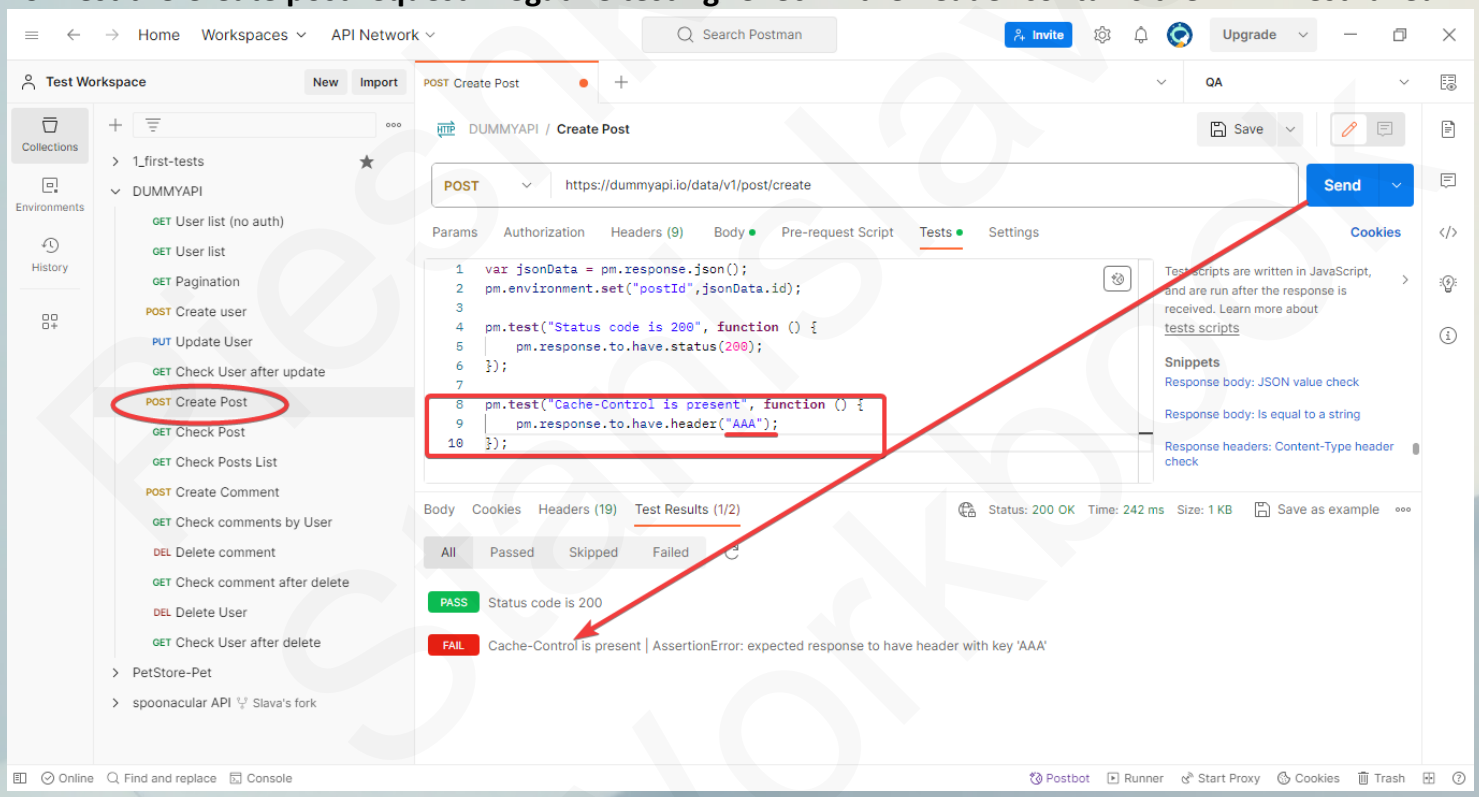
27. Test the Create post request. Check if the header contains the Content-Type. Test passed.



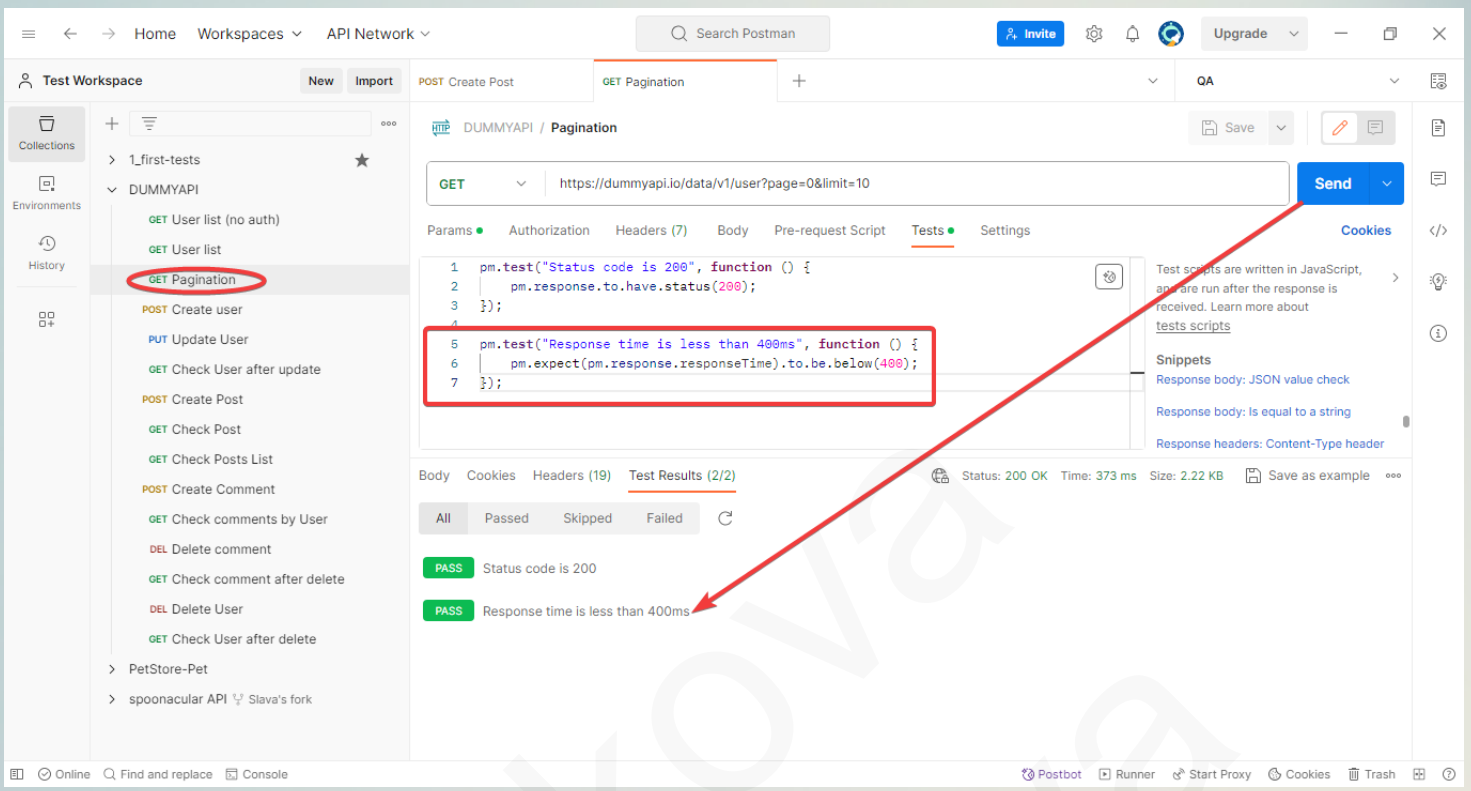
28. Test the Create post request. Check if the header contains the Cache-Control. Test passed.



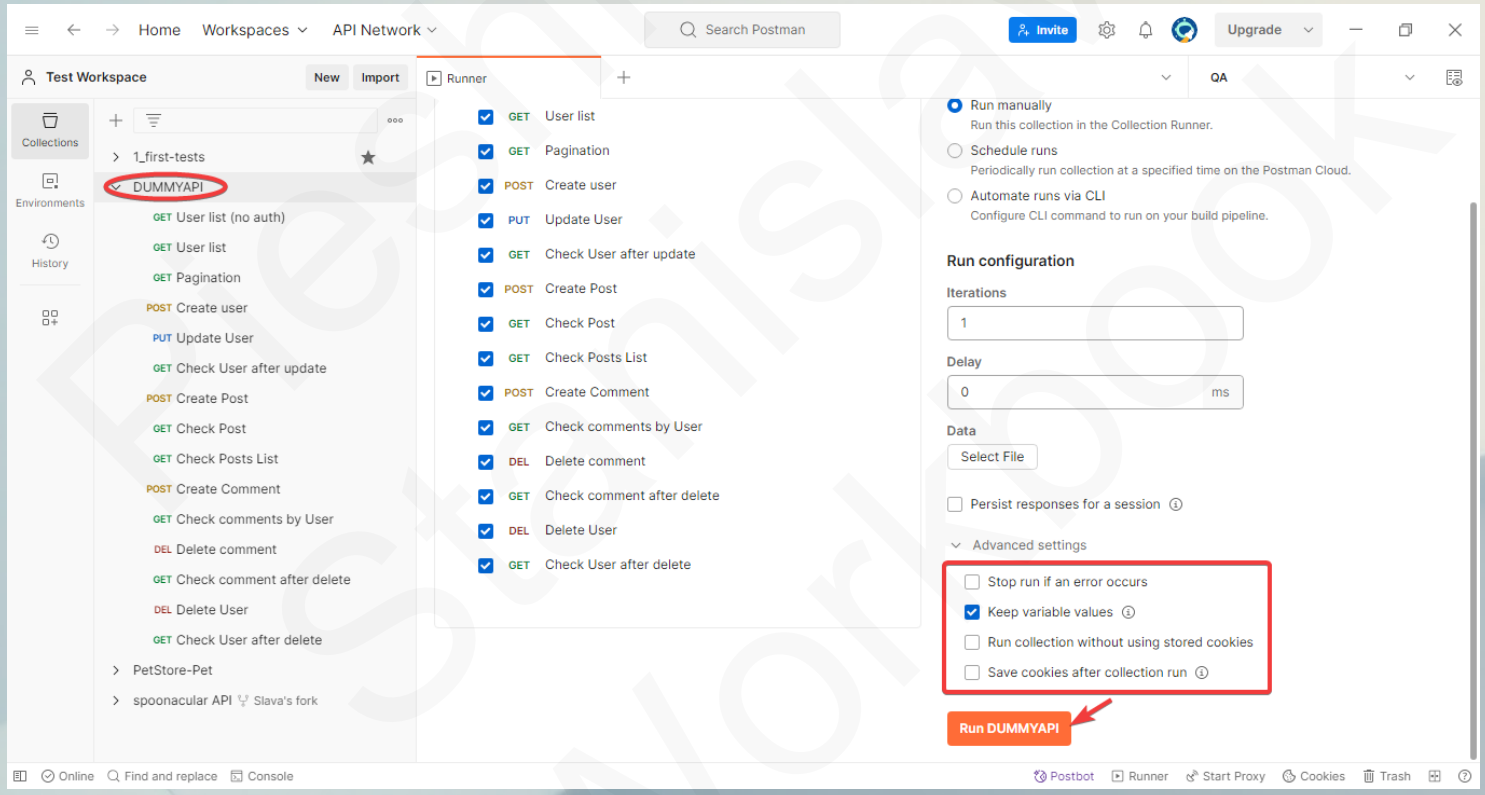
29. Test the Create post request. Negative testing. Check if the header contains the AAA. Test failed.



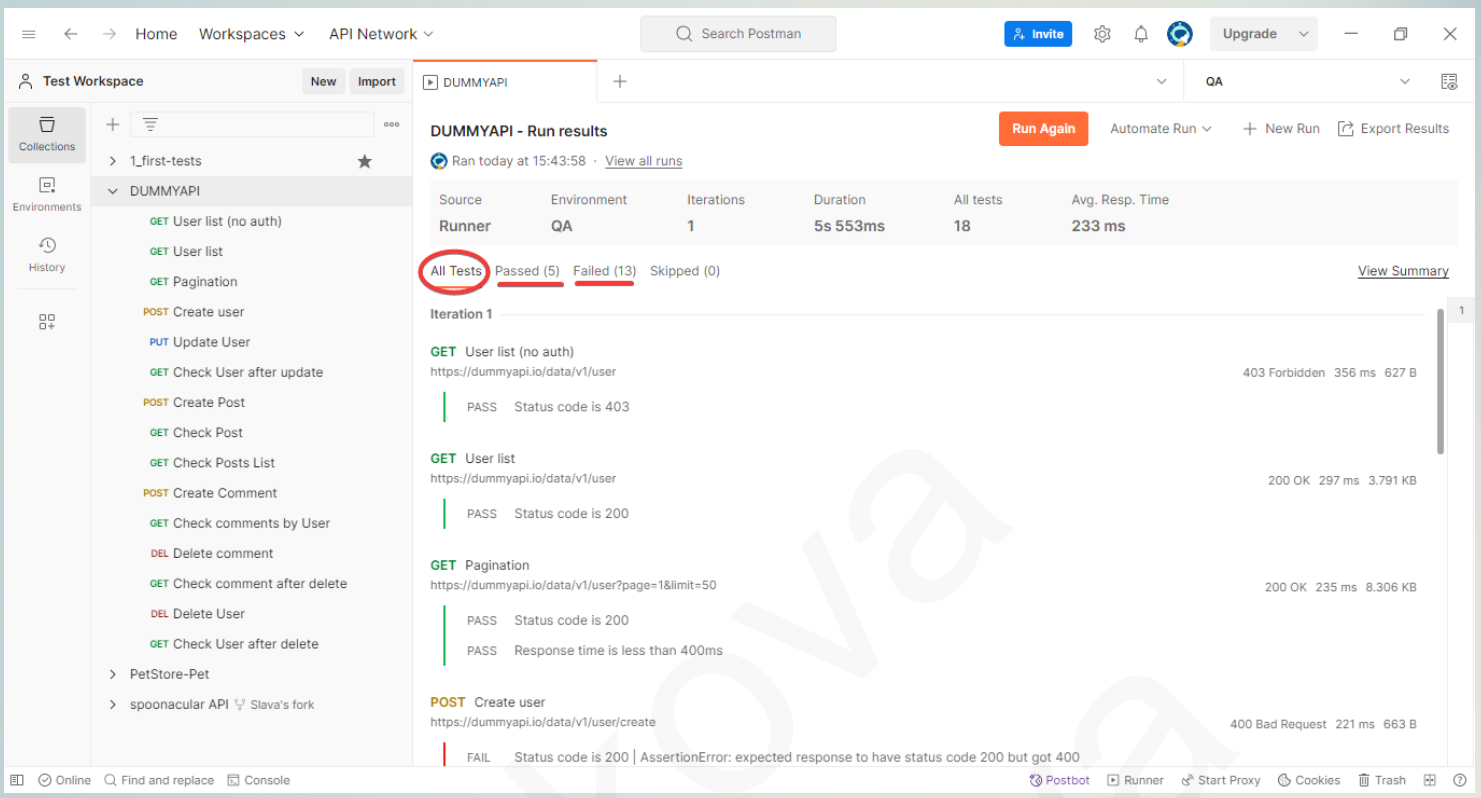
30. Make sure the response time is no more than 400ms. Test passed.



31. Run the entire collection of tests.

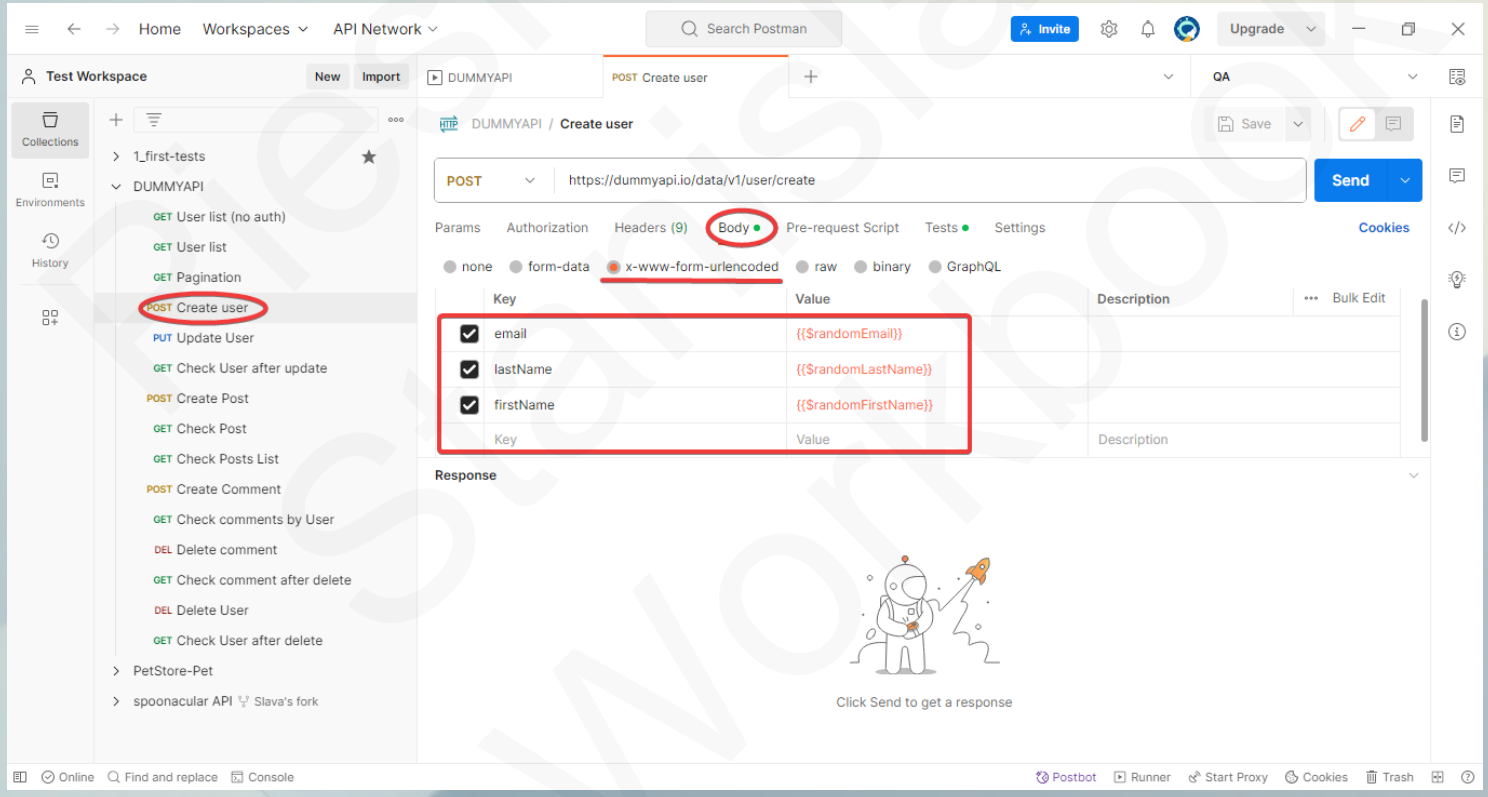


The 5 tests passed, 13 tests failed.

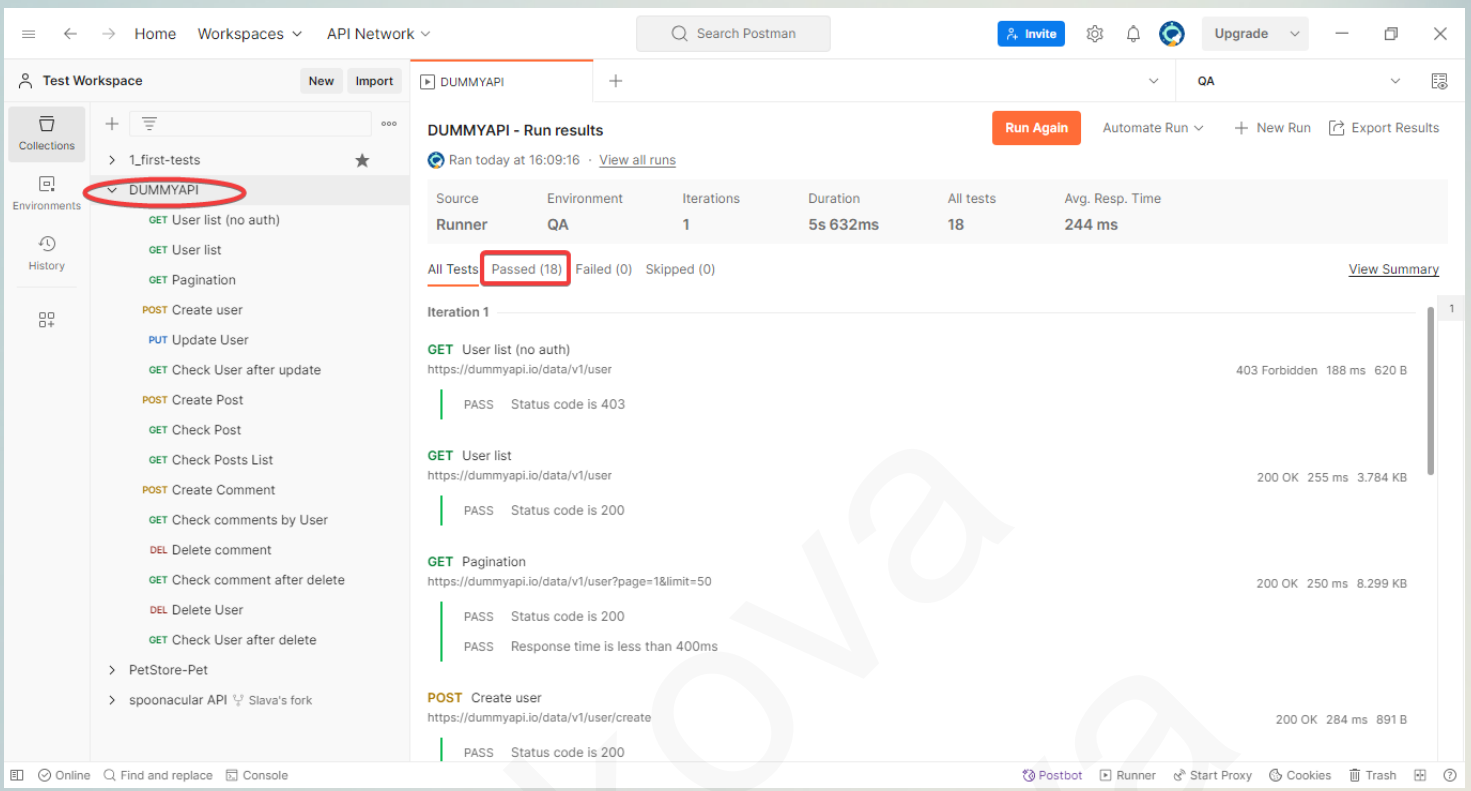


The errors started happening with the Create user request. This is because the email was not changed. It's impossible to create a new user with the same email.

32. To avoid manually entering new user data every time, you can use urlencoded. Create random values for First Name, Last Name, and email.



33. Run the test collection again. Passed 18 tests out of 18.



34. We test the system load. Run 10,000 iterations with a delay of 5 seconds. We study the moment when the server will fail.