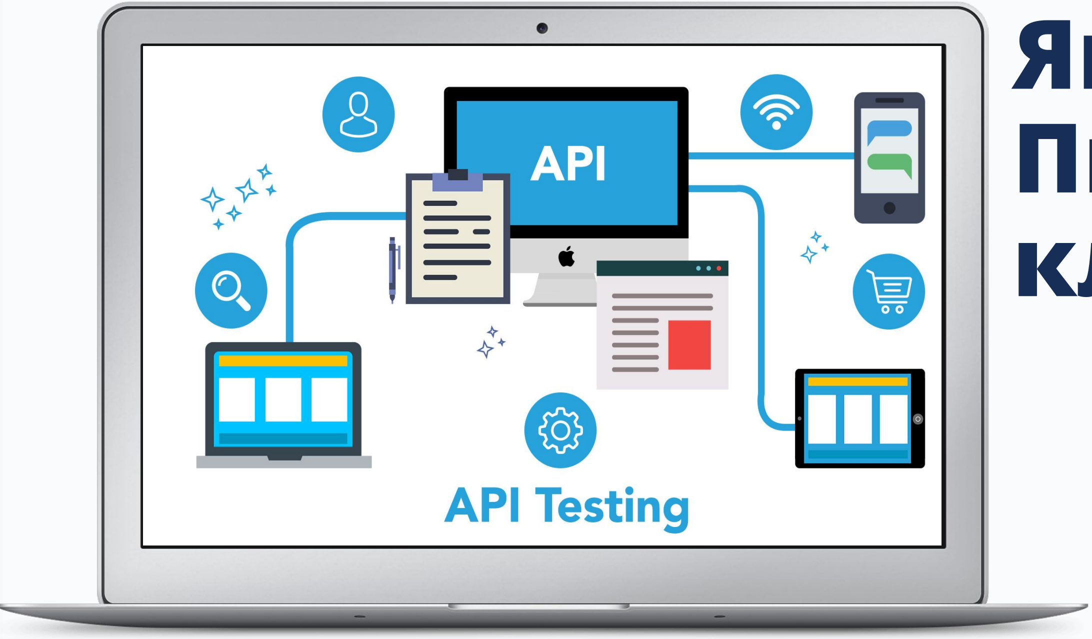
◎ Become QA Auto

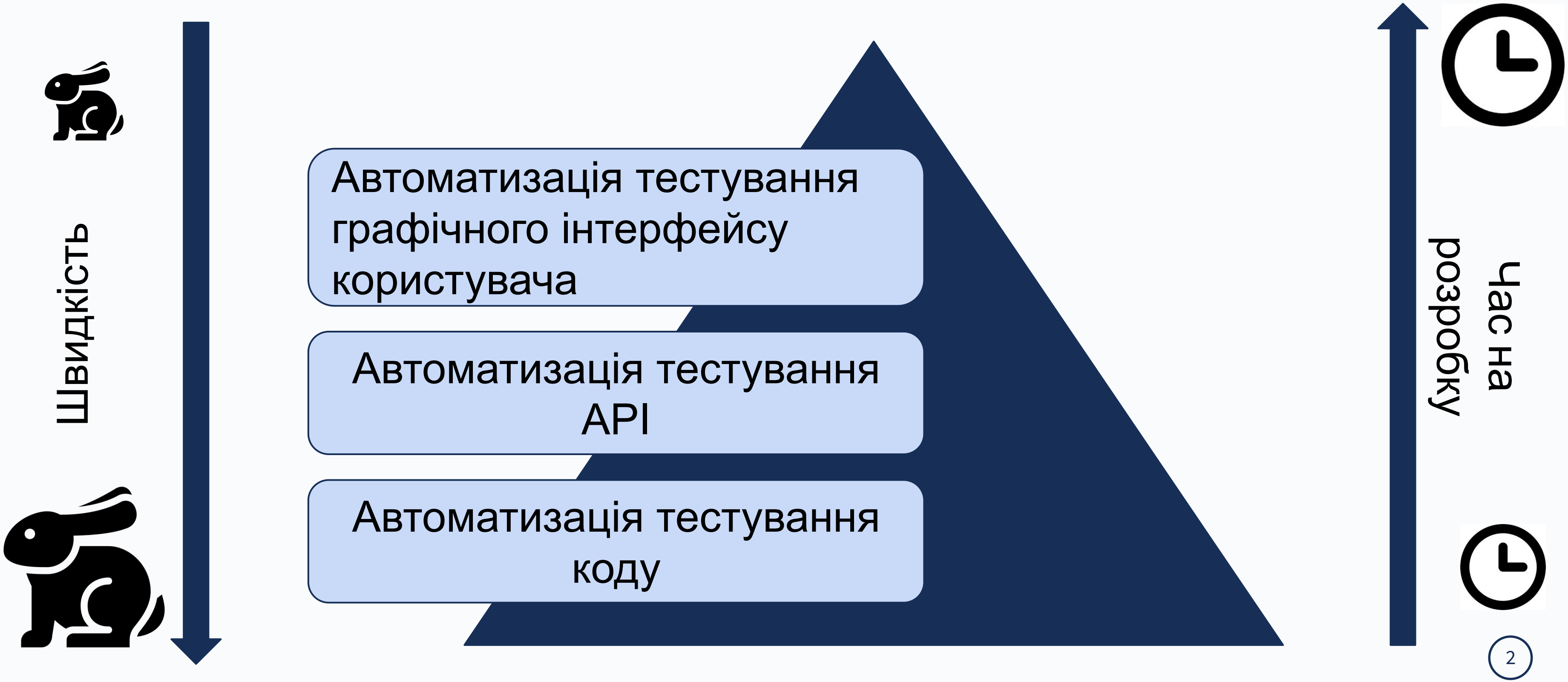
PROMETHEUS



Як тестувати API. Пишемо свій API клієнт

Бутенко Сергій

Властивості автоматизації тестування



Швидкість

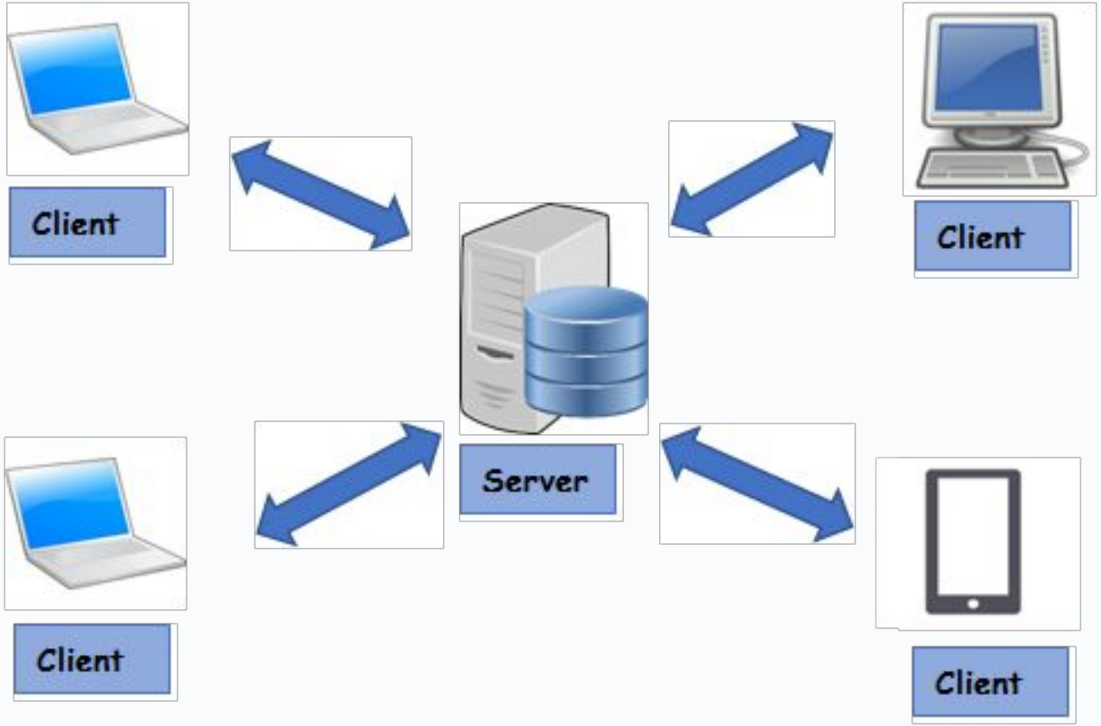
Автоматизація тестування графічного інтерфейсу користувача

Автоматизація тестування API

Автоматизація тестування коду

Час на розробку

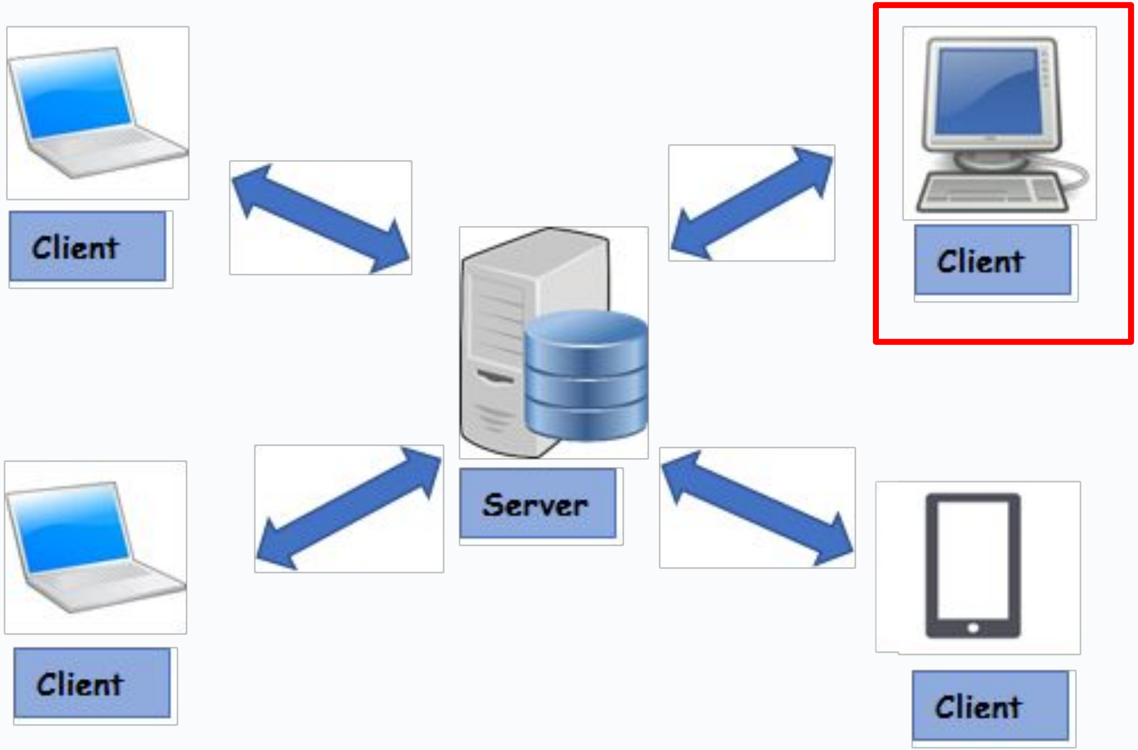
Клієнт-серверна архітектура



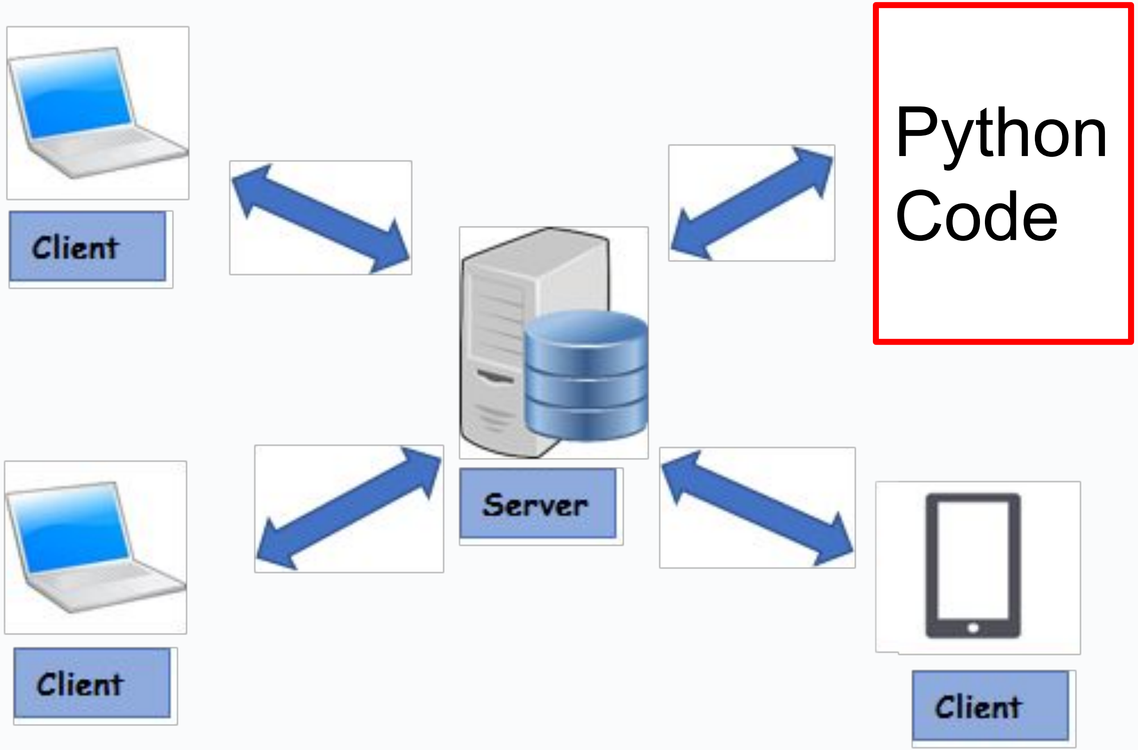
API тестування. Задача

Постановка задачі
Зробити клієнт, який би використовував GITHUB API для комунікації з сервером та використати його в тесті

API тестування

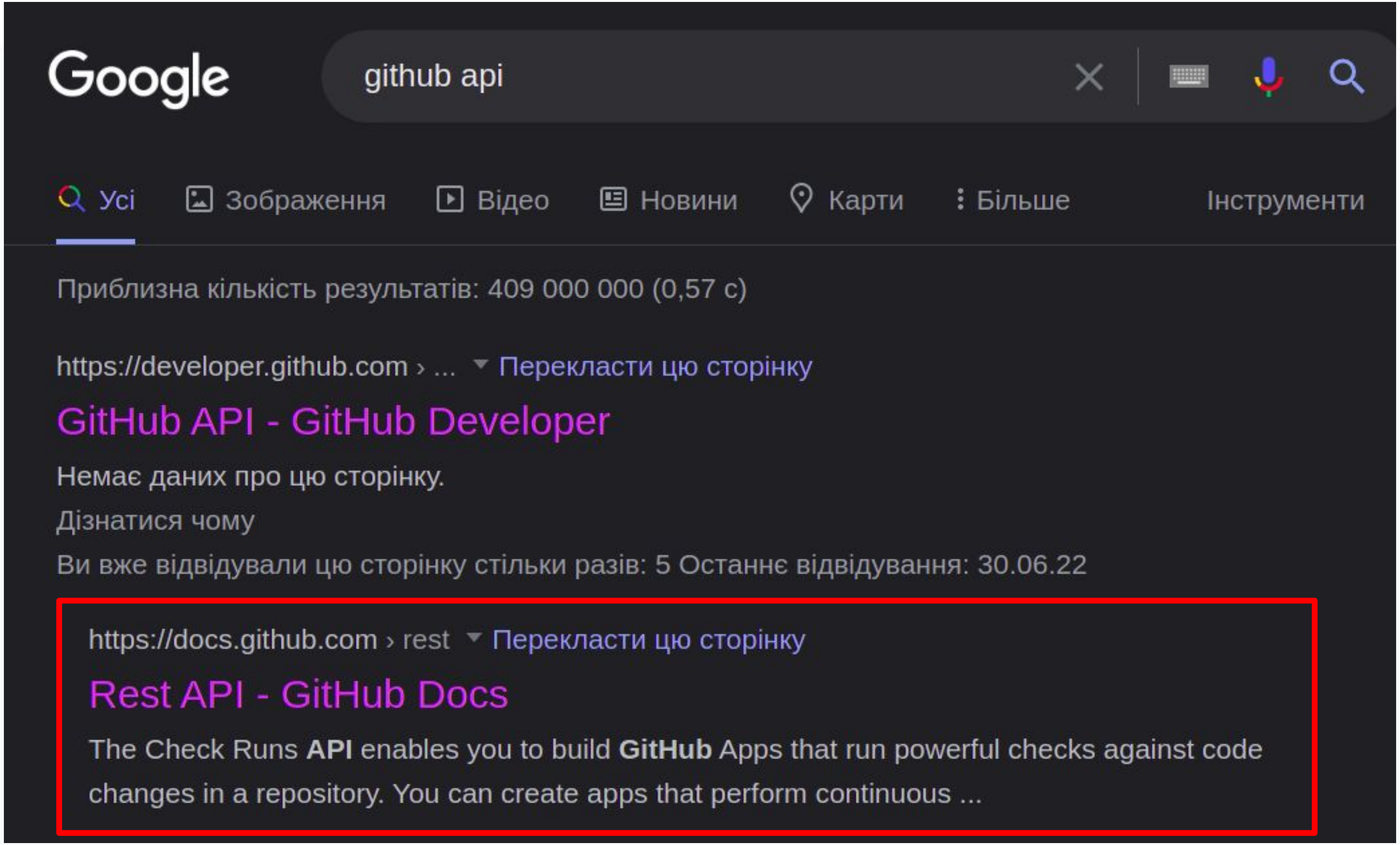


API тестування



Python
Code

Практика



Приблизна кількість результатів: 409 000 000 (0,57 с)

https://developer.github.com › ... ▾ Перекласти цю сторінку

GitHub API - GitHub Developer

Немає даних про цю сторінку.

Дізнатися чому

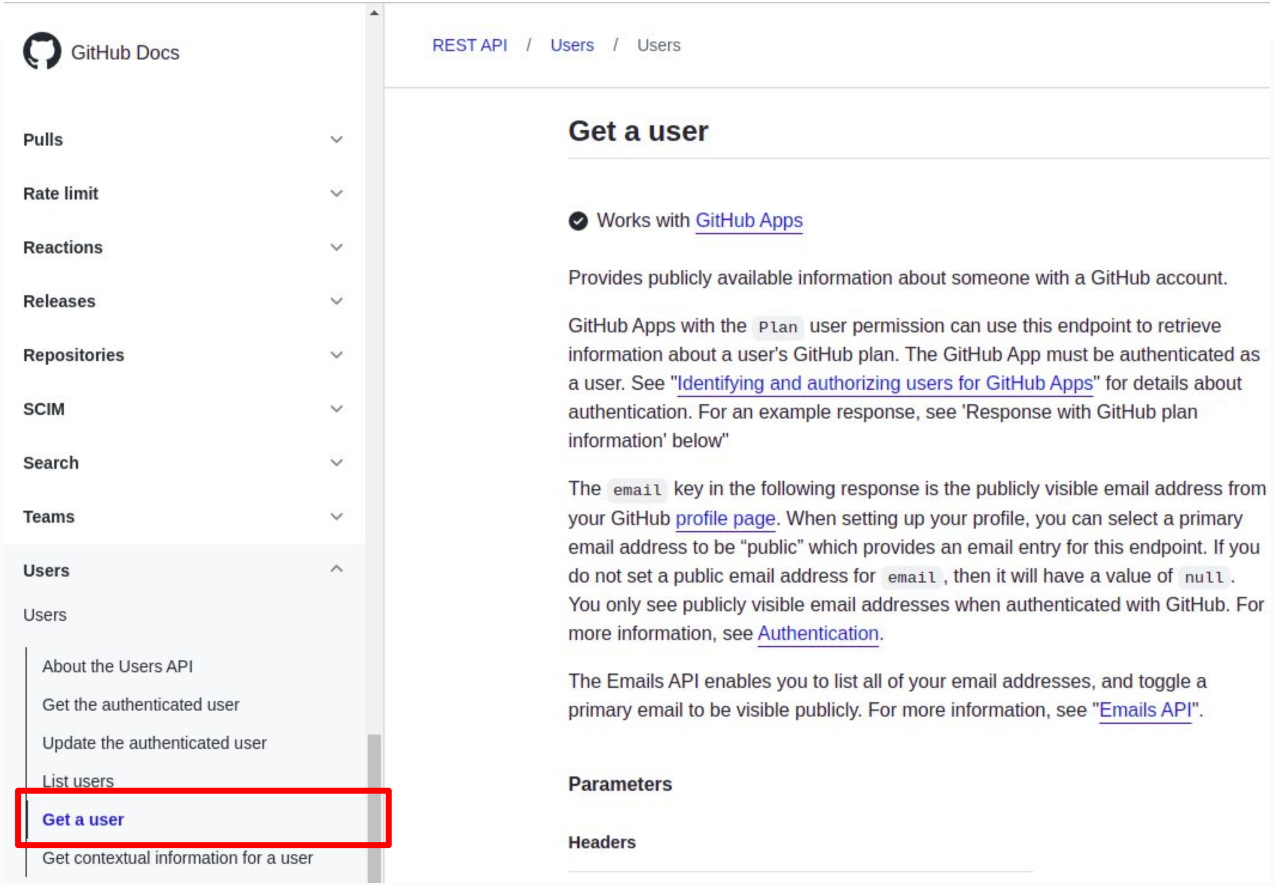
Ви вже відвідували цю сторінку стільки разів: 5 Останнє відвідування: 30.06.22

https://docs.github.com › rest ▾ Перекласти цю сторінку

Rest API - GitHub Docs

The Check Runs **API** enables you to build **GitHub** Apps that run powerful checks against code changes in a repository. You can create apps that perform continuous ...

Практика



GitHub Docs

REST API / Users / Users

Get a user

✓ Works with GitHub Apps

Provides publicly available information about someone with a GitHub account.

GitHub Apps with the `Plan` user permission can use this endpoint to retrieve information about a user's GitHub plan. The GitHub App must be authenticated as a user. See "Identifying and authorizing users for GitHub Apps" for details about authentication. For an example response, see 'Response with GitHub plan information' below"

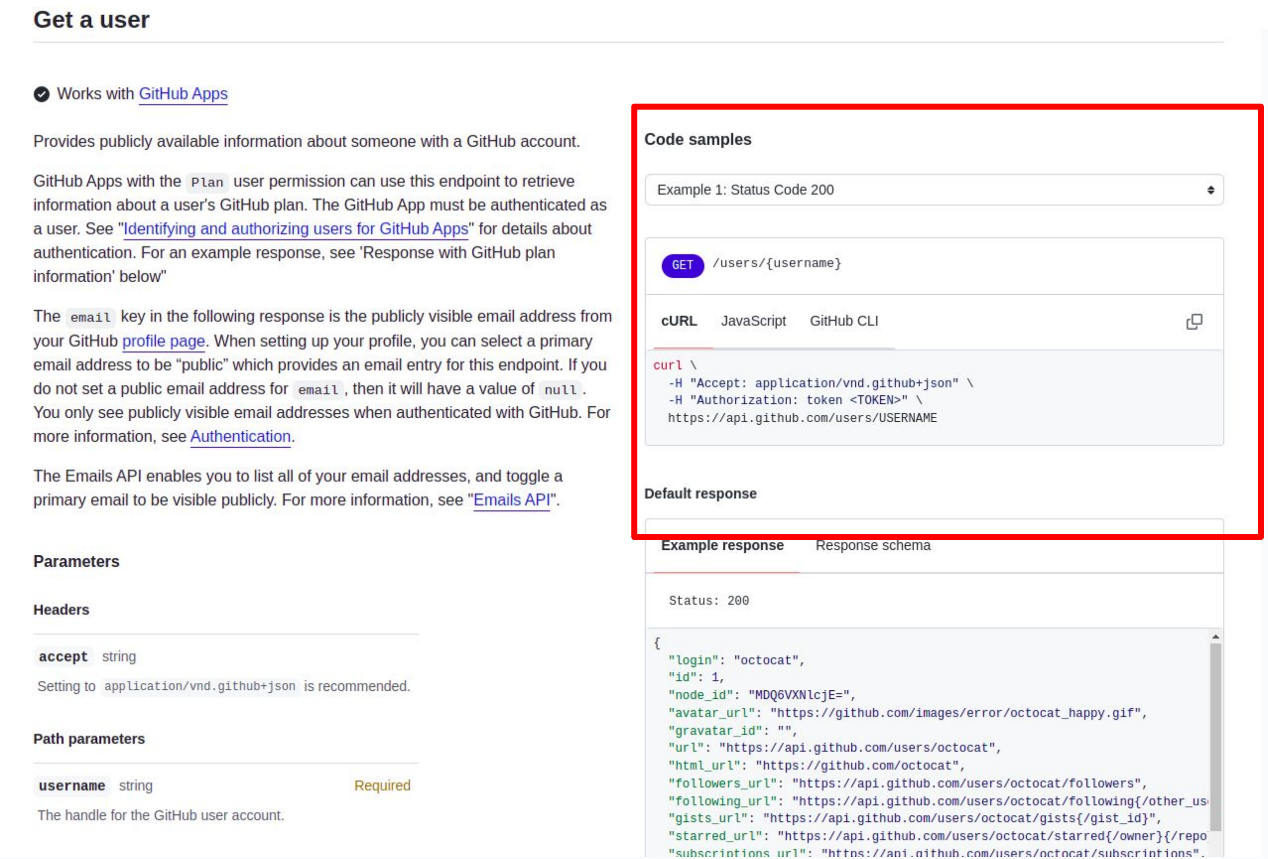
The `email` key in the following response is the publicly visible email address from your GitHub profile page. When setting up your profile, you can select a primary email address to be “public” which provides an email entry for this endpoint. If you do not set a public email address for `email`, then it will have a value of `null`. You only see publicly visible email addresses when authenticated with GitHub. For more information, see Authentication.

The Emails API enables you to list all of your email addresses, and toggle a primary email to be visible publicly. For more information, see "Emails API".

Parameters

Headers

Практика



Get a user

✓ Works with GitHub Apps

Provides publicly available information about someone with a GitHub account.

GitHub Apps with the `Plan` user permission can use this endpoint to retrieve information about a user's GitHub plan. The GitHub App must be authenticated as a user. See "Identifying and authorizing users for GitHub Apps" for details about authentication. For an example response, see 'Response with GitHub plan information' below"

The `email` key in the following response is the publicly visible email address from your GitHub profile page. When setting up your profile, you can select a primary email address to be “public” which provides an email entry for this endpoint. If you do not set a public email address for `email`, then it will have a value of `null`. You only see publicly visible email addresses when authenticated with GitHub. For more information, see Authentication.

The Emails API enables you to list all of your email addresses, and toggle a primary email to be visible publicly. For more information, see "Emails API".

Parameters

Headers

accept string

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

Path parameters

username string Required

The handle for the GitHub user account.

Code samples

Example 1: Status Code 200

GET `/users/{username}`

cURL JavaScript GitHub CLI

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

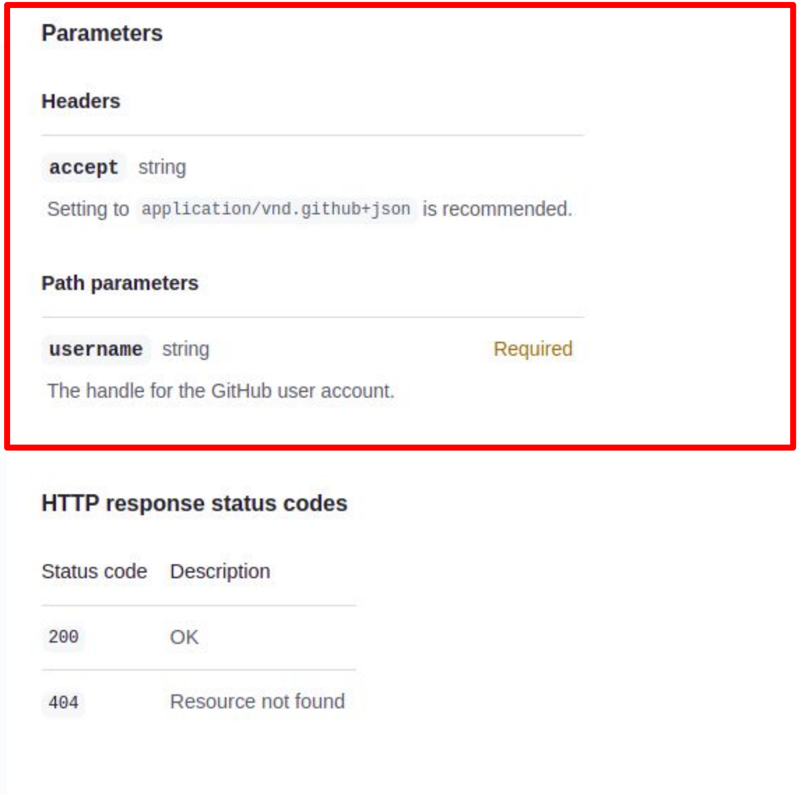
Default response

Example response Response schema

```
Status: 200
```

```
{
  "login": "octocat",
  "id": 1,
  "node_id": "MDQ6VXNlcjE=",
  "avatar_url": "https://github.com/images/error/octocat_happy.gif",
  "gravatar_id": "",
  "url": "https://api.github.com/users/octocat",
  "html_url": "https://github.com/octocat",
  "followers_url": "https://api.github.com/users/octocat/followers",
  "following_url": "https://api.github.com/users/octocat/following{/other_us
  "gists_url": "https://api.github.com/users/octocat/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/octocat/starred{/owner}{/repo
  "subscriptions_url": "https://api.github.com/users/octocat/subscriptions",
```

Практика



Parameters

Headers

accept string

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

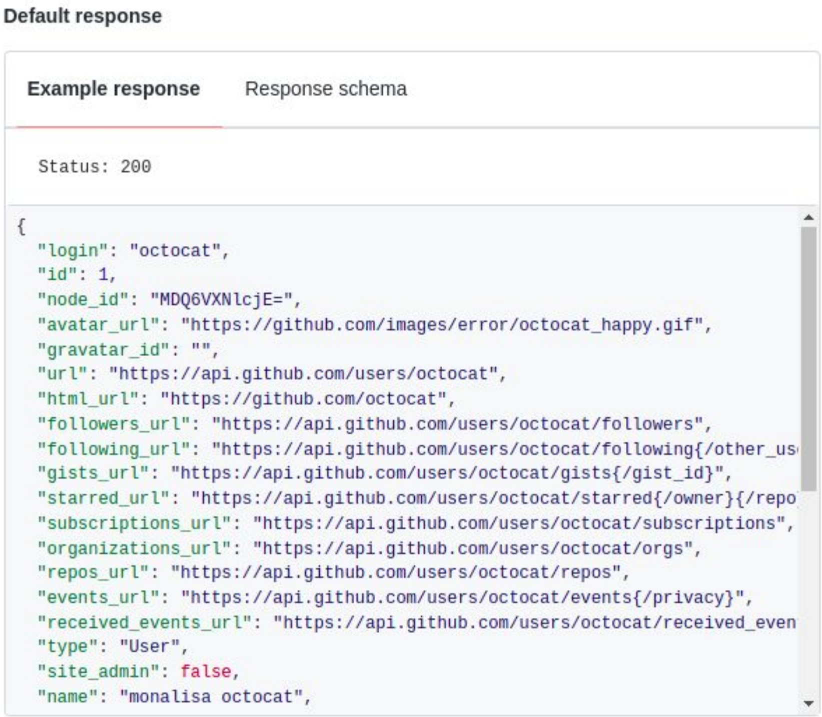
Path parameters

username string Required

The handle for the GitHub user account.

HTTP response status codes

Status code	Description
200	OK
404	Resource not found

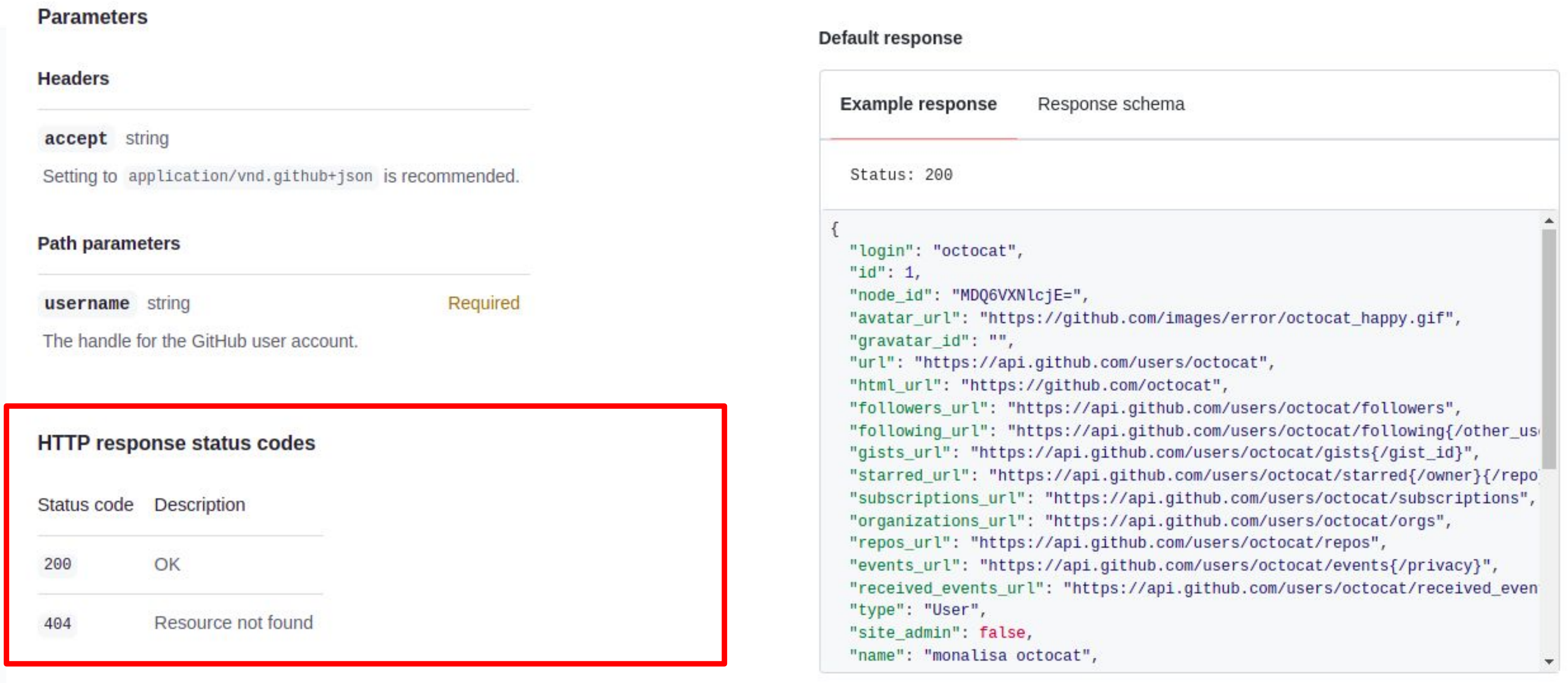


Default response

```
Status: 200
```

```
{
  "login": "octocat",
  "id": 1,
  "node_id": "MDQ6VXNlcjE=",
  "avatar_url": "https://github.com/images/error/octocat_happy.gif",
  "gravatar_id": "",
  "url": "https://api.github.com/users/octocat",
  "html_url": "https://github.com/octocat",
  "followers_url": "https://api.github.com/users/octocat/followers",
  "following_url": "https://api.github.com/users/octocat/following{/other_us
  "gists_url": "https://api.github.com/users/octocat/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/octocat/starred{/owner}{/repo
  "subscriptions_url": "https://api.github.com/users/octocat/subscriptions",
  "organizations_url": "https://api.github.com/users/octocat/orgs",
  "repos_url": "https://api.github.com/users/octocat/repos",
  "events_url": "https://api.github.com/users/octocat/events{/privacy}",
  "received_events_url": "https://api.github.com/users/octocat/received_even
  "type": "User",
  "site_admin": false,
  "name": "monalisa octocat",
```

Практика



Parameters

Headers

accept string

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

Path parameters

username string Required

The handle for the GitHub user account.

HTTP response status codes

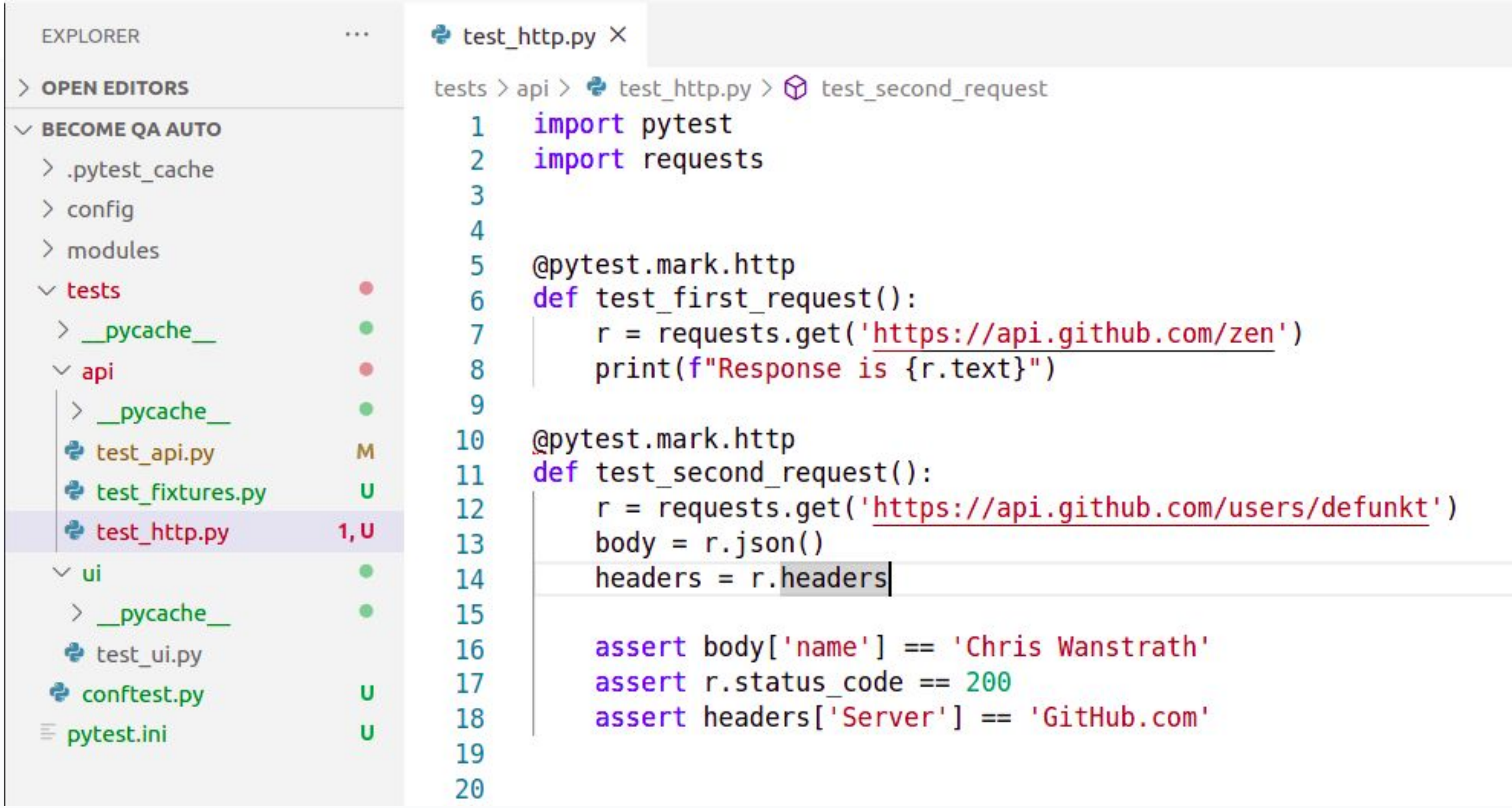
Status code	Description
200	OK
404	Resource not found

Default response

```
Status: 200
```

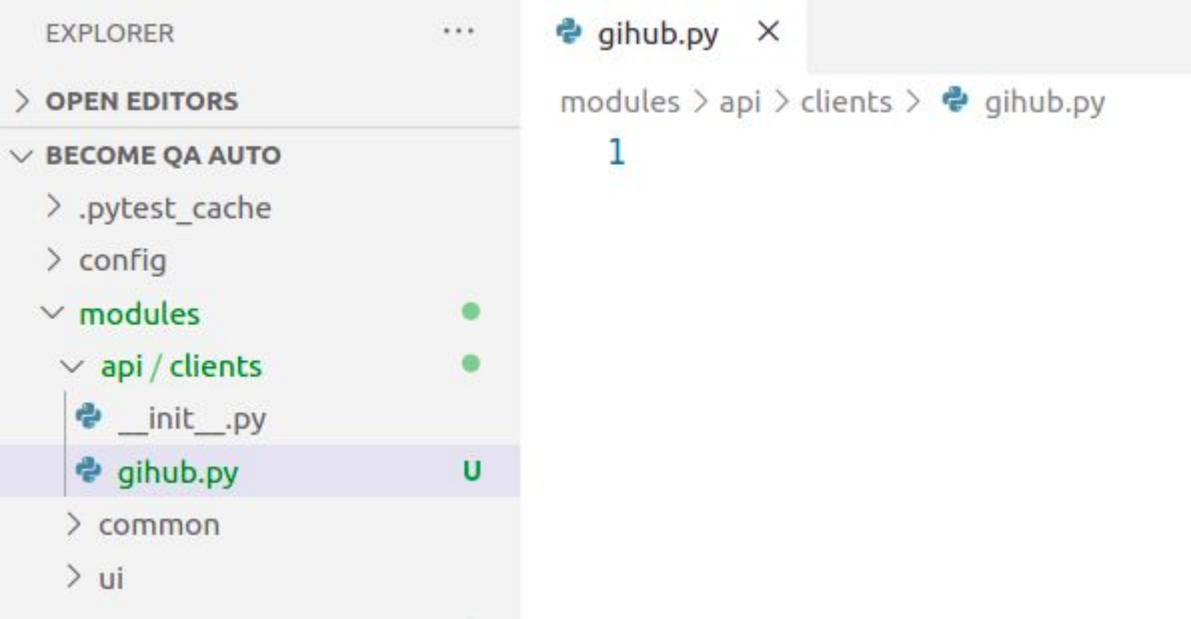
```
{
  "login": "octocat",
  "id": 1,
  "node_id": "MDQ6VXNlcjE=",
  "avatar_url": "https://github.com/images/error/octocat_happy.gif",
  "gravatar_id": "",
  "url": "https://api.github.com/users/octocat",
  "html_url": "https://github.com/octocat",
  "followers_url": "https://api.github.com/users/octocat/followers",
  "following_url": "https://api.github.com/users/octocat/following{/other_us
  "gists_url": "https://api.github.com/users/octocat/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/octocat/starred{/owner}{/repo
  "subscriptions_url": "https://api.github.com/users/octocat/subscriptions",
  "organizations_url": "https://api.github.com/users/octocat/orgs",
  "repos_url": "https://api.github.com/users/octocat/repos",
  "events_url": "https://api.github.com/users/octocat/events{/privacy}",
  "received_events_url": "https://api.github.com/users/octocat/received_even
  "type": "User",
  "site_admin": false,
  "name": "monalisa octocat",
```

Практика



```python
import pytest
import requests


@pytest.mark.http
def test_first_request():
    r = requests.get('https://api.github.com/zen')
    print(f"Response is {r.text}")

@pytest.mark.http
def test_second_request():
    r = requests.get('https://api.github.com/users/defunkt')
    body = r.json()
    headers = r.headers

    assert body['name'] == 'Chris Wanstrath'
    assert r.status_code == 200
    assert headers['Server'] == 'GitHub.com'


```

Практика



EXPLORER ···

> OPEN EDITORS

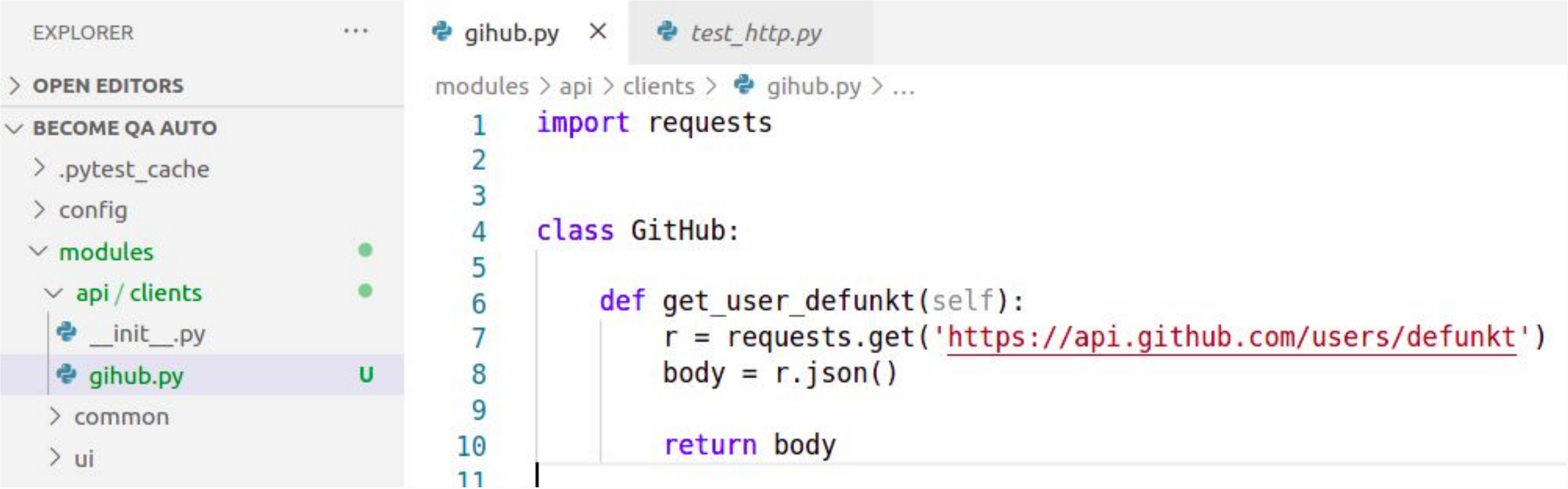
BECOME QA AUTO
- .pytest_cache
- config
- modules ●
  - api / clients ●
    - __init__.py
    - gihub.py U
  - common
  - ui

gihub.py ×

modules > api > clients > gihub.py

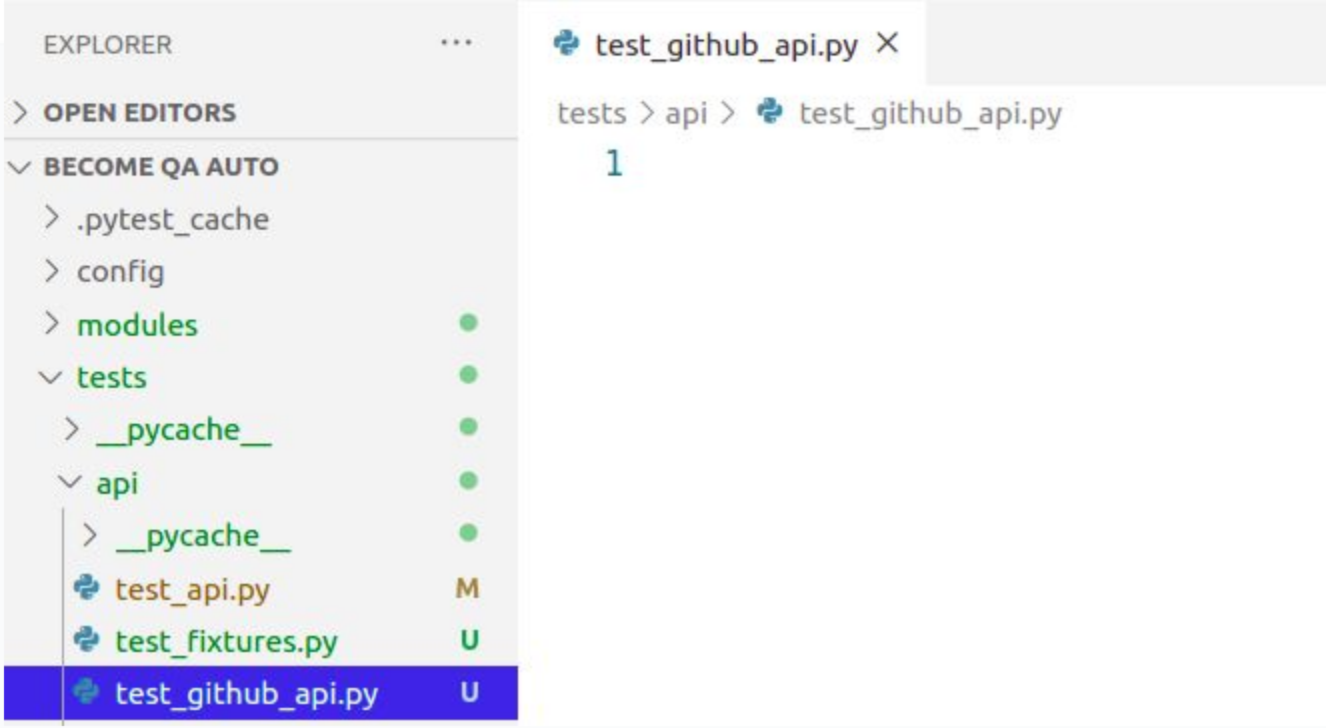
```
1
```

Практика



```python
import requests


class GitHub:

    def get_user_defunkt(self):
        r = requests.get('https://api.github.com/users/defunkt')
        body = r.json()

        return body

```

Практика



EXPLORER ···

› OPEN EDITORS

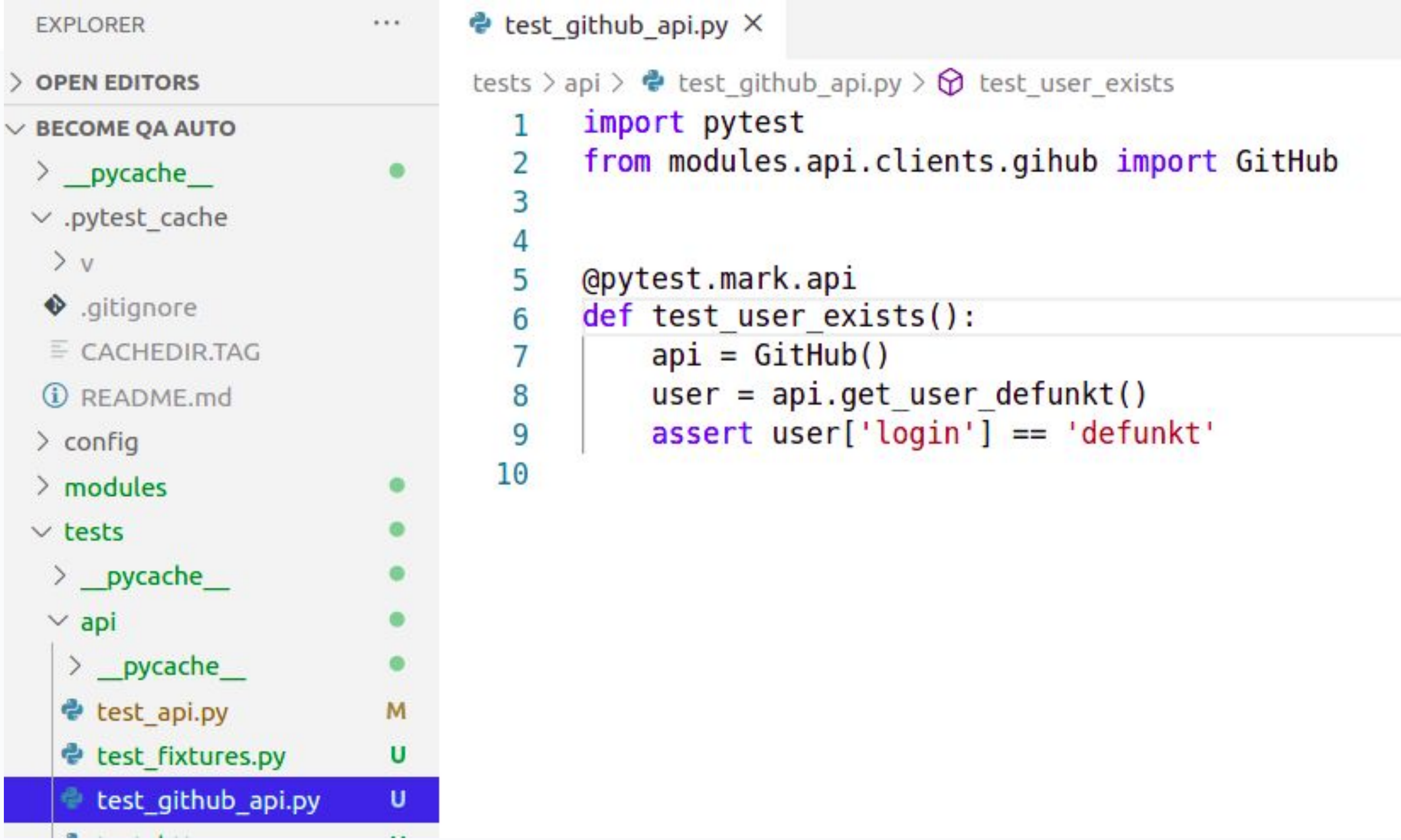
⌄ BECOME QA AUTO

- › .pytest_cache
- › config
- › modules ●
- ⌄ tests ●
  - › __pycache__ ●
  - ⌄ api ●
    - › __pycache__ ●
    - test_api.py M
    - test_fixtures.py U
    - test_github_api.py U

test_github_api.py ×

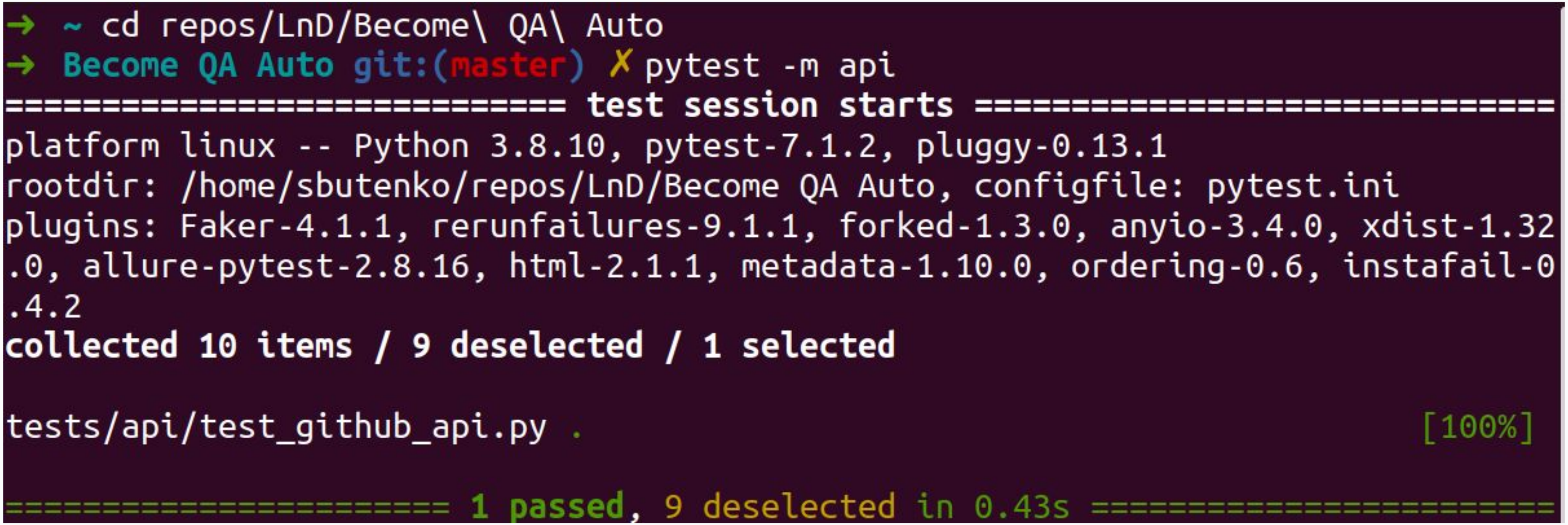
tests › api › test_github_api.py

Практика



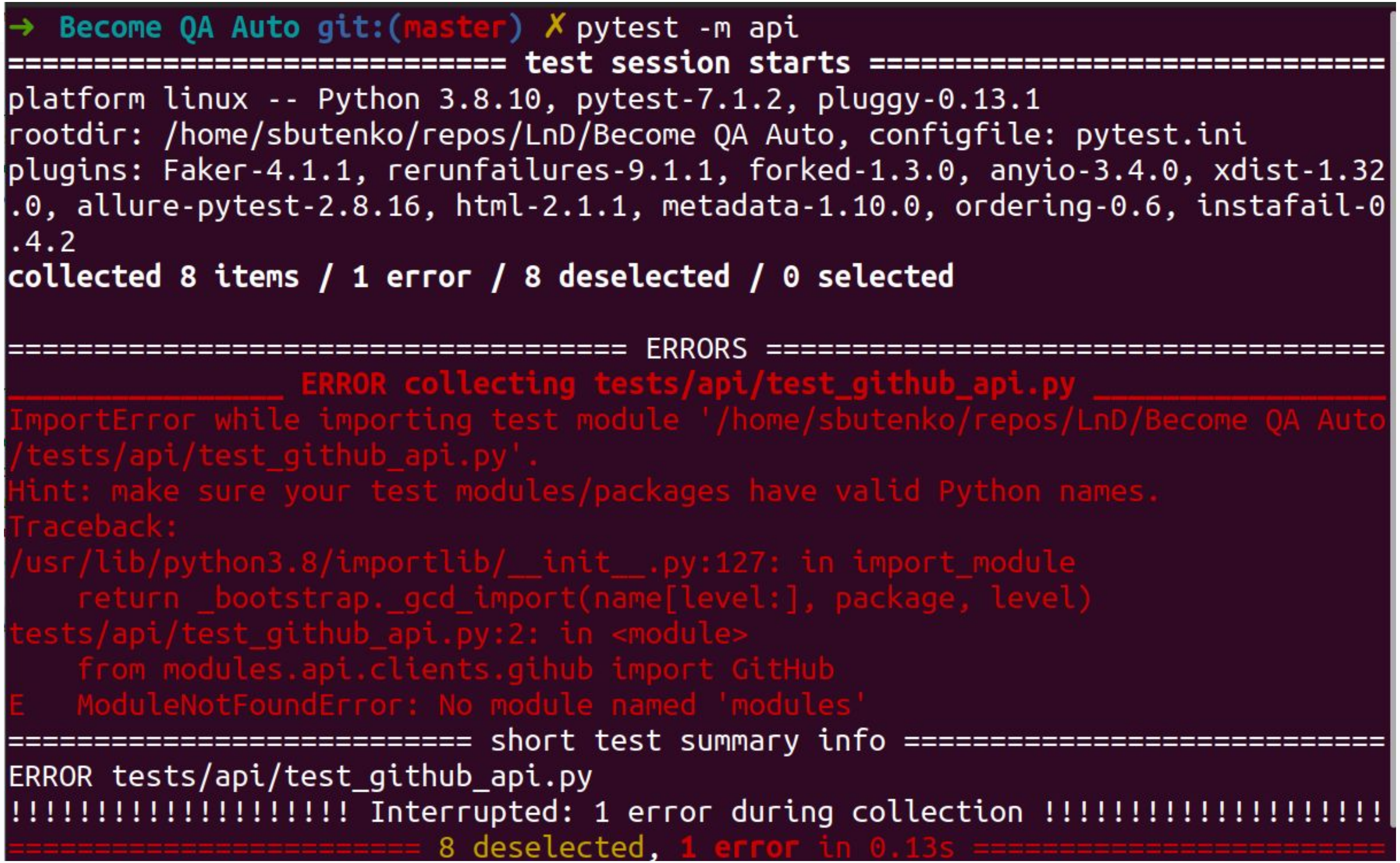
```python
import pytest
from modules.api.clients.gihub import GitHub


@pytest.mark.api
def test_user_exists():
    api = GitHub()
    user = api.get_user_defunkt()
    assert user['login'] == 'defunkt'

```

Практика



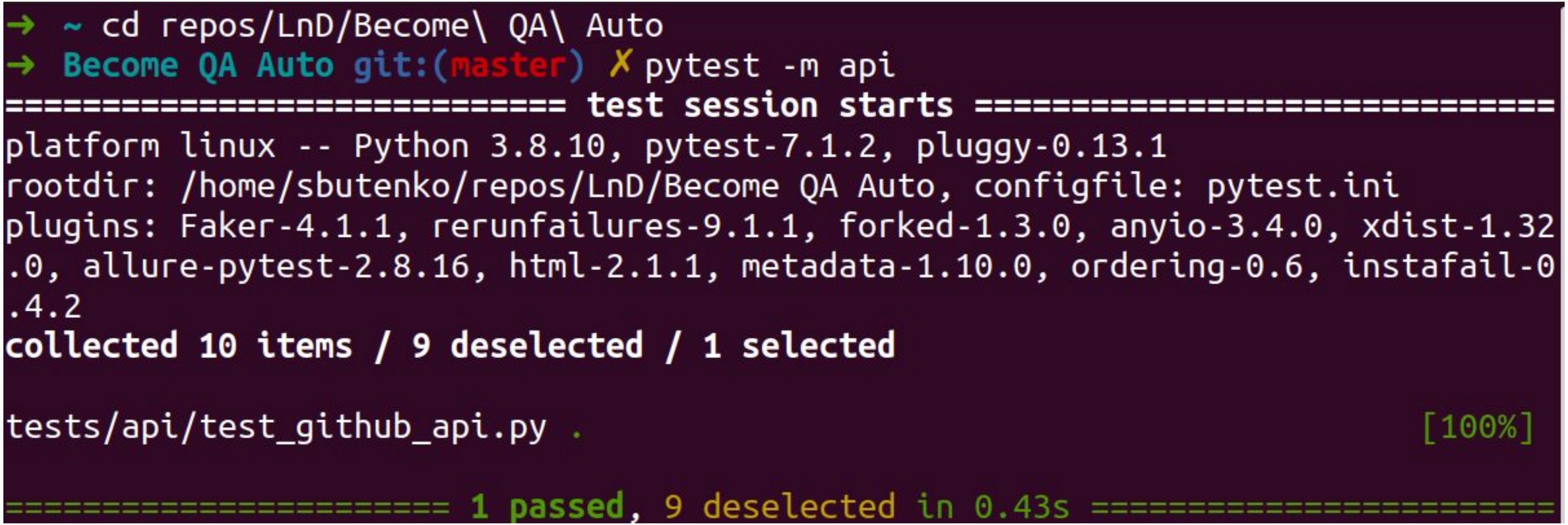
```
➜  ~ cd repos/LnD/Become\ QA\ Auto
➜  Become QA Auto git:(master) ✗ pytest -m api
============================ test session starts =============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: 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 10 items / 9 deselected / 1 selected

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

======================= 1 passed, 9 deselected in 0.43s =======================
```

Практика



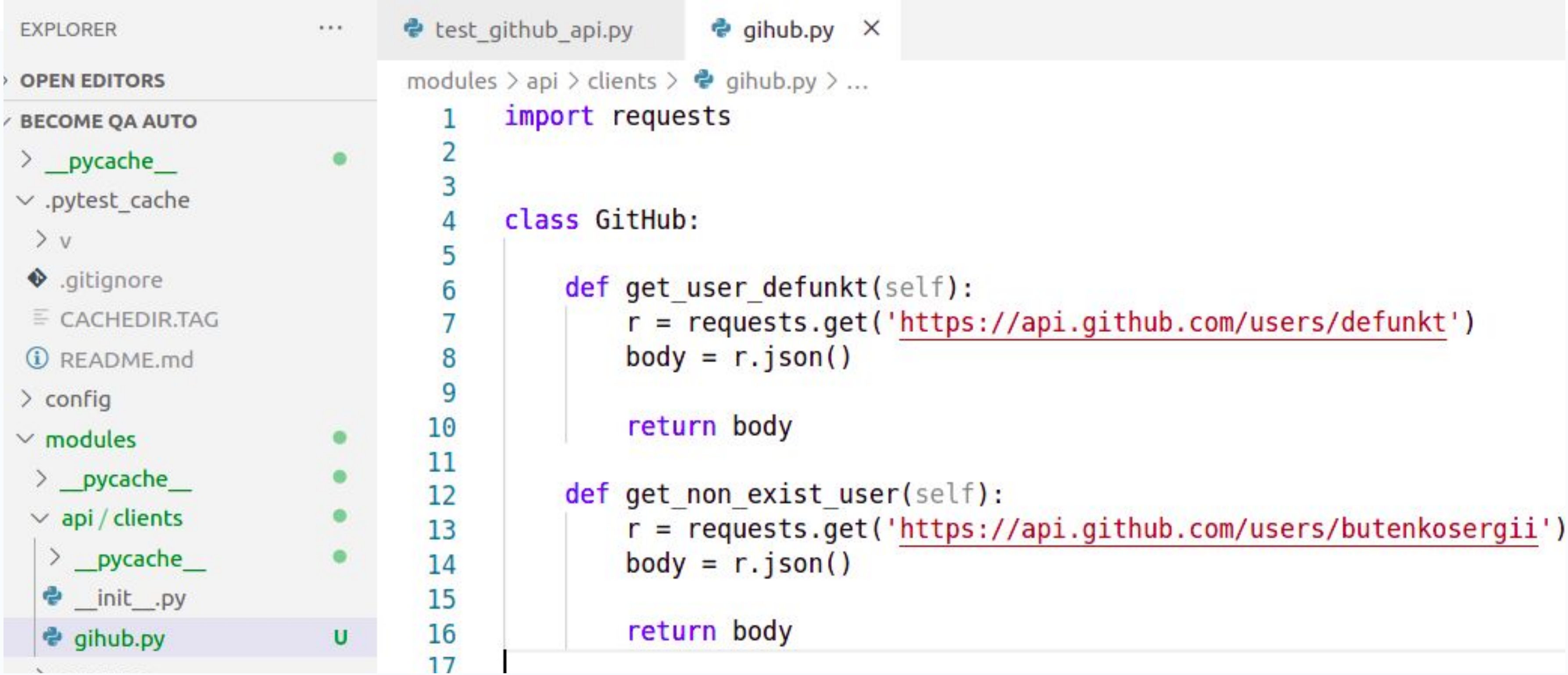
```
→ Become QA Auto git:(master) ✗ pytest -m api
============================== test session starts ===============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, 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 8 items / 1 error / 8 deselected / 0 selected

==================================== ERRORS ====================================
________________ ERROR collecting tests/api/test_github_api.py _________________
ImportError while importing test module '/home/sbutenko/repos/LnD/Become QA Auto
/tests/api/test_github_api.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
/usr/lib/python3.8/importlib/__init__.py:127: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
tests/api/test_github_api.py:2: in <module>
    from modules.api.clients.gihub import GitHub
E   ModuleNotFoundError: No module named 'modules'
=========================== short test summary info ============================
ERROR tests/api/test_github_api.py
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!
======================== 8 deselected, 1 error in 0.13s ========================
```

Практика



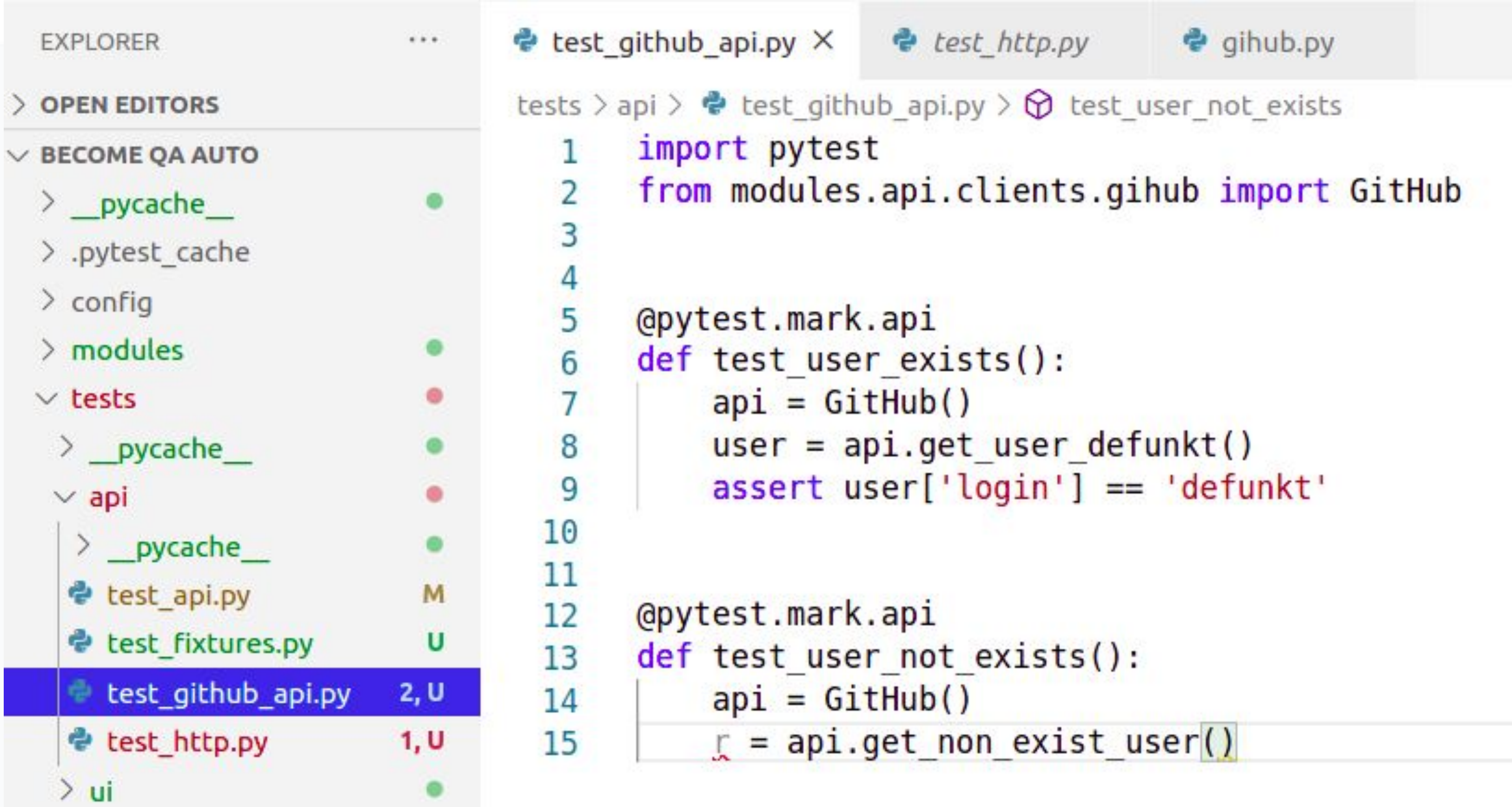
```
➜  ~ cd repos/LnD/Become\ QA\ Auto
➜  Become QA Auto git:(master) ✗ pytest -m api
============================ test session starts =============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: 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 10 items / 9 deselected / 1 selected

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

====================== 1 passed, 9 deselected in 0.43s =======================
```

Практика



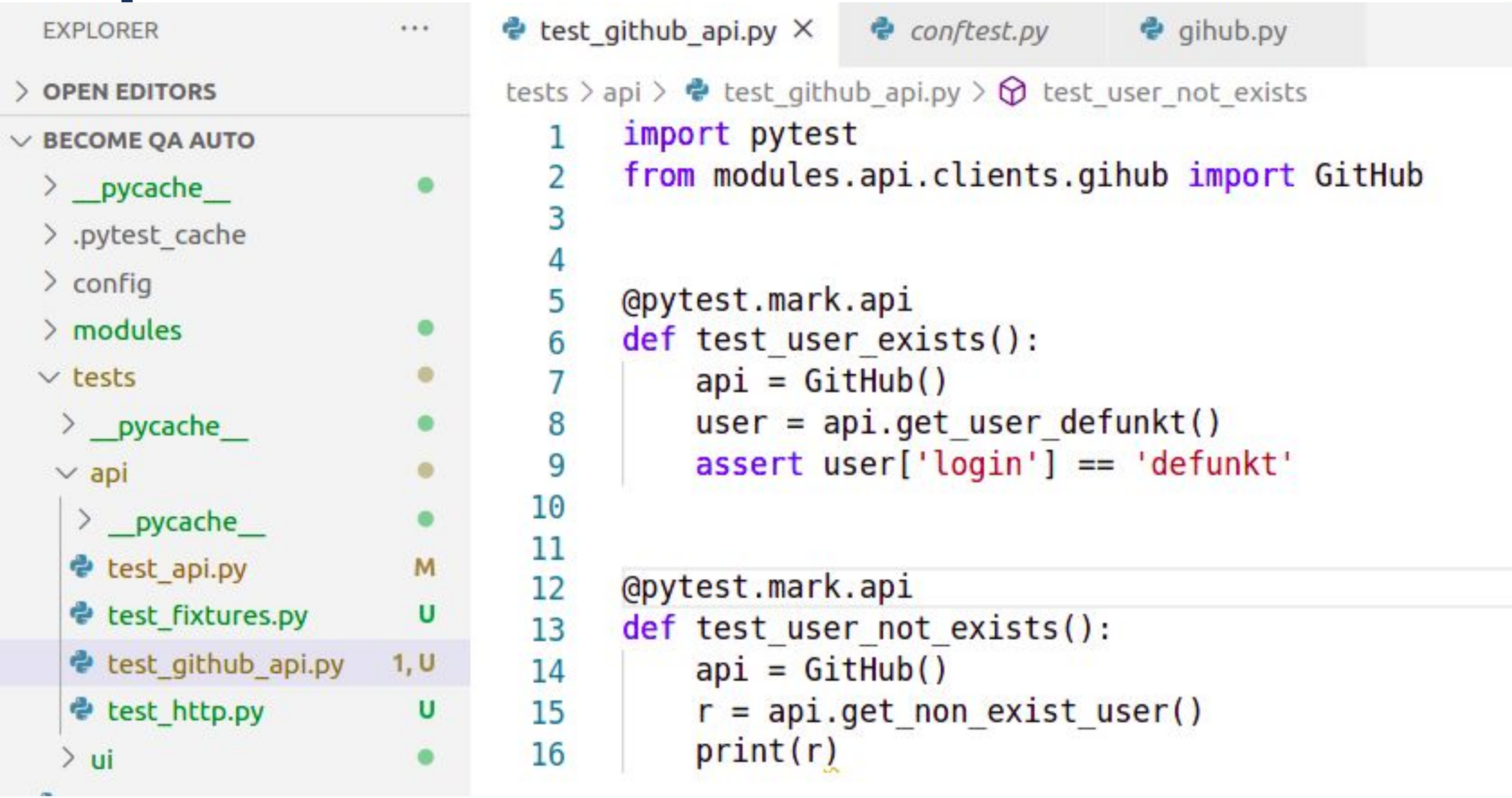
```python
import requests


class GitHub:

    def get_user_defunkt(self):
        r = requests.get('https://api.github.com/users/defunkt')
        body = r.json()

        return body

    def get_non_exist_user(self):
        r = requests.get('https://api.github.com/users/butenkosergii')
        body = r.json()

        return body

```

Практика



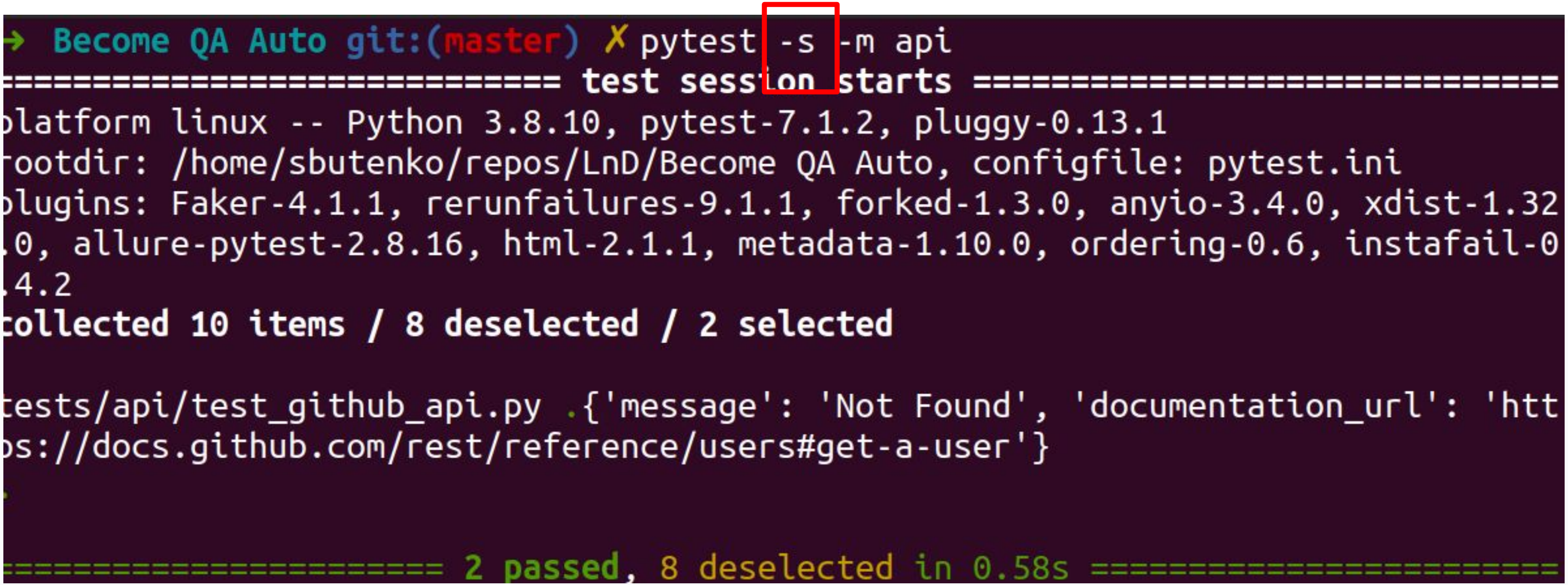
```python
import pytest
from modules.api.clients.gihub import GitHub


@pytest.mark.api
def test_user_exists():
    api = GitHub()
    user = api.get_user_defunkt()
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists():
    api = GitHub()
    r = api.get_non_exist_user()
```

Практика



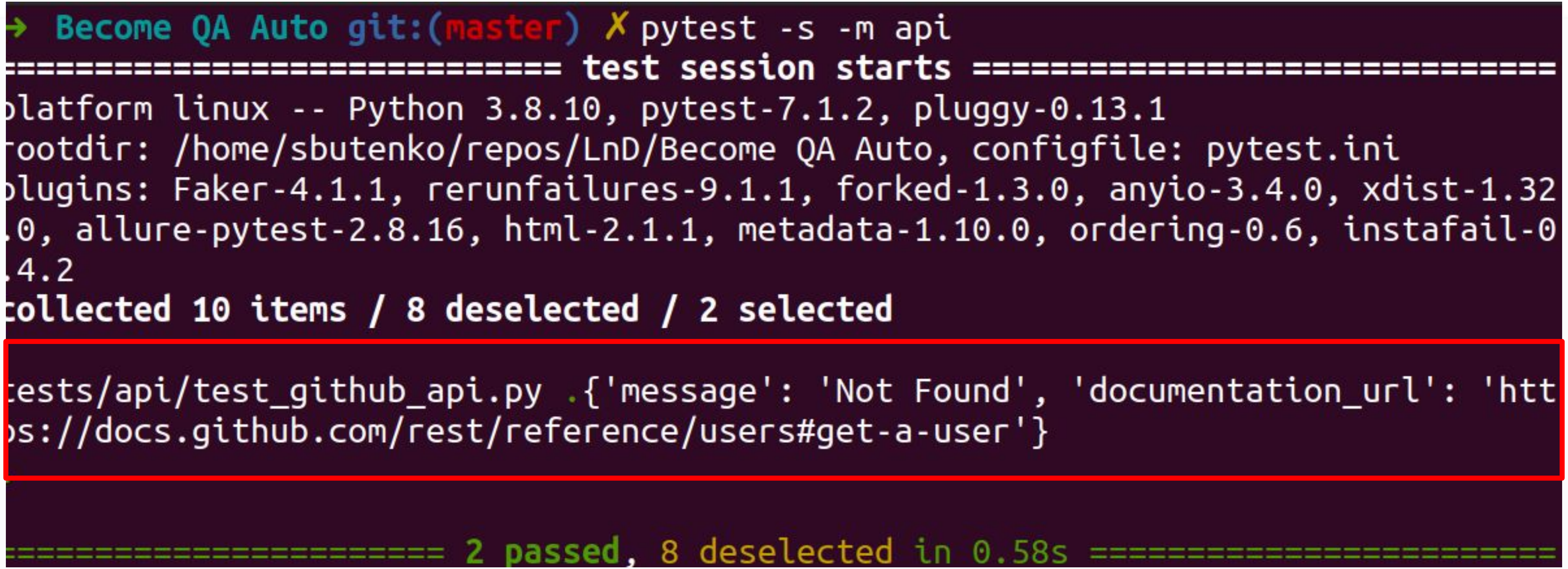
```
import pytest
from modules.api.clients.gihub import GitHub


@pytest.mark.api
def test_user_exists():
    api = GitHub()
    user = api.get_user_defunkt()
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists():
    api = GitHub()
    r = api.get_non_exist_user()
    print(r)
```

Практика



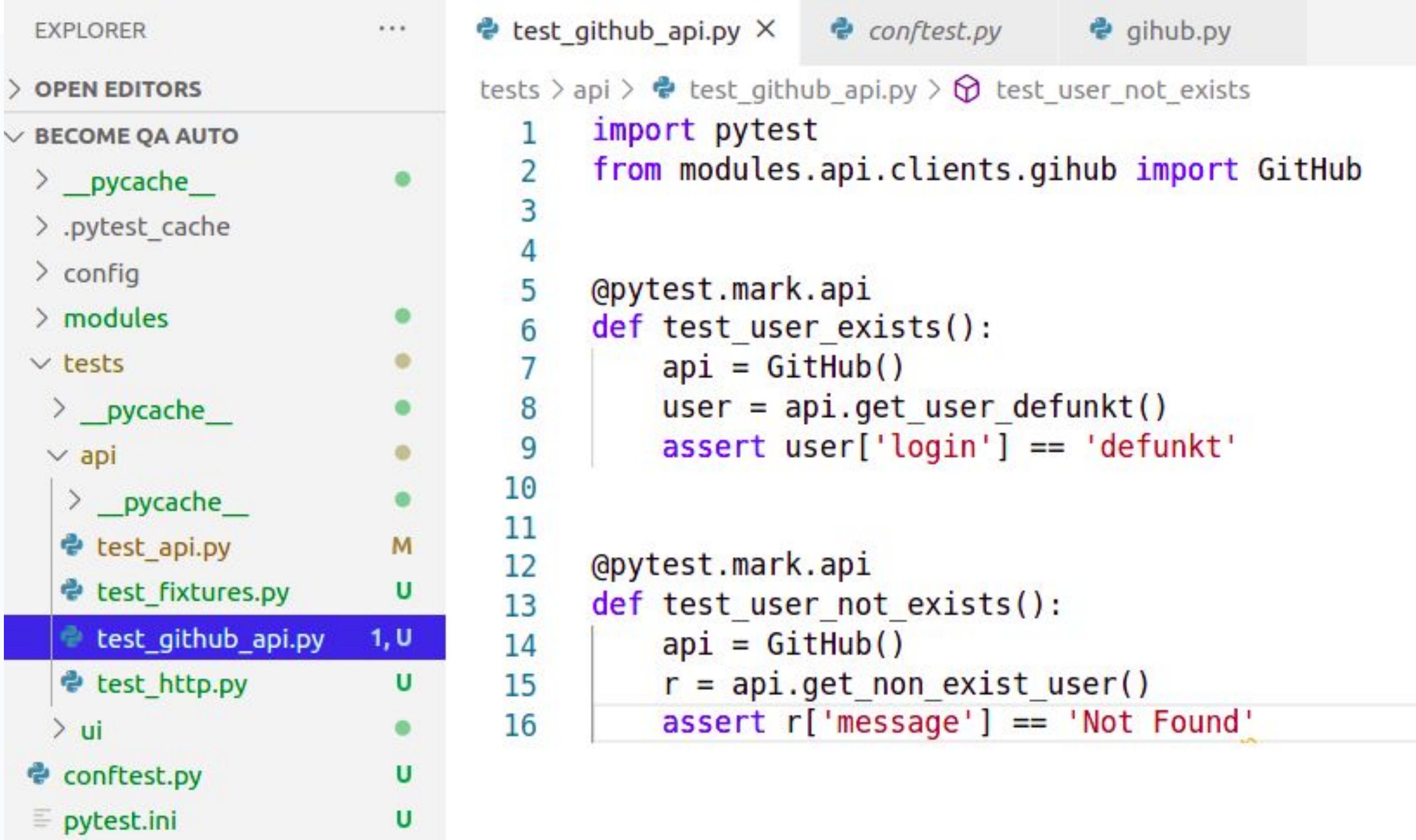
```
→ Become QA Auto git:(master) ✗ pytest -s -m api
============================== test session starts ===============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, 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 10 items / 8 deselected / 2 selected

tests/api/test_github_api.py .{'message': 'Not Found', 'documentation_url': 'htt
ps://docs.github.com/rest/reference/users#get-a-user'}
.

======================== 2 passed, 8 deselected in 0.58s =========================
```

Практика



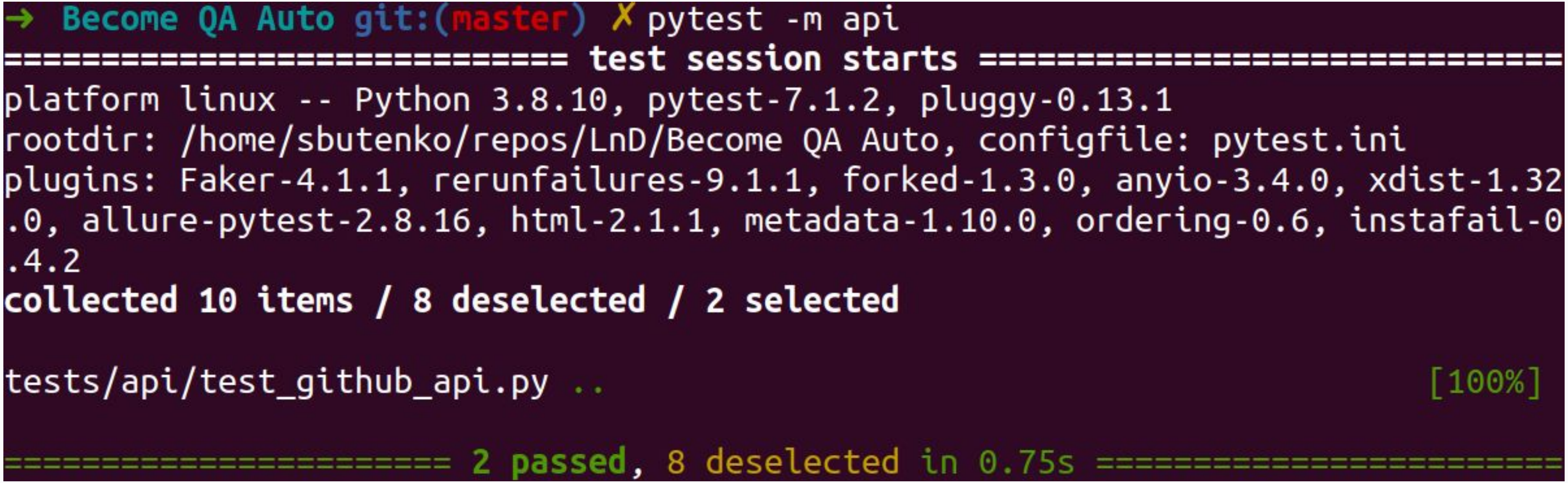
```
→ Become QA Auto git:(master) ✗ pytest -s -m api
============================== test session starts ===============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, 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 10 items / 8 deselected / 2 selected

tests/api/test_github_api.py .{'message': 'Not Found', 'documentation_url': 'htt
ps://docs.github.com/rest/reference/users#get-a-user'}


======================= 2 passed, 8 deselected in 0.58s ==========================
```

Практика



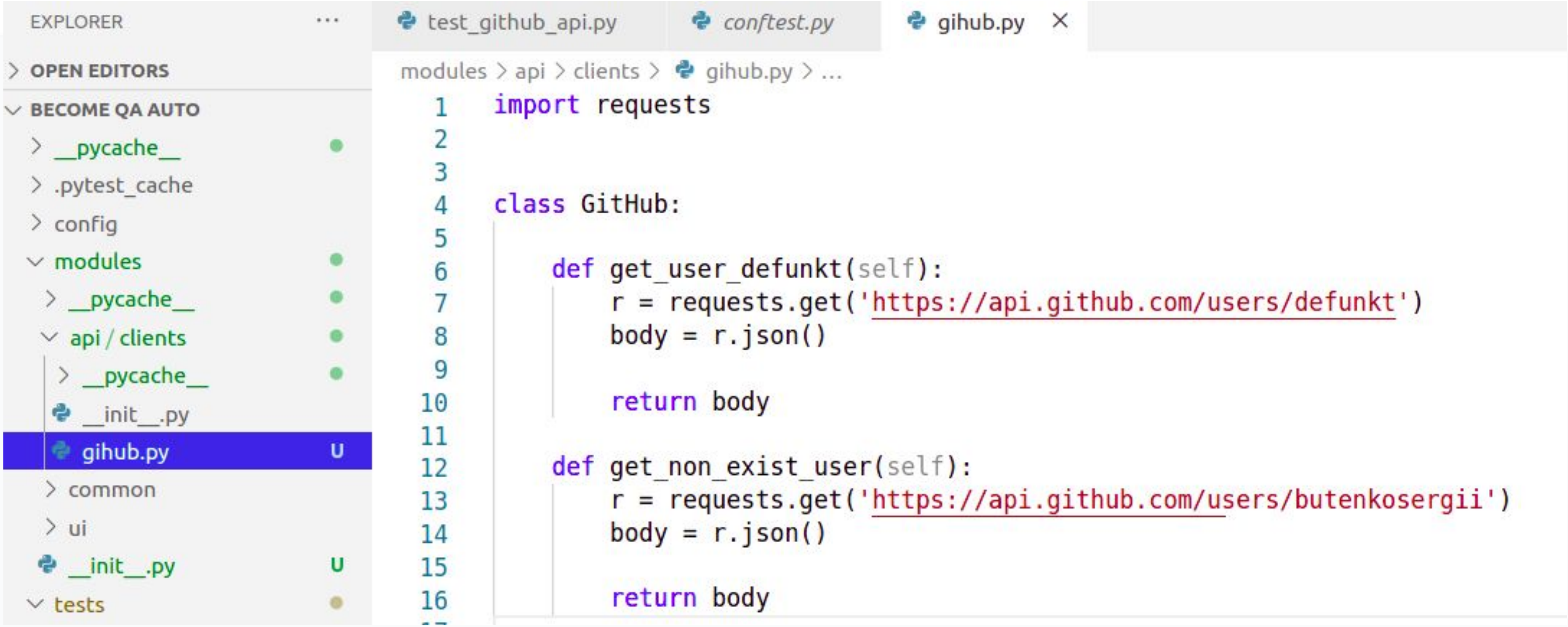
```python
import pytest
from modules.api.clients.gihub import GitHub


@pytest.mark.api
def test_user_exists():
    api = GitHub()
    user = api.get_user_defunkt()
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists():
    api = GitHub()
    r = api.get_non_exist_user()
    assert r['message'] == 'Not Found'
```

Практика



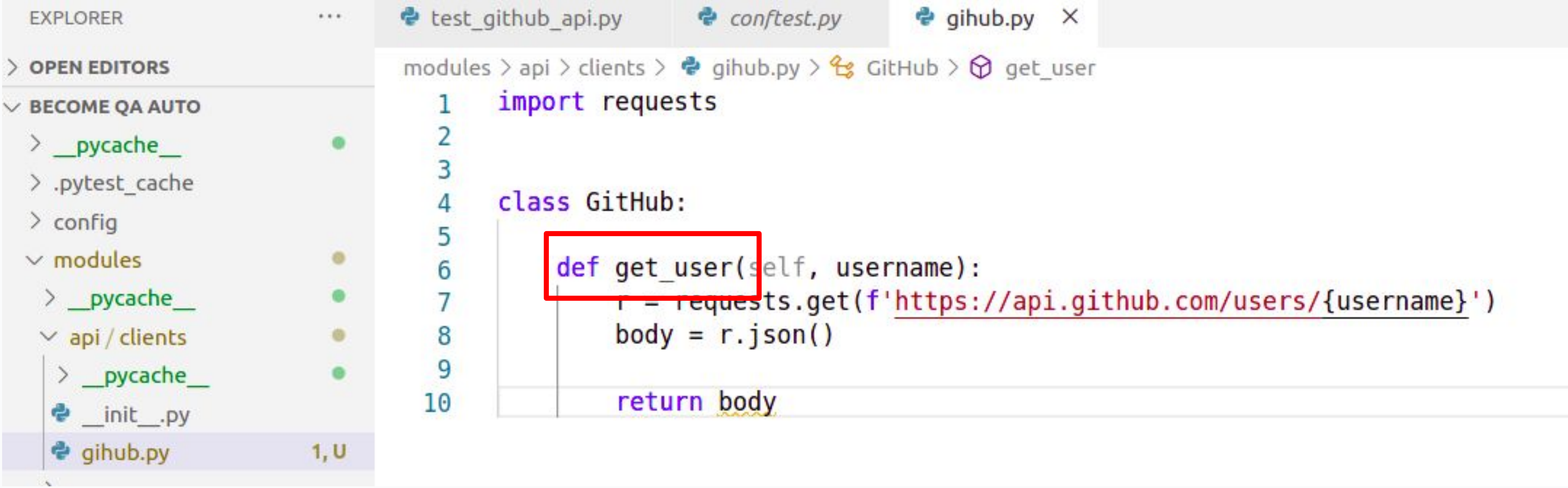
```
➜  Become QA Auto git:(master) ✗ pytest -m api
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, 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 10 items / 8 deselected / 2 selected

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

======================= 2 passed, 8 deselected in 0.75s ========================
```

Практика



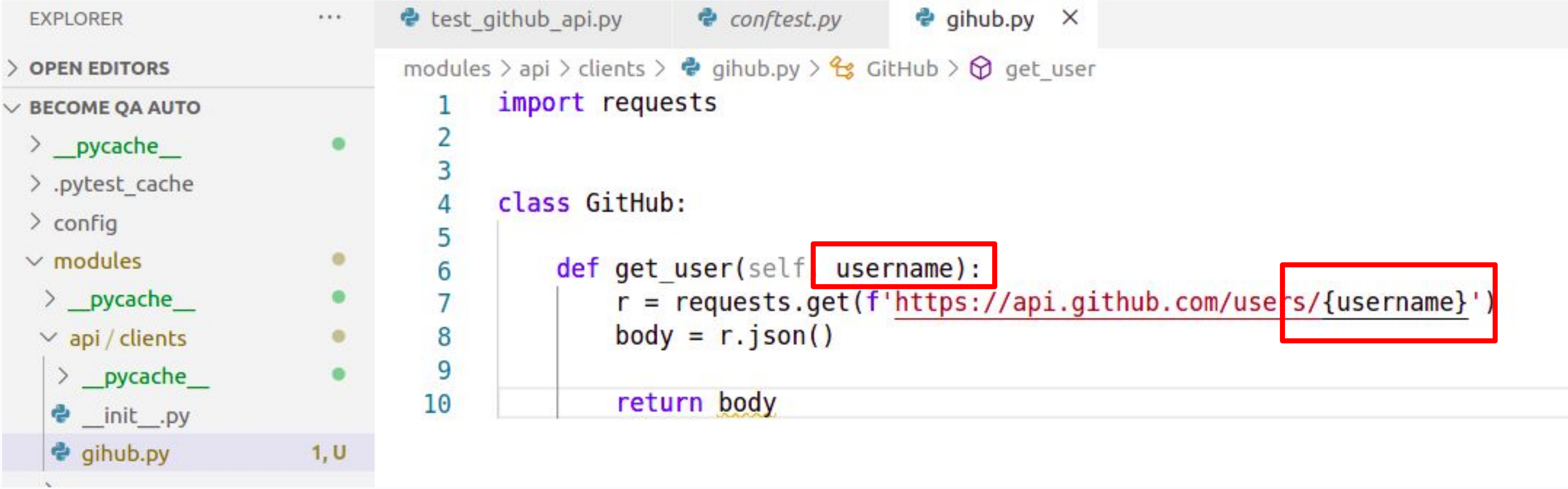
```python
import requests


class GitHub:

    def get_user_defunkt(self):
        r = requests.get('https://api.github.com/users/defunkt')
        body = r.json()

        return body

    def get_non_exist_user(self):
        r = requests.get('https://api.github.com/users/butenkosergii')
        body = r.json()

        return body
```

Практика



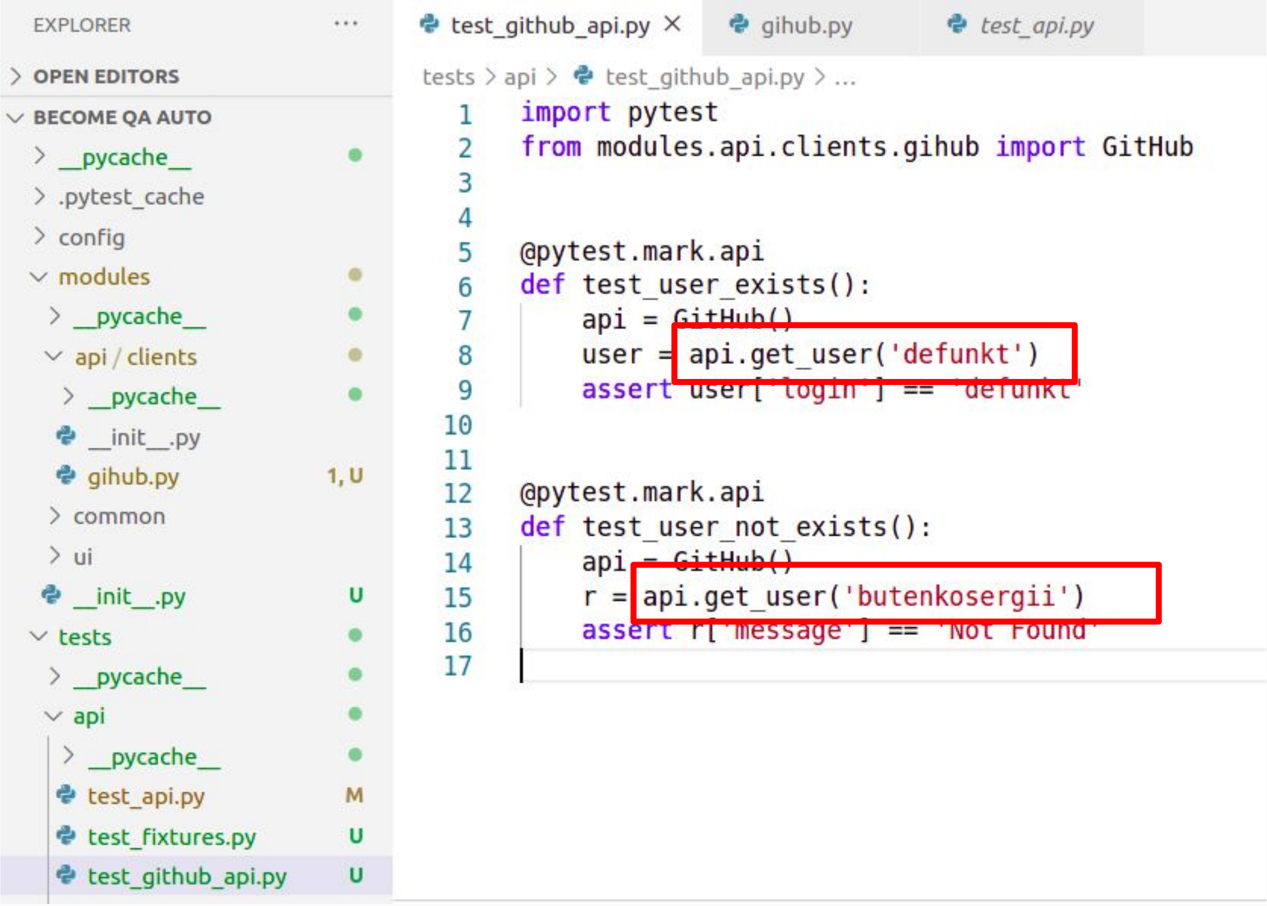
```python
import requests


class GitHub:

    def get_user(self, username):
        r = requests.get(f'https://api.github.com/users/{username}')
        body = r.json()

        return body
```

Практика



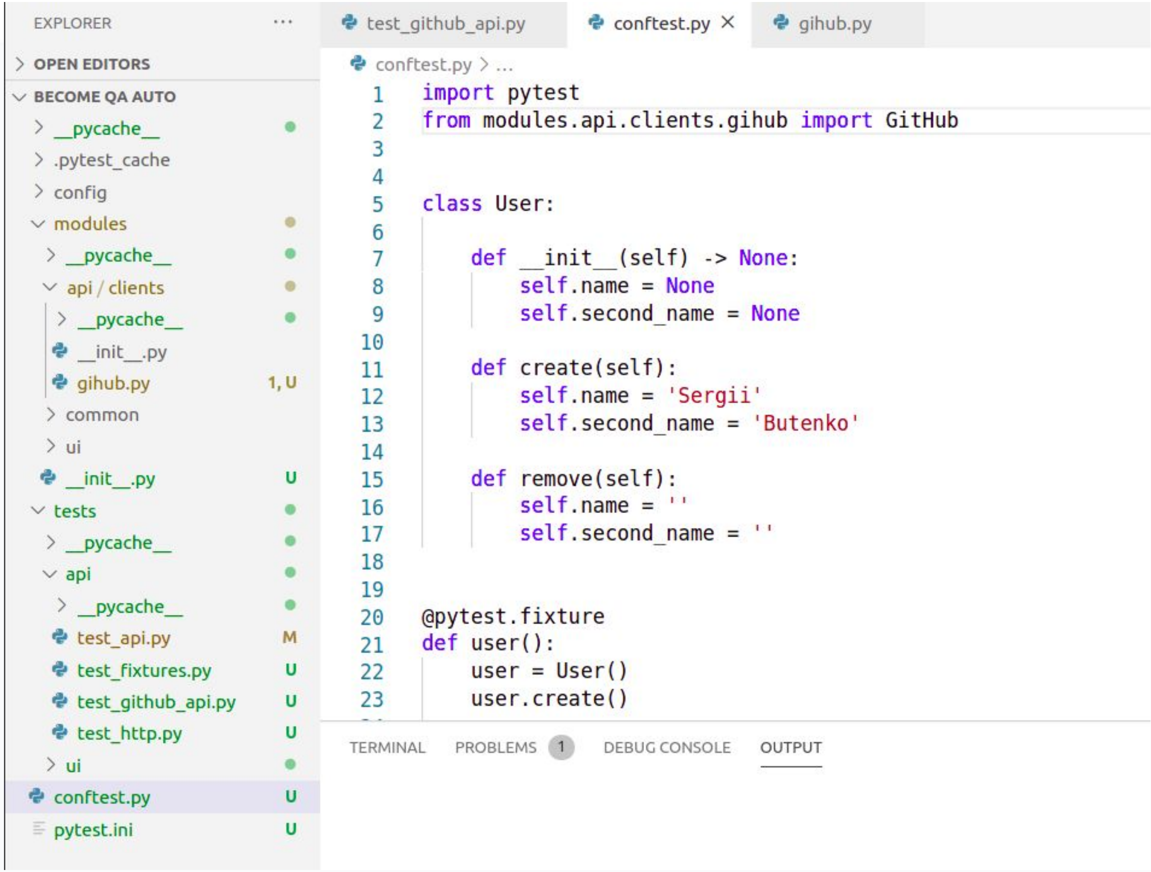
```python
import requests


class GitHub:

    def get_user(self, username):
        r = requests.get(f'https://api.github.com/users/{username}')
        body = r.json()

        return body
```

Практика



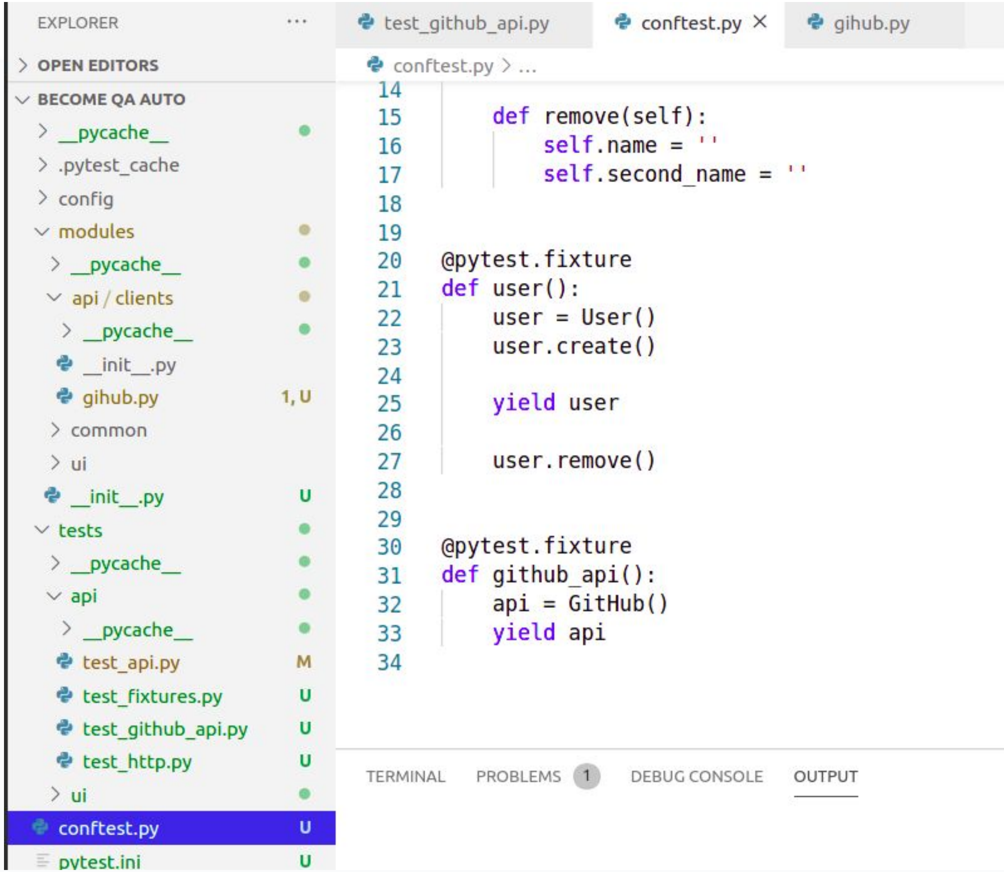
```python
import pytest
from modules.api.clients.gihub import GitHub


@pytest.mark.api
def test_user_exists():
    api = GitHub()
    user = api.get_user('defunkt')
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists():
    api = GitHub()
    r = api.get_user('butenkosergii')
    assert r['message'] == 'Not Found'

```

Практика



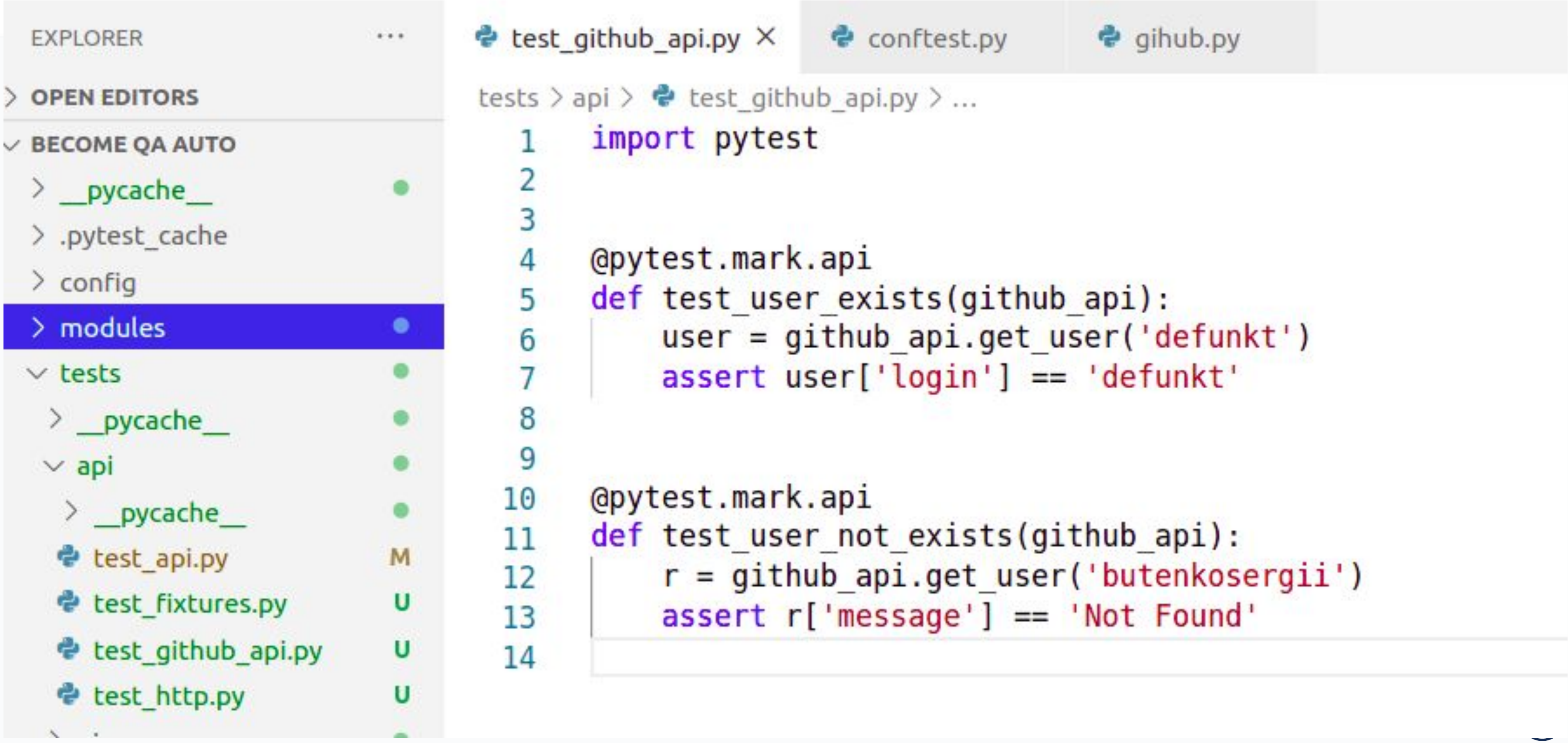
```python
import pytest
from modules.api.clients.gihub import GitHub


class User:

    def __init__(self) -> None:
        self.name = None
        self.second_name = None

    def create(self):
        self.name = 'Sergii'
        self.second_name = 'Butenko'

    def remove(self):
        self.name = ''
        self.second_name = ''


@pytest.fixture
def user():
    user = User()
    user.create()
```

Практика



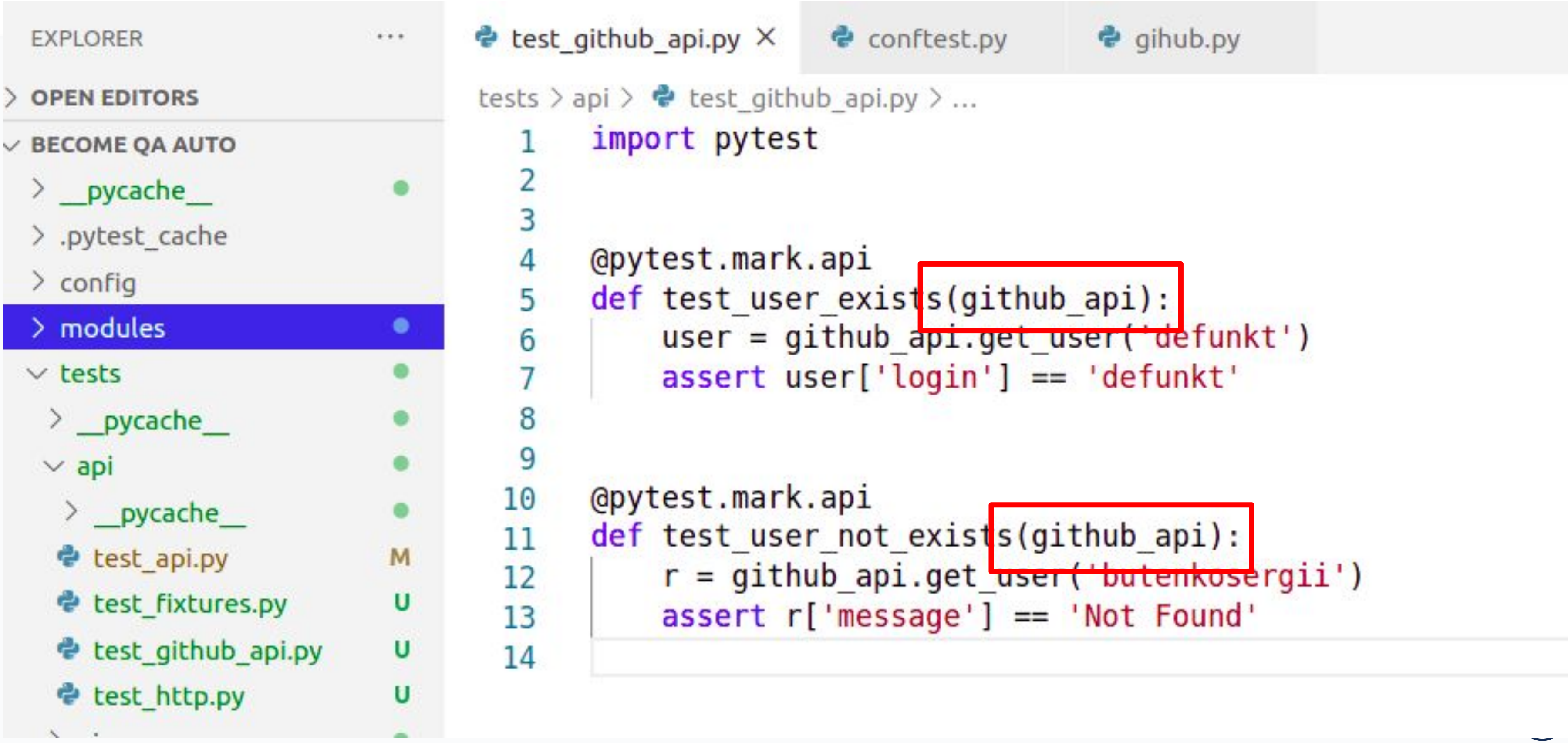
```python

    def remove(self):
        self.name = ''
        self.second_name = ''


@pytest.fixture
def user():
    user = User()
    user.create()

    yield user

    user.remove()


@pytest.fixture
def github_api():
    api = GitHub()
    yield api

```

Практика



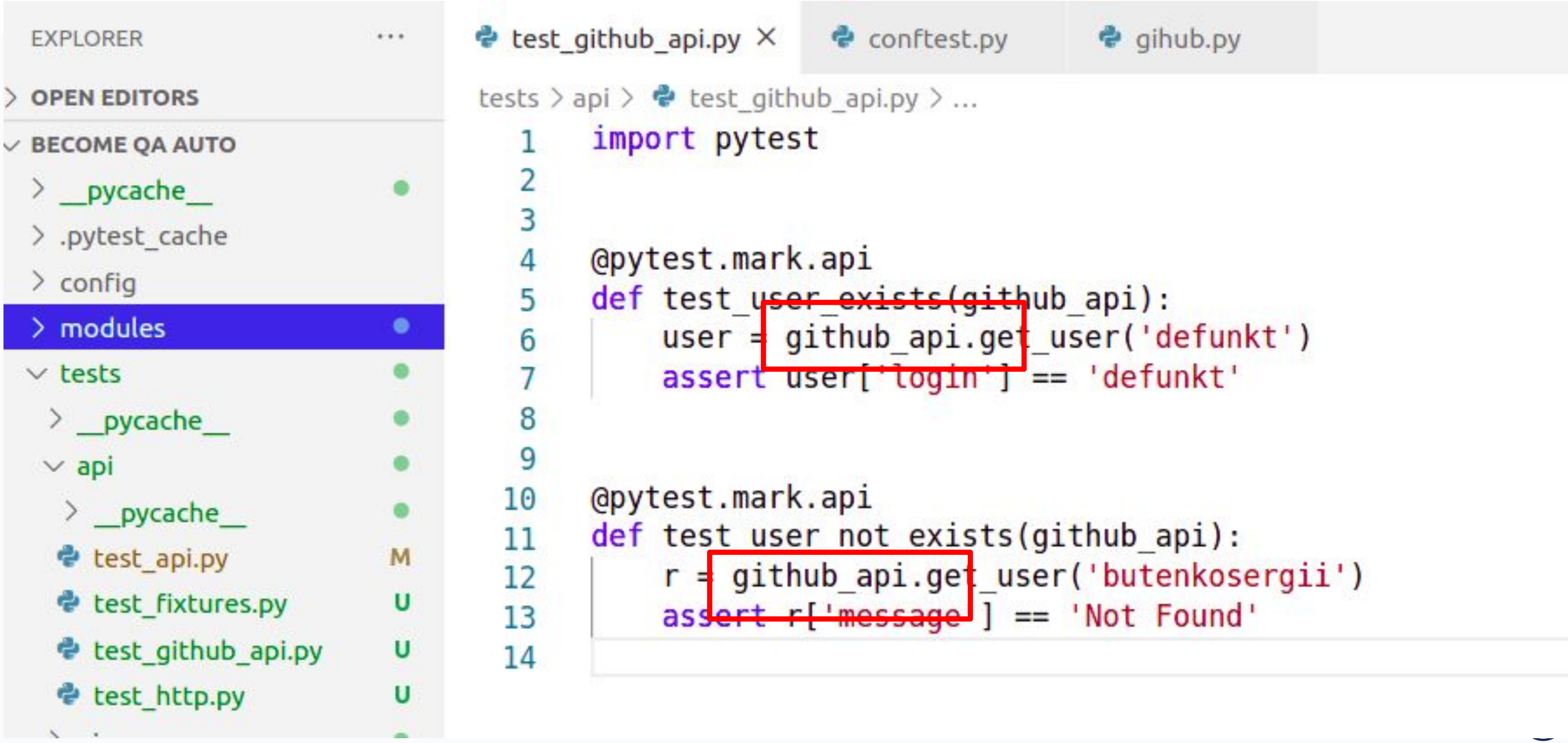
```python
import pytest


@pytest.mark.api
def test_user_exists(github_api):
    user = github_api.get_user('defunkt')
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists(github_api):
    r = github_api.get_user('butenkosergii')
    assert r['message'] == 'Not Found'

```

Практика



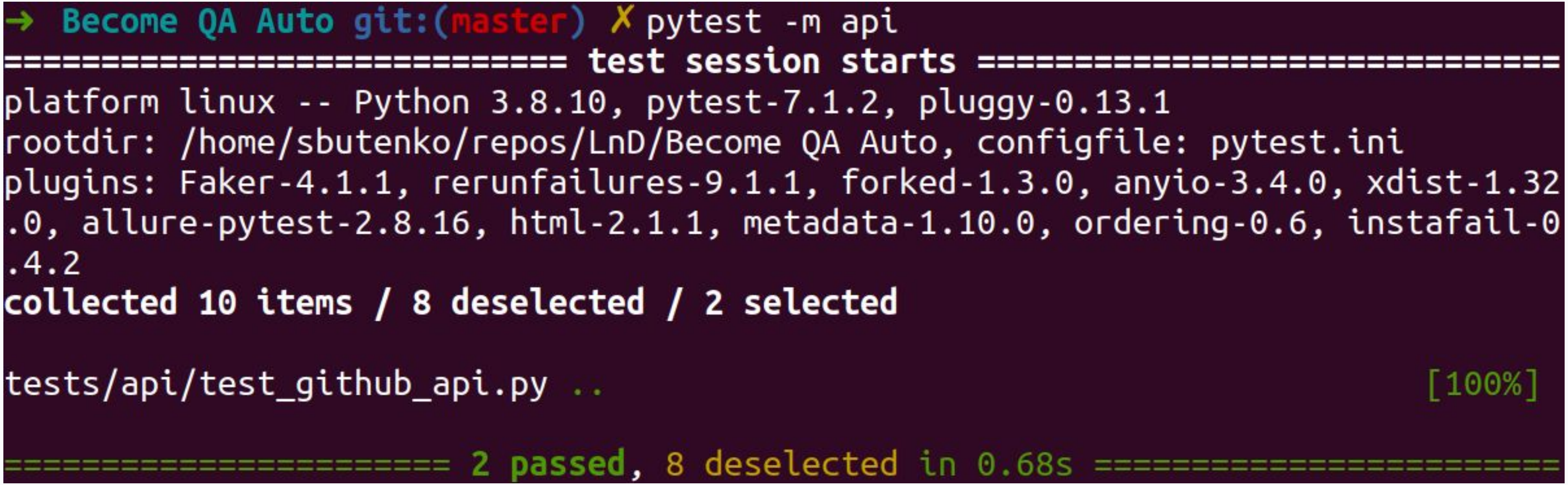
```python
import pytest


@pytest.mark.api
def test_user_exists(github_api):
    user = github_api.get_user('defunkt')
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists(github_api):
    r = github_api.get_user('butenkosergii')
    assert r['message'] == 'Not Found'
```

Практика



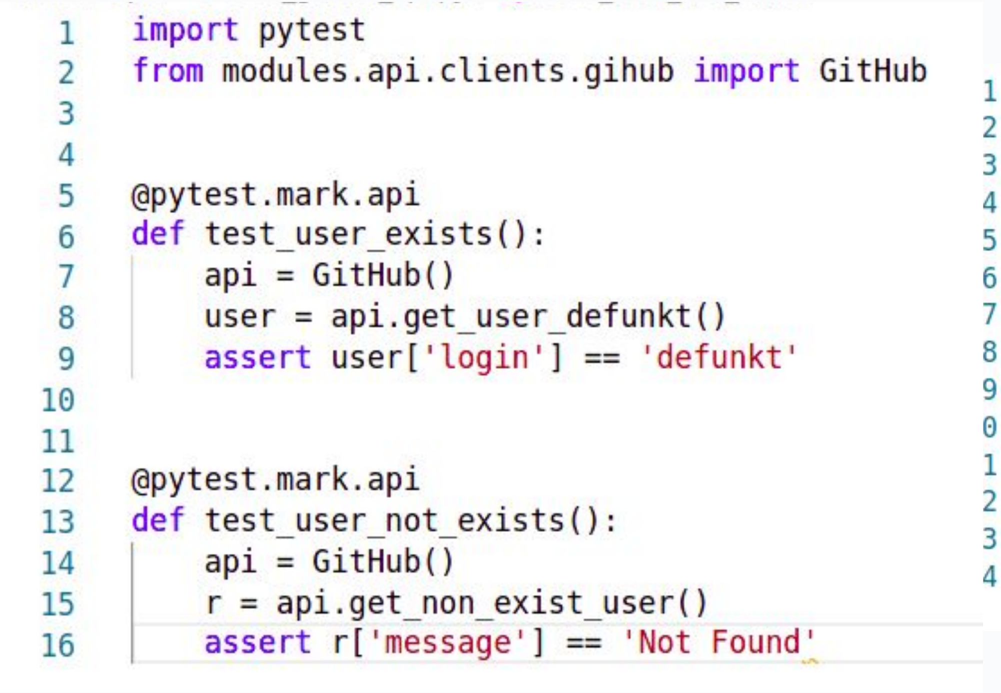
```python
import pytest


@pytest.mark.api
def test_user_exists(github_api):
    user = github_api.get_user('defunkt')
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists(github_api):
    r = github_api.get_user('butenkosergii')
    assert r['message'] == 'Not Found'

```

Практика

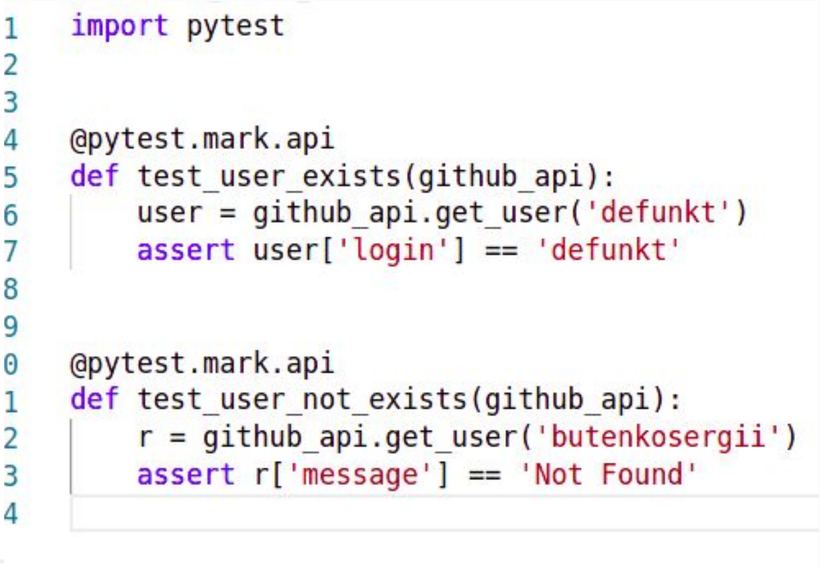


```
➜  Become QA Auto git:(master) ✗ pytest -m api
============================ test session starts ============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, 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 10 items / 8 deselected / 2 selected

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

======================= 2 passed, 8 deselected in 0.68s =======================
```

Практика

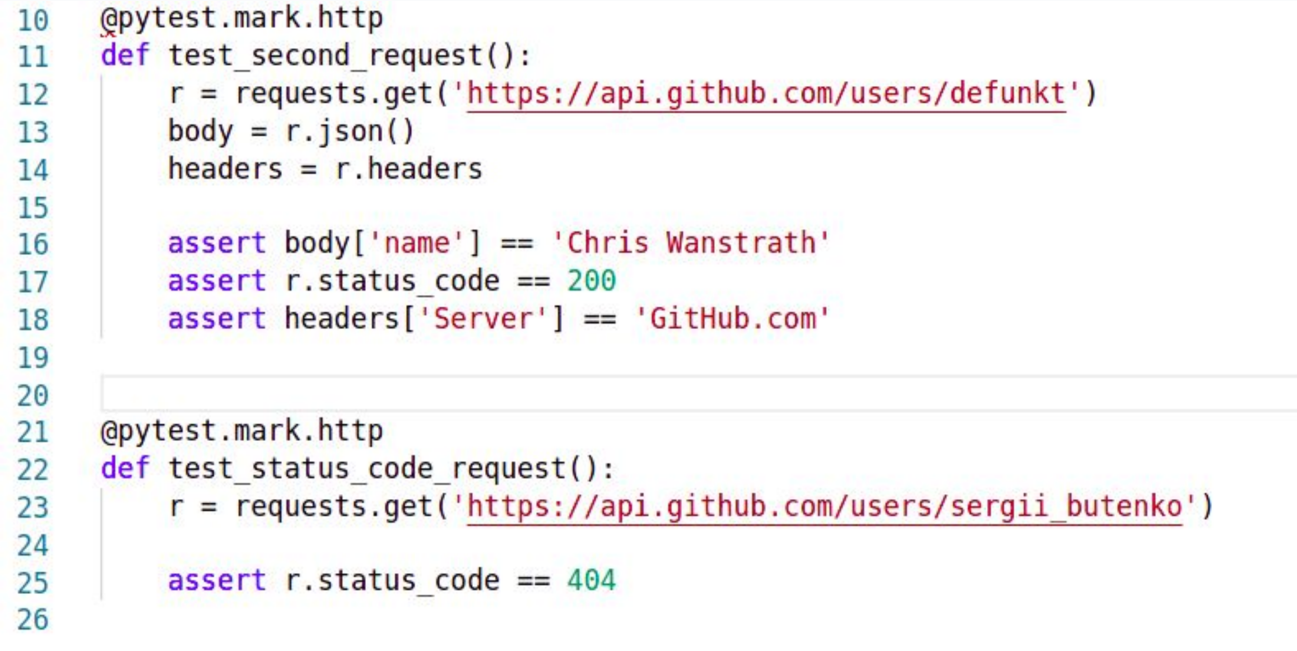


```python
import pytest
from modules.api.clients.gihub import GitHub


@pytest.mark.api
def test_user_exists():
    api = GitHub()
    user = api.get_user_defunkt()
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists():
    api = GitHub()
    r = api.get_non_exist_user()
    assert r['message'] == 'Not Found'
```



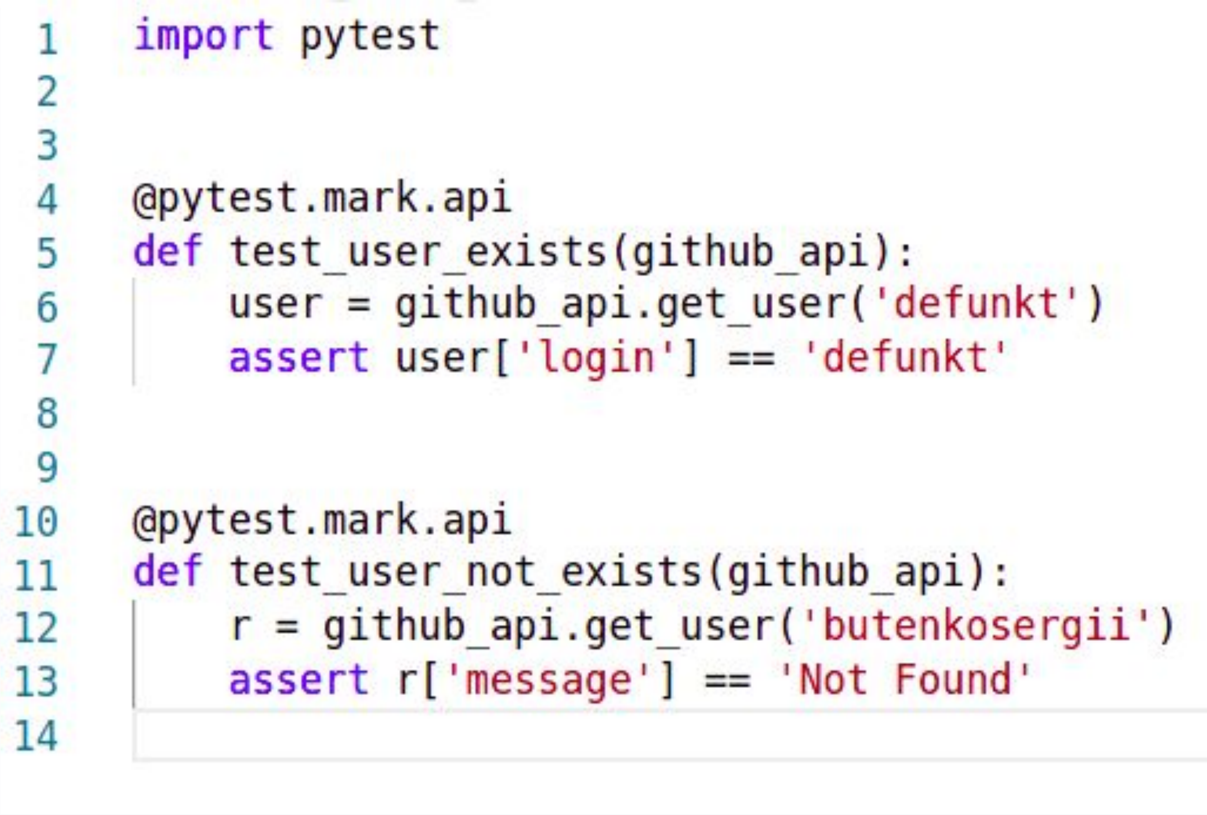
```python
import pytest


@pytest.mark.api
def test_user_exists(github_api):
    user = github_api.get_user('defunkt')
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists(github_api):
    r = github_api.get_user('butenkosergii')
    assert r['message'] == 'Not Found'
```

Практика



```python
@pytest.mark.http
def test_second_request():
    r = requests.get('https://api.github.com/users/defunkt')
    body = r.json()
    headers = r.headers

    assert body['name'] == 'Chris Wanstrath'
    assert r.status_code == 200
    assert headers['Server'] == 'GitHub.com'


@pytest.mark.http
def test_status_code_request():
    r = requests.get('https://api.github.com/users/sergii_butenko')

    assert r.status_code == 404
```

Практика



```python
import pytest


@pytest.mark.api
def test_user_exists(github_api):
    user = github_api.get_user('defunkt')
    assert user['login'] == 'defunkt'


@pytest.mark.api
def test_user_not_exists(github_api):
    r = github_api.get_user('butenkosergii')
    assert r['message'] == 'Not Found'

```

Підсумки

✓ Навчилися:

1. **Створювати** власний API клієнт на базі API документації
2. **Використовувати** API клієнт в тестах

✓ З’ясували що використанні API клієнту в тестах допомагає нам в:

1. **Зменшенні** кількості повторювального кода в тестах
2. **Спрощує** подальшу підтримку тестів
3. Дозволяє **ефективно використовувати** фікстури в тестах