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 (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

  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“}”

Код теста

```
#!/bin/bash

echo "#include <iostream>" > test.cpp
echo "int main(){std::cout << \"HELLO WORLD!\" << std::endl;}" >> test.cpp
g++ test.cpp -o a.out

set -v

curl @"localhost:50002/static"                  -X POST -F file=@a.out; echo
curl @"localhost:50002/static?path=1\2\3\4" -X POST -F file=@a.out; echo

curl @"localhost:50002/static/a.out"            -X GET > a__.out   ; echo
chmod +x a__.out                                                   ; echo
./a__.out                                                          ; echo

curl @"localhost:50002/static/1\2\3\4\a.out" -X GET > a_.out       ; echo
chmod +x a_.out                                                    ; echo
./a_.out                                                           ; echo

curl @"localhost:50002/static/a.out"            -X DELETE          ; echo
curl @"localhost:50002/static/1\2\3\4\a.out" -X DELETE             ; echo

curl @"localhost:50002/static/a.out"            -X DELETE          ; echo
curl @"localhost:50002/static/1\2\3\4\a.out" -X DELETE             ; echo

curl @"localhost:50002/static/a.out"            -X GET             ; echo
curl @"localhost:50002/static/1\2\3\4\a.out" -X GET                ; echo

rm a*.out test.cpp
```

Выход терминала в ходе тестов

```
curl @"localhost:50002/static"                  -X POST -F file=@a.out; echo
{}
curl @"localhost:50002/static?path=1\2\3\4" -X POST -F file=@a.out; echo
{}

curl @"localhost:50002/static/a.out"            -X GET > a__.out   ; echo
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8384  100  8384    0     0  4047k      0 --:--:-- --:--:-- --:--:-- 8187
    k
```

```
chmod +x a__.out                                                    ; echo
./a__.out                                                           ; echo
HELLO WORLD!

curl @"localhost:50002/static/1\2\3\4\a.out" -X GET > a_.out        ; echo
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8384  100  8384    0     0  4885k      0 --:--:-- --:--:-- --:--:-- 8187
    k
chmod +x a_.out                                                     ; echo
./a_.out                                                            ; echo
HELLO WORLD!

curl @"localhost:50002/static/a.out"          -X DELETE             ; echo
{}
curl @"localhost:50002/static/1\2\3\4\a.out" -X DELETE              ; echo
{}

curl @"localhost:50002/static/a.out"          -X DELETE             ; echo
{
  "error": "Not found"
}
curl @"localhost:50002/static/1\2\3\4\a.out" -X DELETE              ; echo
{
  "error": "Not found"
}

curl @"localhost:50002/static/a.out"          -X GET                ; echo
{
  "error": "Not found"
}
curl @"localhost:50002/static/1\2\3\4\a.out" -X GET                 ; echo
{
  "error": "Not found"
}

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

```
#!/bin/bash
python ../beacon_backend/beacon_backend.py 50003 &
python ../data_backend/data_backend.py localhost:50003 10.0.0.10:5432 50001 &
python ../file_backend/file_backend.py localhost:50003 50002 &

echo "sleep 30"
sleep 50

killall -9 python
```

Выход терминала в ходе тестов

```
 * Running on http://0.0.0.0:50003/ (Press CTRL+C to quit)
 * Restarting with stat
Starting with settings: beacon: localhost:50003 self: 0.0.0.0:50002
Starting with settings: Beacon: localhost:50003 DB: 10.0.0.10:5432, self:
    0.0.0.0:50001
Beacon is down. Waiting to reconnect.
Incoming request from 127.0.0.1: port = 50003, state = Unable to find beacon
127.0.0.1 - - [28/May/2015 02:23:31] "POST /services/fileserver HTTP/1.1" 200 -
Beacon is back up.
 * Running on http://0.0.0.0:50002/ (Press CTRL+C to quit)
 * Restarting with stat
Incoming request from 127.0.0.1: port = 50003, state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:31] "POST /services/database HTTP/1.1" 200 -
 * Running on http://0.0.0.0:50001/ (Press CTRL+C to quit)
 * Restarting with stat
Starting with settings: beacon: localhost:50003 self: 0.0.0.0:50002
Incoming request from 127.0.0.1: port = 50003, state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:32] "POST /services/fileserver HTTP/1.1" 200 -
Starting with settings: Beacon: localhost:50003 DB: 10.0.0.10:5432, self:
    0.0.0.0:50001
Incoming request from 127.0.0.1: port = 50003, state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:32] "POST /services/database HTTP/1.1" 200 -
Incoming request from 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:41] "PUT /services/fileserver/127.0.0.1:50003
    HTTP/1.1" 200 -
Incoming request from 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:41] "PUT /services/database/127.0.0.1:50003
    HTTP/1.1" 200 -
Incoming request from 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:42] "PUT /services/fileserver/127.0.0.1:50003
```

HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:42] "PUT /services/database/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:51] "PUT /services/fileserver/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:51] "PUT /services/database/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:52] "PUT /services/fileserver/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:23:52] "PUT /services/database/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:01] "PUT /services/fileserver/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:01] "PUT /services/database/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:02] "PUT /services/fileserver/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:02] "PUT /services/database/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:11] "PUT /services/fileserver/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:11] "PUT /services/database/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:12] "PUT /services/fileserver/127.0.0.1:50003 HTTP/1.1" 200 -
Incoming request **from** 127.0.0.1:50003: state = Operating normally
127.0.0.1 - - [28/May/2015 02:24:12] "PUT /services/database/127.0.0.1:50003 HTTP/1.1" 200 -