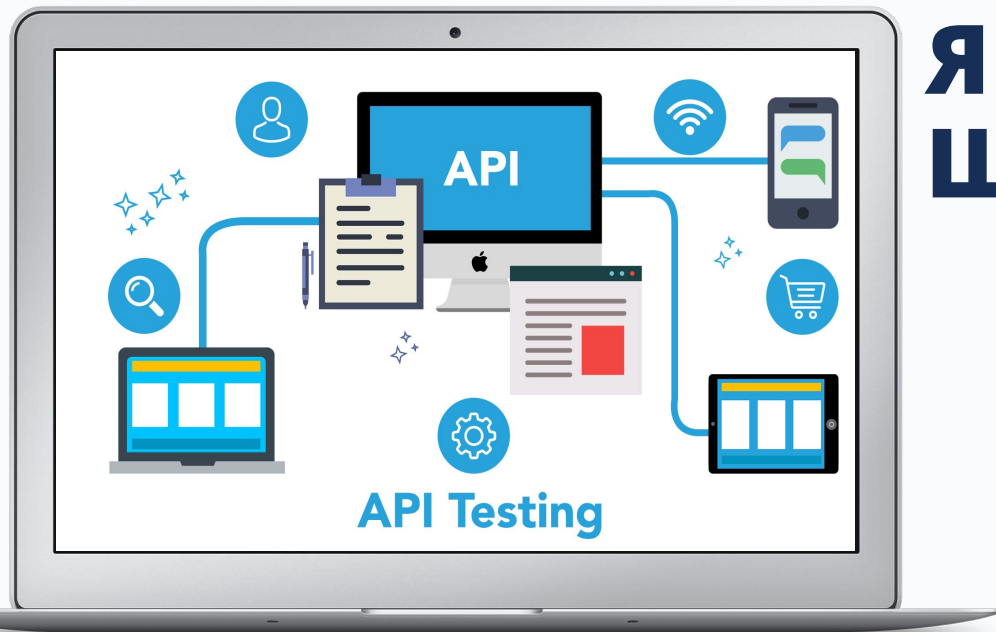




## Become QA Auto



# Як тестувати API. Ще тести

Бутенко Сергій



# План лекції



Опис робочого процесу



Практика

# ⦿ Робочий процес



● **Постановка  
задачі**

● **Аналіз  
залежностей**

● **Написання  
тестів**

● **Запуск тестів**

● **Звіт**

## ⦿ Постановка задачі



### Automate TestCases for Repository Search feature



Edit



Add comment

Assign

More ▾

Escalate



Details



Description

Prioritise and automate testcases related to repository search feature. Please find API documentation by the link <https://docs.github.com/en/rest/search#search-repositories>

# Аналіз залежностей



A	B	C	D	
ID	TestCase Summary	Steps To Reproduce	Expected Result	
1	User is able to search for an existing repo	1. Send HTTP request with existing repository name in parameters	Repository found	
2	User is able to search for non existing repo	1. Send HTTP request with non existing repository name in parameters	Repository not found	
3	User is able to search for the repo with name that consists from 1 character	1. Send HTTP request with existing repository name with length of 1 char	No error occurred	

```
{
  "total_count": 40,
  "incomplete_results": false,
  "items": [
    {
```

# Аналіз залежностей



## Search repositories

✔ Works with [GitHub Apps](#)

Find repositories via various criteria. This method returns up to 100 results [per page](#).

When searching for repositories, you can get text match metadata for the **name** and **description** fields when you pass the `text-match` media type. For more details about how to receive highlighted search results, see [Text match metadata](#).

For example, if you want to search for popular Tetris repositories written in assembly code, your query might look like this:

```
q=tetris+language:assembly&sort=stars&order=desc
```

This query searches for repositories with the word `tetris` in the name, the description, or the README. The results are limited to repositories where the primary language is assembly. The results are sorted by stars in descending order, so that the most popular repositories appear first in the search results.

### Code samples

**GET** /search/repositories

**cURL**   JavaScript   GitHub CLI



```
curl \
-H "Accept: application/vnd.github+json" \
-H "Authorization: token <TOKEN>" \
https://api.github.com/search/repositories
```

### Response

**Example response**

[Response schema](#)

# Аналіз залежностей



## Search repositories

✔ Works with [GitHub Apps](#)

Find repositories via various criteria. This method returns up to 100 results [per page](#).

When searching for repositories, you can get text match metadata for the **name** and **description** fields when you pass the `text-match` media type. For more details about how to receive highlighted search results, see [Text match metadata](#).

For example, if you want to search for popular Tetris repositories written in assembly code, your query might look like this:

```
q=tetris+language:assembly&sort=stars&order=desc
```

This query searches for repositories with the word `tetris` in the name, the description, or the README. The results are limited to repositories where the primary language is assembly. The results are sorted by stars in descending order, so that the most popular repositories appear first in the search results.

### Code samples

GET

/search/repositories

cURL

JavaScriptGitHub CLI

curl \

-H "Accept: application/vnd.github+json" \

-H "Authorization: token <TOKEN>" \

https://api.github.com/search/repositories

### Response

Example response

Response schema



# Аналіз залежностей



## Search repositories

✔ Works with [GitHub Apps](#)

Find repositories via various criteria. This method returns up to 100 results [per page](#).

When searching for repositories, you can get text match metadata for the **name** and **description** fields when you pass the `text-match` media type. For more details about how to receive highlighted search results, see [Text match metadata](#).

For example, if you want to search for popular Tetris repositories written in assembly code, your query might look like this:

```
q=tetris+language:assembly&sort=stars&order=desc
```

This query searches for repositories with the word `tetris` in the name, the description, or the README. The results are limited to repositories where the primary language is assembly. The results are sorted by stars in descending order, so that the most popular repositories appear first in the search results.

### Code samples

GET /search/repositories

cURL JavaScript GitHub CLI



```
curl \
-H "Accept: application/vnd.github+json" \
-H "Authorization: token <TOKEN>" \
https://api.github.com/search/repositories
```

### Response

Example response

Response schema

# ○ Аналіз залежностей



## Parameters

### Headers

**accept** string

Setting to `application/vnd.github+json` is recommended.

### Query parameters

**q** string

Required

The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as the web interface for GitHub. To learn more about the format of the query, see [Constructing a search query](#). See "[Searching for repositories](#)" for a detailed list of qualifiers.

**sort** string

Sorts the results of your query by number of `stars`, `forks`, or `help-wanted-issues` or how recently the items were `updated`. Default: [best match](#)

Can be one of: `stars`, `forks`, `help-wanted-issues`, `updated`

# ○ Аналіз залежностей



## Response

Example response

Response schema

Status: 200

```
{
  "total_count": 40,
  "incomplete_results": false,
  "items": [
    {
      "id": 3081286,
      "node_id": "MDEwOlJlcG9zaXRvcnkzMDgxMjg2",
      "name": "Tetris",
      "full_name": "dtrupenn/Tetris",
      "owner": {
        "login": "dtrupenn",
        "id": 872147,
        "node_id": "MDQ6VXNlcjg3MjE0Nw==",
        "avatar_url": "https://secure.gravatar.com/avatar/e7956084e75f239de8",
        "gravatar_id": "",
        "url": "https://api.github.com/users/dtrupenn",
        "received_events_url": "https://api.github.com/users/dtrupenn/receiv",
        "type": "User",
        "html_url": "https://github.com/octocat",
        "followers_url": "https://api.github.com/users/octocat/followers",
```

# Аналіз залежностей



A	B	C	D	
ID	TestCase Summary	Steps To Reproduce	Expected Result	
1	User is able to search for an existing repo	1. Send HTTP request with existing repository name in parameters	Repository found	
2	User is able to search for non existing repo	1. Send HTTP request with non existing repository name in parameters	Repository not found	
3	User is able to search for the repo with name that consists from 1 character	1. Send HTTP request with existing repository name with length of 1 char	No error occurred	

# Написання тестів



EXPLORER

...

> OPEN EDITORS

▼ BECOME QA AUTO

> \_\_pycache\_\_

> .pytest\_cache

> config

▼ modules

> \_\_pycache\_\_

▼ api / clients

> \_\_pycache\_\_

\_\_init\_\_.py

gihub.py 1, U

> common

gihub.py X

modules > api > clients > gihub.py > GitHub > get\_user

```
1 import requests
2
3
4 class GitHub:
5
6     def get_user(self, username):
7         r = requests.get(f'https://api.github.com/users/{username}')
8         body = r.json()
9
10        return body
```

# Написання тестів



EXPLORER

...

> OPEN EDITORS

✓ BECOME QA AUTO

> \_\_pycache\_\_

> .pytest\_cache

> config

✓ modules

> \_\_pycache\_\_

✓ api / clients

> \_\_pycache\_\_

\_\_init\_\_.py

github.py

> common

> ui

\_\_init\_\_.py

> tests

conftest.py

pytest.ini

github.py

modules > api > clients > github.py > ...

```
1  import requests
2
3
4  class GitHub:
5      def get_user(self, username):
6          r = requests.get(f"https://api.github.com/users/{username}")
7          body = r.json()
8
9          return body
10
11     def search_repo(self, name):
12         r = requests.get(
13             "https://api.github.com/search/repositories",
14             params={"q": name}
15         )
16         body = r.json()
17
18         return body
19
```



# Написання тестів



EXPLORER

OPEN EDITORS

BECOME QA AUTO

> \_\_pycache\_\_

> .pytest\_cache

> config

> modules

tests

> \_\_pycache\_\_

\_\_init\_\_.cpython-38.pyc U

conftest.cpython-38-... U

api

> \_\_pycache\_\_

test\_api.py M

test\_fixtures.py U

test\_github\_api.py 1, U

test\_http.py U

> ui

conftest.py U

pytest.ini U

github.py

test\_github\_api.py X

tests > api > test\_github\_api.py > test\_repo\_can\_be\_found

```
1 import pytest
2
3
4 @pytest.mark.api
5 def test_user_exists(github_api):
6     user = github_api.get_user('defunkt')
7     assert user['login'] == 'defunkt'
8
9
10 @pytest.mark.api
11 def test_user_not_exists(github_api):
12     r = github_api.get_user('butenkosergii')
13     assert r['message'] == 'Not Found'
14
15
16 @pytest.mark.api
17 def test_repo_can_be_found(github_api):
18     r = github_api.search_repo('become-qa-auto')
19     assert r['total_count'] == 13
20     assert 'become-qa-auto' in r['items'][0]['name']
```

15

## ⦿ Написання тестів



```
→ Become QA Auto git:(master) X pytest -m api
===== test session starts =====
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, forked-1.3.0, anyio-3.4.0, xdist-1.32
.0, allure-pytest-2.8.16, html-2.1.1, metadata-1.10.0, ordering-0.6, instafail-0
.4.2
collected 11 items / 8 deselected / 3 selected

tests/api/test_github_api.py ... [100%]

===== 3 passed, 8 deselected in 1.19s =====
```



# Написання тестів



EXPLORER

OPEN EDITORS

BECOME QA AUTO

> \_\_pycache\_\_

> .pytest\_cache

> config

> modules

> tests

> \_\_pycache\_\_

> \_\_init\_\_.cpython-38.pyc

> conftest.cpython-38...

> api

> \_\_pycache\_\_

> test\_api.py

> test\_fixtures.py

> test\_github\_api.py

> test\_http.py

> ui

> conftest.py

> pytest.ini

test\_github\_api.py

test\_repo\_cannot\_be\_found

```
7 | assert user['login'] == 'defunkt'
8 |
9 |
10 | @pytest.mark.api
11 | def test_user_not_exists(github_api):
12 |     r = github_api.get_user('butenko-sergii')
13 |     assert r['message'] == 'Not Found'
14 |
15 |
16 | @pytest.mark.api
17 | def test_repo_can_be_found(github_api):
18 |     r = github_api.search_repo('become-qa-auto')
19 |     assert r['total_count'] == 13
20 |     assert 'become-qa-auto' in r['items'][0]['name']
21 |
22 |
23 | @pytest.mark.api
24 | def test_repo_cannot_be_found(github_api):
25 |     r = github_api.search_repo('sergiibutenko_repo_non_exist')
26 |     assert r['total_count'] == 0
27 |
```

17

## ⦿ Написання тестів



```
→ Become QA Auto git:(master) X pytest -m api
===== test session starts =====
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, forked-1.3.0, anyio-3.4.0, xdist-1.32
.0, allure-pytest-2.8.16, html-2.1.1, metadata-1.10.0, ordering-0.6, instafail-0
.4.2
collected 12 items / 8 deselected / 4 selected

tests/api/test_github_api.py .... [100%]

===== 4 passed, 8 deselected in 1.80s =====
```

# Написання тестів



EXPLORER

OPEN EDITORS

BECOME QA AUTO

> \_\_pycache\_\_

> .pytest\_cache

> config

> modules

tests

> \_\_pycache\_\_

\_\_init\_\_.cpython-38.pyc U

conftest.cpython-38-... U

api

> \_\_pycache\_\_

test\_api.py M

test\_fixtures.py U

test\_github\_api.py 6, U

test\_http.py U

> ui

conftest.py U

github.py

test\_github\_api.py X

tests > api > test\_github\_api.py > test\_repo\_with\_single\_char\_be\_found

```
17 def test_repo_can_be_found(github_api):
18     r = github_api.search_repo('become-qa-auto')
19     assert r['total_count'] == 13
20     assert 'become-qa-auto' in r['items'][0]['name']
21
22
23 @pytest.mark.api
24 def test_repo_cannot_be_found(github_api):
25     r = github_api.search_repo('sergiibutenko_repo_non_exist')
26     assert r['total_count'] == 0
27
28
29
30 @pytest.mark.api
31 def test_repo_with_single_char_be_found(github_api):
32     r = github_api.search_repo('s')
33     assert r['total_count'] != 0
34
```

## ⦿ Написання тестів



```
→ Become QA Auto git:(master) X pytest -m api
===== test session starts =====
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, forked-1.3.0, anyio-3.4.0, xdist-1.32
.0, allure-pytest-2.8.16, html-2.1.1, metadata-1.10.0, ordering-0.6, instafail-0
.4.2
collected 13 items / 8 deselected / 5 selected

tests/api/test_github_api.py ..... [100%]

===== 5 passed, 8 deselected in 3.13s =====
```



## Підсумки



- ✓ Написали тести в такому середовищі, з яким ви будете працювати на реальних проєктах
- ✓ Навчилися відправляти HTTP запити з параметрами пошукового рядка