

1. Тестирование

1.1. Модульное тестирование

Производится модульное тестирование модуля сервера данных, отвечающего за работу комплексного запроса `get_free_subtask_by_agent_id`.

Тестирование производится при помощи библиотеки `unittest` в автоматическом режиме. Тест самостоятельно генерирует входные данные. Тест не генерирует дополнительного вывода в случае корректной работы, кроме унифицированного библиотекой `unittest`.

Все тесты были пройдены успешно.

Типовые тесты:

Тестируемый метод	Входные данные	Выходные данные
<code>post_item</code>	<code>'trait', {"name":"trait1", "version":"1.0"}</code>	200 OK
<code>post_item</code>	<code>'agent', {}</code>	200 OK
<code>post_item</code>	<code>'agent', {}</code>	200 OK
<code>post_item</code>	<code>'task', {"max_time":160, "archive_name":"cocoque"}</code>	200 OK
<code>post_item</code>	<code>'subtask', {"task_id":1, "status":"queued", "archive_name":"cocoque0"}</code>	200 OK
<code>post_item</code>	<code>'mtm_traitagent', {"trait_id":1,"agent_id":1}</code>	200 OK
<code>post_item</code>	<code>'mtm_traittask', {"trait_id":1,"task_id":1}</code>	200 OK
<code>get_free_subtask_by_agent_id</code>	1	200 OK
<code>get_free_subtask_by_agent_id</code>	2	404 NOT FOUND
<code>get_free_subtask_by_agent_id</code>	<code>'text'</code>	400 BAD REQUEST

Код теста

```
1 #!/bin/env python
2 import sys
3 import os
4
5 PACKAGE_PARENT = '..'
```

```

6 SCRIPT_DIR = os.path.dirname(os.path.realpath(os.path.join(os.getcwd(), os.path.
    expanduser(__file__))))
7 sys.path.append(os.path.normpath(os.path.join (SCRIPT_DIR, PACKAGE_PARENT)))
8
9 import unittest
10 from data_backend import post_item, get_free_subtask_by_agent_id, init_db
11
12 class utDataBackend(unittest.TestCase):
13
14     def test_get_free_subtask_by_agent_id(self):
15         def ut_assert( self , qry, code = 200):
16             self.assertTrue(qry == code)
17         init_db()
18
19         ut_assert( self , post_item('trait', {"name":"trait1","version":"1.0"}))
20         ut_assert( self , post_item('trait', {"name":"trait2","version":"2.0"}))
21         ut_assert( self , post_item('trait', {"name":"trait3","version":"3.0"}))
22         ut_assert( self , post_item('trait', {"name":"trait4","version":"4.0"}))
23
24         ut_assert( self , post_item('agent', {}))
25         ut_assert( self , post_item('agent', {}))
26
27         ut_assert( self , post_item('task', {"max_time":160,"archive_name":"cocoque"
28             }))
29         ut_assert( self , post_item('task', {"max_time":260,"archive_name":"
30             kukareque"}))
31
32         ut_assert( self , post_item('subtask', {"task_id":1,"status":"queued","
33             archive_name":"cocoque0"}))
34         ut_assert( self , post_item('subtask', {"task_id":2,"status":"queued","
35             archive_name":"cocoque1"}))
36         ut_assert( self , post_item('subtask', {"task_id":1,"status":"queued","
37             archive_name":"cocoque2"}))
38         ut_assert( self , post_item('subtask', {"task_id":2,"status":"queued","
39             archive_name":"cocoque3"}))
40         ut_assert( self , post_item('subtask', {"task_id":1,"status":"queued","
41             archive_name":"cocoque4"}))
42
43         ut_assert( self , post_item('mtm_traitagent', {"trait_id":1,"agent_id":2}))
44         ut_assert( self , post_item('mtm_traitagent', {"trait_id":2,"agent_id":3}))
45         ut_assert( self , post_item('mtm_traitagent', {"trait_id":3,"agent_id":4}))
46         ut_assert( self , post_item('mtm_traitagent', {"trait_id":4,"agent_id":1}))
47         ut_assert( self , post_item('mtm_traitagent', {"trait_id":2,"agent_id":2}))

```

```

41     ut_assert( self , post_item('mtm_traitagent', {"trait_id":4,"agent_id":4}))
42
43     ut_assert( self , post_item('mtm_traittask', {"trait_id":1,"task_id":1}))
44     ut_assert( self , post_item('mtm_traittask', {"trait_id":2,"task_id":1}))
45     ut_assert( self , post_item('mtm_traittask', {"trait_id":3,"task_id":1}))
46     ut_assert( self , post_item('mtm_traittask', {"trait_id":2,"task_id":2}))
47     ut_assert( self , post_item('mtm_traittask', {"trait_id":4,"task_id":2}))
48     ut_assert( self , post_item('mtm_traittask', {"trait_id":1,"task_id":2}))
49
50     ut_assert( self , get_free_subtask_by_agent_id(1))
51     ut_assert( self , get_free_subtask_by_agent_id(1))
52     ut_assert( self , get_free_subtask_by_agent_id(1), 404)
53     ut_assert( self , get_free_subtask_by_agent_id(2))
54     ut_assert( self , get_free_subtask_by_agent_id(2))
55     ut_assert( self , get_free_subtask_by_agent_id(2))
56     ut_assert( self , get_free_subtask_by_agent_id(2), 404)
57
58     ut_assert( self , get_free_subtask_by_agent_id(10), 404)
59
60     ut_assert( self , get_free_subtask_by_agent_id('lkkl'), 400)
61
62 if __name__ == '__main__':
63     unittest.main()

```

Унифицированный вывод библиотеки unittest

```

1 -----
2 Ran 1 test in 1.231s
3
4 ОК

```

1.2. Системное тестирование

Производится системное тестирования файлового сервера (методом чёрного ящика). Тест покрывает 100% АПИ файлового сервера.

Входными данными всех тестов являются адрес и порт сервера. При проведении тестирования сервер был запущен по адресу **localhost:50002**

Все тесты были пройдены успешно.

Список тестов:

- Сохранение файла в корневой директории сервера

Ресурсы: /static

HTTP-метод: POST

Параметры запроса: file: a.out, headers: “Content-type”=“multipart/form-data”

Код, содержимое ответа: 200, пустой JSON-объект “{”

- Сохранение файла в произвольной директории сервера

Ресурс: /static

HTTP-метод: POST

Параметры запроса: query: “path=1\2\3\4”, file: a.out, headers: “Content-type”=“multipart/form-data”

Код, содержимое ответа: 200, пустой JSON-объект “{”

- Доступ к существующему файлу в корневой директории сервера

Ресурс: /static/a.out

HTTP-метод: GET

Код, содержимое ответа: 200, содержимое файла a.out

Замечание: содержимое ответа перенаправлено в файл a_.out

- Доступ к существующему файлу в произвольной директории сервера

Ресурс: /static/1\2\3\4\a.out

HTTP-метод: GET

Код, содержимое ответа: 200, содержимое файла a.out

- Удаление существующего файла

Ресурс: /static/a.out

HTTP-метод: DELETE

Код, содержимое ответа: 200, пустой JSON-объект “{”

- Удаление существующего файла в произвольной директории сервера

Ресурс: /static/1\2\3\4\a.out

HTTP-метод: DELETE

Код, содержимое ответа: 200, пустой JSON-объект “{”

- Удаление несуществующего файла

Ресурс: /static/a.out

HTTP-метод: DELETE

Код, содержимое ответа: 404, JSON-объект “{”error“:”Not Found“}”

Замечание: файл “a.out” удалён с сервера в ходе предыдущих тестов

- Удаление несуществующего файла в произвольной директории сервера

Ресурс: /static/1\2\3\4\a.out

HTTP-метод: DELETE

Код, содержимое ответа: 404, JSON-объект “{”error“:”Not Found“}”

- Доступ к несуществующему файлу в корневой директории сервера

Ресурс: /static/a.out

HTTP-метод: GET

Код, содержимое ответа: 404, JSON-объект “{”error“:”Not Found“}”

- Доступ к несуществующему файлу в произвольной директории сервера

Ресурс: /static/1\2\3\4\a.out

HTTP-метод: GET

Код, содержимое ответа: 404, JSON-объект “{”error“:”Not Found“}”

Код теста

```
1 #!/bin/bash
2
3 echo "#include <iostream>" > test.cpp
4 echo "int main(){std::cout << \"HELLO WORLD!\" << std::endl;}" >> test.cpp
5 g++ test.cpp -o a.out
6
7 set -v
8
9 curl @localhost:50002/static" -X POST -F file=@a.out; echo
10 curl @localhost:50002/static?path=1\2\3\4" -X POST -F file=@a.out; echo
11
12 curl @localhost:50002/static/a.out" -X GET > a__.out ; echo
13 chmod +x a__.out ; echo
14 ./a__.out ; echo
15
16 curl @localhost:50002/static/1\2\3\4\a.out" -X GET > a_.out ; echo
```

```

17 chmod +x a_.out ; echo
18 ./a_.out ; echo
19
20 curl @localhost:50002/static/a.out" -X DELETE ; echo
21 curl @localhost:50002/static/1\2\3\4\a.out" -X DELETE ; echo
22
23 curl @localhost:50002/static/a.out" -X DELETE ; echo
24 curl @localhost:50002/static/1\2\3\4\a.out" -X DELETE ; echo
25
26 curl @localhost:50002/static/a.out" -X GET ; echo
27 curl @localhost:50002/static/1\2\3\4\a.out" -X GET ; echo
28
29 rm a*.out test.cpp

```

Выход терминала в ходе тестов

```

1 curl @localhost:50002/static" -X POST -F file=@a.out; echo
2 {}
3 curl @localhost:50002/static?path=1\2\3\4" -X POST -F file=@a.out; echo
4 {}
5
6 curl @localhost:50002/static/a.out" -X GET > a__.out ; echo
7 % Total % Received % Xferd Average Speed Time Time Time Current
8 Dload Upload Total Spent Left Speed
9 100 8384 100 8384 0 0 4047k 0 --:--:-- --:--:-- --:--:-- 8187
k
10 chmod +x a__.out ; echo
11 ./a__.out ; echo
12 HELLO WORLD!
13
14 curl @localhost:50002/static/1\2\3\4\a.out" -X GET > a_.out ; echo
15 % Total % Received % Xferd Average Speed Time Time Time Current
16 Dload Upload Total Spent Left Speed
17 100 8384 100 8384 0 0 4885k 0 --:--:-- --:--:-- --:--:-- 8187
k
18 chmod +x a_.out ; echo
19 ./a_.out ; echo
20 HELLO WORLD!
21
22 curl @localhost:50002/static/a.out" -X DELETE ; echo
23 {}
24 curl @localhost:50002/static/1\2\3\4\a.out" -X DELETE ; echo
25 {}
26

```

```

27 curl @localhost:50002/static/a.out"          -X DELETE          ; echo
28 {
29     "error": "Not found"
30 }
31 curl @localhost:50002/static/1\2\3\4\a.out" -X DELETE          ; echo
32 {
33     "error": "Not found"
34 }
35
36 curl @localhost:50002/static/a.out"          -X GET            ; echo
37 {
38     "error": "Not found"
39 }
40 curl @localhost:50002/static/1\2\3\4\a.out" -X GET            ; echo
41 {
42     "error": "Not found"
43 }
44
45 rm a*.out test.cpp

```

1.3. Интеграционное тестирование

Производится интеграционное тестирование подсистем работы с файлами, данными и подсистемы мониторинга.

Тест покрывает прецедент регистрации подсистем работы с файлами и данными в подсистеме мониторинга. Тест не требует входных данных.

Код теста

```

1 #!/bin/bash
2 python ../beacon_backend/beacon_backend.py 50003 &
3 python ../data_backend/data_backend.py localhost:50003 10.0.0.10:5432 50001 &
4 python ../file_backend/file_backend.py localhost:50003 50002 &
5
6 echo "sleep 30"
7 sleep 50
8
9 killall -9 python

```

Выход терминала в ходе тестов

```

1 * Running on http://0.0.0.0:50003/ (Press CTRL+C to quit)
2 * Restarting with stat
3 Starting with settings : beacon:localhost:50003 self : 0.0.0.0:50002

```

```

4 Starting with settings : Beacon: localhost:50003 DB: 10.0.0.10:5432, self :
   0.0.0.0:50001
5 Beacon is down. Waiting to reconnect.
6 Incoming request from 127.0.0.1: port = 50003, state = Unable to find beacon
7 127.0.0.1 -- [28/May/2015 02:23:31] "POST /services/fileserver HTTP/1.1" 200 -
8 Beacon is back up.
9 * Running on http://0.0.0.0:50002/ (Press CTRL+C to quit)
10 * Restarting with stat
11 Incoming request from 127.0.0.1: port = 50003, state = Operating normally
12 127.0.0.1 -- [28/May/2015 02:23:31] "POST /services/database HTTP/1.1" 200 -
13 * Running on http://0.0.0.0:50001/ (Press CTRL+C to quit)
14 * Restarting with stat
15 Starting with settings : beacon:localhost:50003 self : 0.0.0.0:50002
16 Incoming request from 127.0.0.1: port = 50003, state = Operating normally
17 127.0.0.1 -- [28/May/2015 02:23:32] "POST /services/fileserver HTTP/1.1" 200 -
18 Starting with settings : Beacon: localhost:50003 DB: 10.0.0.10:5432, self :
   0.0.0.0:50001
19 Incoming request from 127.0.0.1: port = 50003, state = Operating normally
20 127.0.0.1 -- [28/May/2015 02:23:32] "POST /services/database HTTP/1.1" 200 -
21 Incoming request from 127.0.0.1:50003: state = Operating normally
22 127.0.0.1 -- [28/May/2015 02:23:41] "PUT /services/fileserver/127.0.0.1:50003
   HTTP/1.1" 200 -
23 Incoming request from 127.0.0.1:50003: state = Operating normally
24 127.0.0.1 -- [28/May/2015 02:23:41] "PUT /services/database/127.0.0.1:50003
   HTTP/1.1" 200 -
25 Incoming request from 127.0.0.1:50003: state = Operating normally
26 127.0.0.1 -- [28/May/2015 02:23:42] "PUT /services/fileserver/127.0.0.1:50003
   HTTP/1.1" 200 -
27 Incoming request from 127.0.0.1:50003: state = Operating normally
28 127.0.0.1 -- [28/May/2015 02:23:42] "PUT /services/database/127.0.0.1:50003
   HTTP/1.1" 200 -
29 Incoming request from 127.0.0.1:50003: state = Operating normally
30 127.0.0.1 -- [28/May/2015 02:23:51] "PUT /services/fileserver/127.0.0.1:50003
   HTTP/1.1" 200 -
31 Incoming request from 127.0.0.1:50003: state = Operating normally
32 127.0.0.1 -- [28/May/2015 02:23:51] "PUT /services/database/127.0.0.1:50003
   HTTP/1.1" 200 -
33 Incoming request from 127.0.0.1:50003: state = Operating normally
34 127.0.0.1 -- [28/May/2015 02:23:52] "PUT /services/fileserver/127.0.0.1:50003
   HTTP/1.1" 200 -
35 Incoming request from 127.0.0.1:50003: state = Operating normally
36 127.0.0.1 -- [28/May/2015 02:23:52] "PUT /services/database/127.0.0.1:50003
   HTTP/1.1" 200 -

```


37 Incoming request from 127.0.0.1:50003: state = Operating normally
38 127.0.0.1 -- [28/May/2015 02:24:01] "PUT /services/fileserver/127.0.0.1:50003
HTTP/1.1" 200 -
39 Incoming request from 127.0.0.1:50003: state = Operating normally
40 127.0.0.1 -- [28/May/2015 02:24:01] "PUT /services/database/127.0.0.1:50003
HTTP/1.1" 200 -
41 Incoming request from 127.0.0.1:50003: state = Operating normally
42 127.0.0.1 -- [28/May/2015 02:24:02] "PUT /services/fileserver/127.0.0.1:50003
HTTP/1.1" 200 -
43 Incoming request from 127.0.0.1:50003: state = Operating normally
44 127.0.0.1 -- [28/May/2015 02:24:02] "PUT /services/database/127.0.0.1:50003
HTTP/1.1" 200 -
45 Incoming request from 127.0.0.1:50003: state = Operating normally
46 127.0.0.1 -- [28/May/2015 02:24:11] "PUT /services/fileserver/127.0.0.1:50003
HTTP/1.1" 200 -
47 Incoming request from 127.0.0.1:50003: state = Operating normally
48 127.0.0.1 -- [28/May/2015 02:24:11] "PUT /services/database/127.0.0.1:50003
HTTP/1.1" 200 -
49 Incoming request from 127.0.0.1:50003: state = Operating normally
50 127.0.0.1 -- [28/May/2015 02:24:12] "PUT /services/fileserver/127.0.0.1:50003
HTTP/1.1" 200 -
51 Incoming request from 127.0.0.1:50003: state = Operating normally
52 127.0.0.1 -- [28/May/2015 02:24:12] "PUT /services/database/127.0.0.1:50003
HTTP/1.1" 200 -