

# 1. Тестирование

## 1.1. Модульное тестирование

Статический класс DataMethods:

Представляет собой интерфейс бекенда данных.

Производится тестирование класса DataMethods с использованием методики тестирования – разбиение на уровне класса на категории по функциональности. Категория объединяет в себе методы класса, выполняющие близкую по смыслу функциональность.

Методы класса можно разбить на 3 категории по функциональности:

- методы получения данных;
- методы изменения данных;
- методы удаления данных.

Таблица 1: Методы класса DataMethodsFilter

Название метода	Примечания
get_item	param: table [ str ] param: value_json[ dict ] Возвращает все элементы таблицы с именем table, удовлетворяющие фильтру value_json
put_item	param: table [ str ] param: value_json [ dict ] Изменяет все элементы таблицы с именем table, удовлетворяющие фильтру value_json
delete_item	param: table [ str ] param: value_json [ dict ] Удаляет все элементы таблицы с именем table, удовлетворяющие фильтру value_json

Таблица 2: Категория 1 – Тестирование метода получения данных

Название теста	TestGetItemEmpty
Тестируемый метод	get_item
Описание теста	Проверка получения данных из БД, пустой таблицы trait
Ожидаемый результат	Словарь с пустым списком в ключе “result”
Степень важности	Фатальная
Результат теста	Тест пройден

Таблица 3: Категория 1 – Тестирование метода получения данных

Название теста	TestGetItemGeneral
Тестируемый метод	get_item
Описание теста	Проверка изменения данных в БД, в таблице trait, предварительно наполненной записями name = “test_name”, version = “1.0” name = “test_name”, version = “2.0”, по фильтру {“name”:“test_name”}
Ожидаемый результат	Словарь {“result”:[{“name”:“test_name”, “version”:“1.0”}, {“name”:“test_name”, “version”:“2.0”}]}
Степень важности	Фатальная
Результат теста	Тест пройден

Таблица 4: Категория 2 – Тестирование метода изменения данных

Название теста	TestPutItem
Тестируемый метод	put_item
Описание теста	Проверка изменения данных в БД, в таблице trait, предварительно наполненной записями name = "test_name", version = "1.0" name = "test_name", version = "2.0", по фильтру {"name":"test_name","changes": {"version":"3.0"}}
Ожидаемый результат	Словарь {"count": 2}, отражающий наличие двух произведенных изменений в БД
Степень важности	Фатальная
Результат теста	Тест пройден

Таблица 5: Категория 3 – Тестирование метода удаления данных

Название теста	TestDeleteItem
Тестируемый метод	delete_item
Описание теста	Проверка удаления данных в БД, в таблице trait, предварительно наполненной записями name = "test_name", version = "1.0" name = "test_name", version = "2.0", по фильтру {"name":"test_name"}
Ожидаемый результат	Словарь {"count": 2}, отражающий наличие двух произведенных удалений в БД
Степень важности	Фатальная
Результат теста	Тест пройден

## Вывод по результатам тестирования

Все тесты пройдены успешно, класс готов к использованию.

## 1.2. Системное тестирование

Производится системное тестирования файлового сервера (стратегия чёрного ящика). Тест покрывает все прецеденты взаимодействия с файловым сервером.

При формировании тестовых наборов использовалась методика эквивалентного разбиения для входных данных.

Классы эквивалентности для входных данных:

Параметр	Допустимые классы эквивалентности	Недопустимые классы эквивалентности
Имена файлов	Строки, не содержащие символы &,\$,/,,:*,? и другие спецсимволы	Строки, содержащие запрещённые символы
Адрес сервера	Строки вида 'http://address:port'	Строки, не подходящие под описание
Удалённый ресурс	Такие же строки, как и допустимые для имён файлов	Такие же строки, как и недопустимые для имён файлов
Запрос-файл	Запрос с содержимым файла в теле и с хедером 'Content-type':'multipart/form-data'	Запрос без необходимого заголовка либо испорченным телом

Тесты:

№	Описание теста	Входные данные	Ожидаемый результат	Полученный результат
1	Проверка возможности сохранения файла	Файл-запрос	Сообщение об успешном сохранении файла	Сообщение протокола HTTP 200 OK, пустой JSON-объект
2	Проверка доступа к существующему файлу	Запрос по адресу файла	Содержимое файла	Сообщение протокола HTTP 200 OK, содержимое искомого файла
3	Проверка удаления существующего файла	Запрос на удаление файла	Сообщение об успешном удалении файла	Сообщение протокола HTTP 200 OK, пустой JSON-объект
4	Проверка удаления несуществующего файла	Запрос на удаление файла по несуществующему пути	Сообщение об ошибке	Сообщение протокола HTTP 404 NOT FOUND, JSON-объект с сообщением об ошибке
5	Проверка доступа к несуществующему файлу	Запрос по адресу несуществующего файла	Сообщение об ошибке	Сообщение протокола HTTP 404 NOT FOUND, JSON-объект с сообщением об ошибке

Подробное описание тестов:

- Сохранение файла в корневой директории сервера

**Ресурс:** /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"}

## Тестирование надёжности и доступности серверов

№	Описание теста	Входные данные	Ожидаемый результат	Полученный результат
1	Провести тест, в котором несколько пользователей одновременно загружают файл на сервер	Пользовательские файлы, их имена	Все файлы загружены по верным адресам	Каждый из пользователей успешно загрузил файл на сервер
2	Провести тест, в котором происходит разрыв соединения в ходе загрузки файла	Пользовательский файл	Загрузка возобновляется по восстановлении соединения	Возникла ошибка. Необходима повторная загрузка файла.

Отчёт об обнаруженных ошибках:

Сервис: файловый сервер	
Степень важности: средняя	
Надёжность	Сервис некорректно обрабатывает сценарий разрыва соединения

## Выводы по результатам системного тестирования

Сервис может быть использован только в средах, где сеть можно считать достаточно надёжной. Для использования в рамках сетей где часты разрывы необходимо производить доработку сервиса.

## 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/filesserver 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/filesserver 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/filesserver/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/filesserver/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/filesserver/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 -