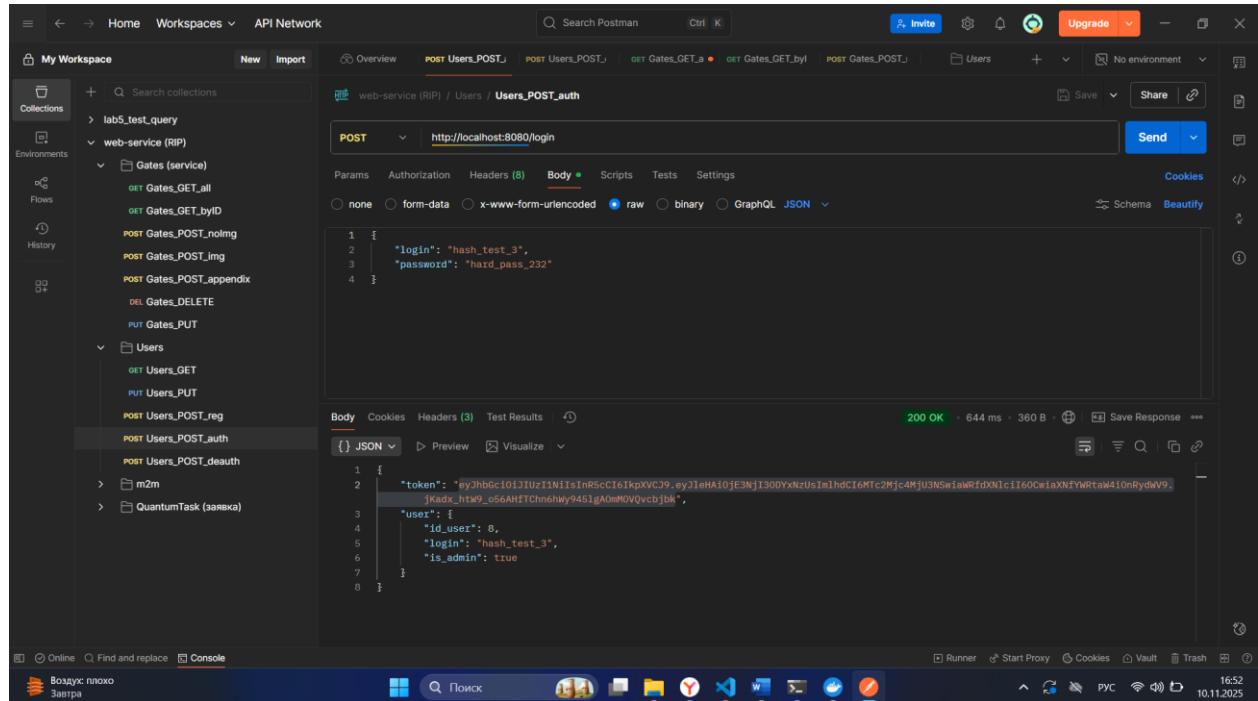


# Исправлено после демонстрации

Установлено приложение



POST <http://localhost:8080/login>

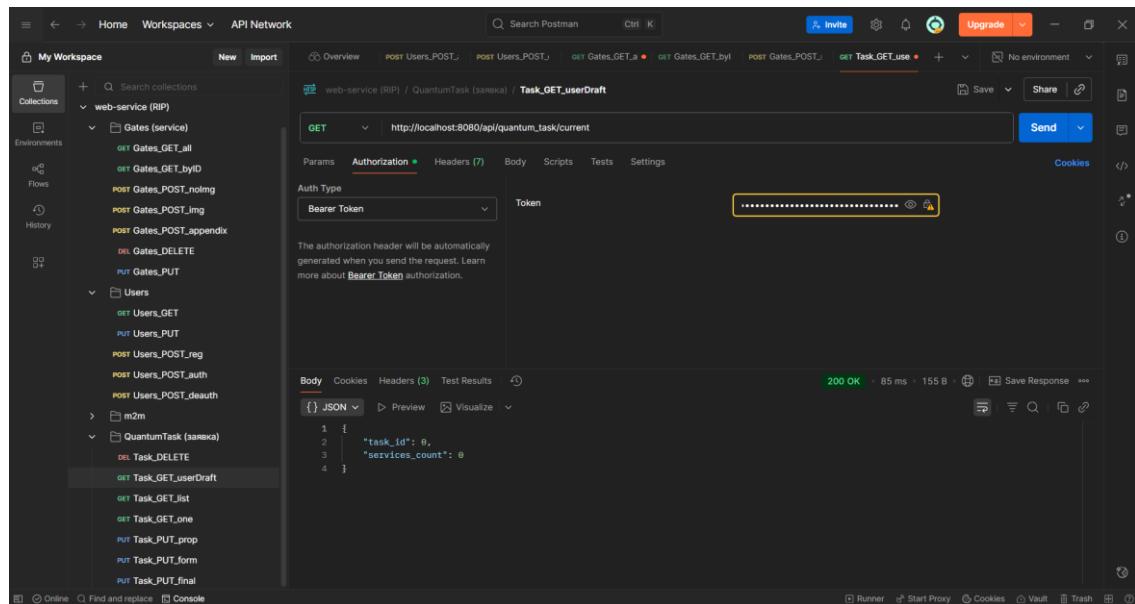
Body

```
{ "login": "hash_test_3", "password": "haid_pass_232" }
```

200 OK

```
{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NjI3ODYxNzUsImhdCT6Mtc2Mjc4MjU3NSwiaWFzdXNlIi60OwiAxNfYWRtaW4iOnRydWV9.[KdxLhtw_056AHFTchmsh9451gA0mM0VQvcb]bk", "user": { "id": 8, "login": "hash_test_3", "is_admin": true }}
```

Убран ненужный статус



GET <http://localhost:8080/api/quantum.task/current>

Auth Type

Bearer Token

Token

200 OK

```
{ "task_id": 0, "services_count": 0 }
```

Убран пароль и другие лишние поля при возвращении пользовательских данных

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, Flows, History.
- Request URL:** http://localhost:8080/api/users/:id
- Method:** GET
- Body:** JSON (empty)
- Response:** 200 OK, 12 ms, 140 B. The response body is a JSON object with one key-value pair: "login": "user1".

Поля м-м и услуги в одной вложенности без лишних скобок.

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, Flows, History.
- Request URL:** http://localhost:8080/api/quantum\_tasks
- Method:** GET
- Body:** JSON (empty)
- Response:** 200 OK, 56 ms, 5.63 KB. The response body is a complex JSON object representing multiple tasks, each with fields like id\_task, task\_status, creation\_date, id\_user, conclusion\_date, task\_description, res\_koeff\_0, res\_koeff\_1, and gates\_degrees (which contains further nested objects).

## Коллекция из 21-го запроса в postman

**web-service (RIP)**

- Gates (service)
  - GET Service\_GET\_all
  - GET Service\_GET\_byID
  - POST Service\_POST\_noImg
  - POST Service\_POST\_img
  - POST Service\_POST\_appendix
  - DEL Service\_DELETE
  - PUT Service\_PUT
- Users
  - GET Users\_GET
  - PUT Users\_PUT
  - POST Users\_POST\_reg
  - POST Users\_POST\_auth
  - POST Users\_POST\_deauth
- m2m
  - DEL m2m\_DELETE
  - PUT m2m\_PUT

**QuantumTask (заявка)**

- Task\_DELETE
- GET Task\_GET\_userDraft
- GET Task\_GET\_list
- GET Task\_GET\_one
- PUT Task\_PUT\_prop
- PUT Task\_PUT\_form
- PUT Task\_PUT\_final

## GET списка заявок

GET /api/quantum\_tasks

Params: Authorization, Headers, Body, Scripts, Settings

Body: Pretty, Raw, Preview, JSON

```

1  [
2   "data": {
3     "items": [
4       {
5         "ID_task": 18,
6         "Taskstatus": "заявка",
7         "CreationDate": "2025-10-12T15:54:26.194924+03:00",
8         "ID_user": 1,
9         "ConclusionDate": "2025-10-12T15:54:54.271286+03:00",
10        "ID_moderator": 1,
11        "FormedDate": "2025-10-12T15:54:49.890848+03:00",
12        "TaskDescription": "turu-puru",
13        "Res_koeff_0": -1,
14        "Res_koeff_1": 0,
15        "GatesDegrees": [
16          {
17            "ID_gate": 3,
18            "ID_task": 18,
19            "Degrees": 45,
20            "Gate": {
21              "ID_gate": 3,
22              "ID_task": 18,
23              "Degrees": 45
24            }
25          }
26        ]
27      }
28    ]
29  }
30 }
```

Terminal: [GIN] 2025/10/13 - 01:55:38 | 200 | 5.721ms | :1: | GET "/api/quantum\_tasks/29"  
[GIN] 2025/10/13 - 02:02:16 | 200 | 86.9477ms | :1: | GET "/api/quantum\_tasks"

## GET иконки корзины

The screenshot shows the Postman interface with a successful HTTP request to `http://localhost:8080/api/quantum_task/current`. The response status is 200 OK, time 10 ms, and size 179 B. The response body is a JSON object:

```
[{"data": {"services_count": 2, "task_id": 28}, "status": "ok"}]
```

The terminal tab shows the command history:

```
[GIN] 2025/10/13 - 00:49:20 | 200 | 34.2746ms | ::1 | POST "/quantum_task/add/gate/2"
[GIN] 2025/10/13 - 00:49:21 | 200 | 14.4806ms | ::1 | GET "/18"
[GIN] 2025/10/13 - 00:49:23 | 200 | 20.1343ms | ::1 | POST "/quantum_task/add/gate/1"
[GIN] 2025/10/13 - 00:49:23 | 200 | 10.0619ms | ::1 | GET "/18"
[GIN] 2025/10/13 - 00:49:27 | 200 | 5.4355ms | ::1 | GET "/api/quantum_task/current"
```

## удалить введенную заявку

The screenshot shows the Postman interface with a successful DELETE request to `http://localhost:8080/api/quantum_tasks/28`. The response status is 200 OK, time 128 ms, and size 155 B. The response body is a JSON object:

```
[{"data": {"id": 28}, "status": "ok"}]
```

The terminal tab shows the command history:

```
[GIN] 2025/10/13 - 00:49:21 | 200 | 14.4806ms | ::1 | GET "/18"
[GIN] 2025/10/13 - 00:49:23 | 200 | 20.1343ms | ::1 | POST "/quantum_task/add/gate/1"
[GIN] 2025/10/13 - 00:49:23 | 200 | 10.0619ms | ::1 | GET "/18"
[GIN] 2025/10/13 - 00:49:27 | 200 | 5.4355ms | ::1 | GET "/api/quantum_task/current"
[GIN] 2025/10/13 - 01:03:08 | 200 | 66.0629ms | ::1 | DELETE "/api/quantum_tasks/28"
```

## GET списка услуг

The screenshot shows the Postman interface with a successful GET request to `http://localhost:8080/api/gates`. The response status is 200 OK, time 6 ms, and size 3.26 KB. The response body is a JSON object:

```
[{"data": {"items": [{"ID_gate": 5, "Title": "Z-axis Rotation Gate", "Description": "Вращает кубик вокруг оси Z на угол тела.", "Status": true, "Image": "http://127.0.0.1:9000/ibm-pictures/img/z-rot-gate.png", "T0j0": null, "T0j1": null, "T1j0": null, "T1j1": null, "Matrix_koeff": null}]}]
```

The terminal tab shows the command history:

```
[GIN] 2025/10/13 - 00:49:27 | 200 | 5.4355ms | ::1 | GET "/api/quantum_task/current"
[GIN] 2025/10/13 - 01:03:08 | 200 | 66.0629ms | ::1 | DELETE "/api/quantum_tasks/28"
[GIN] 2025/10/13 - 01:03:13 | 200 | 1.7718ms | ::1 | GET "/api/gates"
```

## добавить услугу в заявку

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/draft/gates/2`. The response status is 201 Created, time 79 ms, size 180 B. The response body is:

```

1 [
2   "data": {
3     "service_id": 2,
4     "task_id": 29
5   },
6   "status": "ok"
7 ]

```

## добавить другую услугу в заявку

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/draft/gates/1`. The response status is 201 Created, time 18 ms, size 180 B. The response body is:

```

1 [
2   "data": {
3     "service_id": 1,
4     "task_id": 29
5   },
6   "status": "ok"
7 ]

```

## GET иконки корзины

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/quantum_task/current`. The response status is 200 OK, time 17 ms, size 179 B. The response body is:

```

1 [
2   "data": {
3     "services_count": 2,
4     "task_id": 29
5   },
6   "status": "ok"
7 ]

```

[посмотреть заявку](#)

The screenshot shows the Postman interface with a successful API call to `/api/quantum_tasks/29`. The response body is a JSON object:

```
2 "data": { 3 "ID_task": 29, 4 "TaskStatus": "завершён", 5 "CreationDate": "2025-10-13T01:07:23.646753+03:00", 6 "ID_user": 1, 7 "ConclusionDate": "2025-10-13T01:17:26.330398+03:00", 8 "ID_moderator": 1, 9 "Formedate": "2025-10-13T01:16:59.957096+03:00", 10 "TaskDescription": "описание задачи для скрина работы ЛР3", 11 "Res_coeff_0": 1, 12 "Res_coeff_1": 0, 13 "GatesDegrees": [ 14 { 15 "ID_gate": 1, 16 "ID_task": 29, 17 "Degrees": null, 18 "Gate": { } 19 } 20 ] 21 }
```

изменить поле M-M

The screenshot shows the Postman application interface. The left sidebar lists various API endpoints under 'My Workspace'. The main area shows a 'PUT' request to `http://localhost:8080/api/tasks/29/services/2`. The 'Params' tab is selected, showing two parameters: `task_id` with value 29 and `service_id` with value 2. The 'Body' tab displays a JSON response:

```
1 [ { "data": { "degrees": 20, "service_id": 2, "task_id": 29 }, "status": "ok" } ]
```

The status bar at the bottom indicates a successful response with `Status: 200 OK Time: 68 ms Size: 188 B`.

[изменить заявку](#)

The screenshot shows the Postman interface with the following details:

- Header Bar:** POSTMAN, web-service (RIP) / QuantumTask (saneika) / Task\_PUT\_prop, Save, View Documentation, No Environment.
- Left Sidebar:** My Workspace, New HTTP Request, Service\_DELETE, Service\_PUT, Users, GET Users\_GET, PUT Users\_PUT, POST Users\_POST\_reg, POST Users\_POST\_auth, POST Users\_POST\_death, m2m, m2m\_DELETE, m2m\_PUT, QuantumTask (список), Task\_DELETE, GET Task\_GET\_userDraft, GET Task\_GET\_list, GET Task\_GET\_one, PUT Task\_PUT\_prop.
- Request Section:** Method: PUT, URL: http://localhost:8080/api/quantum.tasks/29, Headers: Content-Type: application/json, Body: { "ID": 29, "TaskStatus": "Черновик", "CreationDate": "2025-10-13T01:07:23.646753+03:00", "ID\_user": 1, "ConclusionDate": "0001-01-01T03:00:00+03:00", "ID\_moderator": null, "FormedDate": "0001-01-01T03:00:00+03:00", "TaskDescription": "Описание задачи для скрипта работы ЛР" }.
- Response Section:** Status: 200 OK, Time: 21 ms, Size: 631 B.

завершить введенную заявку (показать ошибку)

The screenshot shows the Postman interface with a failed PUT request to `http://localhost:8080/api/quantum_tasks/29/resolve`. The response status is 400 Bad Request, with the message "only formed can be resolved", "status": "fail".

сформировать заявку

The screenshot shows the Postman interface with a successful PUT request to `http://localhost:8080/api/quantum_tasks/29/form`. The response status is 200 OK, with a JSON payload containing task details.

завершить сформированную заявку

The screenshot shows the Postman interface with a successful PUT request to `http://localhost:8080/api/quantum_tasks/29/resolve`. The response status is 200 OK, with a JSON payload indicating the task is now "заявлен", "status": "resolved".

зарегистрировать нового пользователя

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/api/users/register`. The request body is set to `JSON` and contains the following JSON:

```
1 [ {  
2   "login": "screen_testing_login",  
3   "password": "test_password"  
4 }]
```

The response status is `201 Created`, time `21 ms`, size `183 B`. The response body is:

```
1 [ {  
2   "data": {  
3     "login": "screen_testing_login"  
4   },  
5   "status": "ok"  
6 }]
```

Request details: [GIN] 2025/10/13 - 01:18:45 | 201 | 15.9405ms | :1 | POST | "/api/users/register"

Показать измененные данные через select

The screenshot shows a database interface with two tables:

**users**

| <b>id_user</b> | <b>login</b>         | <b>password</b> | <b>is_admin</b> |
|----------------|----------------------|-----------------|-----------------|
| 2              | admin                | adminpassword   | 1               |
| 1              | user1                | password        |                 |
| 3              | test_login           | test_password   |                 |
| 4              | screen_testing_login | test_password   |                 |

4 строки (0.001 s) Редактировать, Explain, Экспорт

**degrees\_to\_gates**

| <b>id_gate</b> | <b>id_task</b> | <b>degrees</b> |
|----------------|----------------|----------------|
| 2              | 29             | 20             |
| 1              | 29             | NULL           |

29 завершён 2025-10-12 22:07:23.646753+00 1 2025-10-12 22:17:26.330398+00 1 Описание задачи для скрипта работы ЛР3 1 0 2025-10-12 22:16:59.957096+00  
29 строк (0.002 s) Редактировать, Explain, Экспорт

SELECT \* FROM quantum\_tasks ORDER BY id\_task