|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***Доработка функционала выгрузки данных результатов работы распределенной вычислительной системы на сторону web-клиента***

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы (проекта) **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2019 г.*

**АННОТАЦИЯ**

Работа посвящена доработке функционала выгрузки данных результатов работы распределенной вычислительной системы на сторону web-клиента. В результате работы был изучен механизм передачи данных со стороны распределенной вычислительной системы на сторону web-клиента. Для реализации необходимой доработки функционала были проведены доработки существующего функционала на всех трёх этапах передачи информации.

Тип работы: курсовая работа.

Тема работы: Доработка функционала выгрузки данных результатов работы распределенной вычислительной системы на сторону web-клиента.

Объект исследований: Web-клиент распределенной вычислительной системы.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc7003507)

[1. ПОСТАНОВКА ЗАДАЧИ 5](#_Toc7003508)

[2. АРХИТЕКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ 7](#_Toc7003509)

[3. ТЕСТИРОВАНИЕ И ОТЛАДКА 10](#_Toc7003510)

[ЗАКЛЮЧЕНИЕ 12](#_Toc7003511)

# ВВЕДЕНИЕ

Web-клиент распределенной вычислительной системы позволяет частично задействовать функционал системы, располагающейся на сервере, удалено, предоставляя возможность выполнять такие действия как: построение графоориентированных вычислительных алгоритмов, которые в дальнейшем выполняются на сервере, а также на стороне клиента реализован функционал по выгрузке различных данных об экспериментах в табличном формате. В текущей версии web-клиента выгрузка данных об экспериментах реализована таким образом, что со стороны клиента на сторону сервера посылается запрос сразу на всю информацию, хранящуюся в таблице, затем, после ответа от сервера, на стороне клиента происходит отображение сразу всей информации, пришедшей в ответе от сервера. Данный вариант реализации неудобен при работе с большими таблицами (более 10000 строк), так как операция запроса и отображения данных занимает достаточно большой промежуток времени, что приводит к тому, что пользователю приходится ожидать отображения необходимых данных продолжительный период времени. Данная курсовая работа посвящена тому, чтобы доработать текущую версию web-клиента с целью исключения нежелательно долгого отображения больших массивов данных с информацией о проводимых экспериментах.

1. **ПОСТАНОВКА ЗАДАЧИ**

Для оптимизации выгрузки и отображения больших массивов данных на стороне клиента был предложен следующий вариант работы функционала: изначально, при запросе на таблицу со стороны клиента, происходит запрос на первые 50 записей, хранящихся в таблице, параллельно с этим у клиента устанавливается обработчик события прокручивания таблицы, который позволяет определять момент, когда пользователь просмотрел первые 50 записей и желает продолжить изучение следующих записей из таблицы. В данный момент должен формироваться запрос на следующие 50 записей, хранящихся в таблице, и дальнейшее отображение данных записей на стороне клиента. Схема работы представлена на рисунке 1.

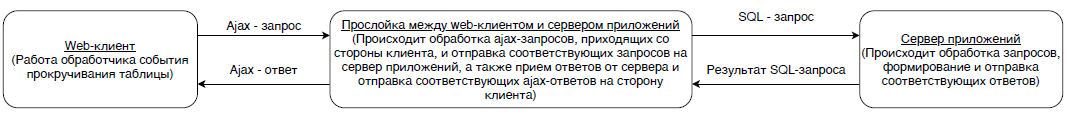
**

Рис. 1 Архитектура работы функционала по запросу массивов данных с информацией о проводимых экспериментах

Реализация данной логики работы функционала по выгрузке данных об экспериментах невозможна без предварительной доработки сервера приложений, так как текущая версия сервера не позволяет обрабатывать   
SQL-запросы на лимитированное количество записей из таблицы. Таким образом, первоначальной задачей стояла задача доработки сервера приложений, с целью реализации функционала по обработке SQL-запросов на лимитированное количество записей из таблицы.

Последняя версия сервера приложений находится на стадии тестирования, в связи с этим была поставлена задача по созданию теста, позволяющего проверить корректность работы функционала сервера приложений по авторизации пользователей. Требовалось написать тест, который позволял бы выполнять тестирование функциональности классов, предоставляющих функционал авторизации на сервере приложений.

В тесте обязательно должны были быть рассмотрены два случая логина/пароля:

1. корректный логин/пароль;
2. некорректный логин/пароль.

Авторизация выполняется с помощью метода authorize(), который является виртуальной функцией базового класса AuthorizationBase. Его выполнение является асинхронным, так что предварительно необходимо задать колбэк с помощью функции setCallbackOnDecisionChanged(). После изменения статуса авторизации, этот колбэк будет вызван вместе со значением AccessState, которое должно быть равно asSuccess в случае корректных логина/пароля и asFailure в случае некорректных логина/пароля.

После проведения авторизации дополнительно необходимо проверить функциональность следующих методов:

* isAuthorized() - должен возвращать true при успешной авторизации;
* who() - должен возвращать "gcdpublic" при авторизации с логином gcdpublic и пустую строку при некорректном логине/пароле;
* getAITypes() - должен возвращать {"COMAPS", "TEST", "META3", "SOCACS", "SOCARI", "SOCLEA", "SOCTCH", "SOCANL"} при авторизации с логином gcdpublic;
* getRoles() - должен возвращать {"server robot", "student", "teacher"} при авторизации с логином gcdpublic;
* getSubsystems() - должен возвращать {"SERVER", "RNDSOC\_TCH", "RNDSOC\_TCH", "RNSSOC\_STU"} при авторизации с логином gcdpublic.

1. **АРХИТЕКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ**

Для реализации тестирования функционала сервера приложений по авторизации с учетом особенностей, указанных в постановке задачи, был реализован алгоритм, блок-схема которого представлена на рисунке 2.

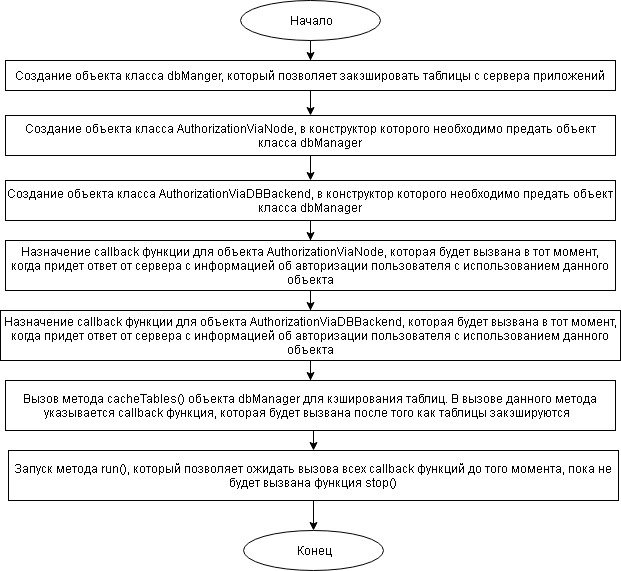


Рис. 2 Блок-схема алгоритма тестирования функционала авторизации

Блок-схемы callback функций, вызов которых происходит когда от сервера приходит ответ на вызов необходимых функций, представлены на рисунках 3-5.

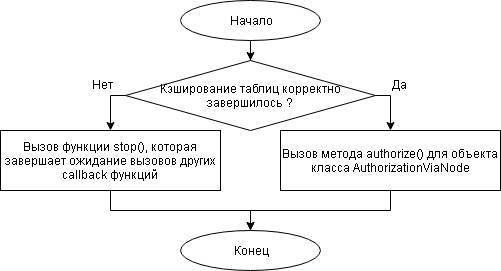


Рис. 3 Блок-схема callback функции, вызов которой происходит после кэширования таблиц

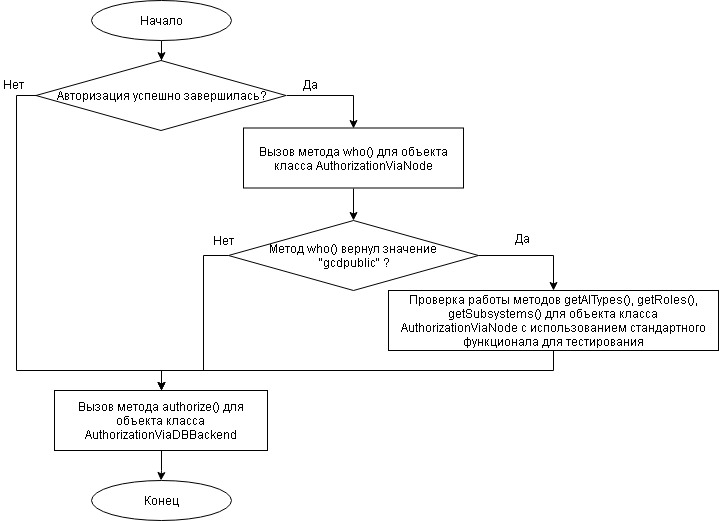


Рис. 4 Блок-схема callback функции, вызов которой происходит после авторизации пользователя с использованием объекта класса AuthorizationViaNode

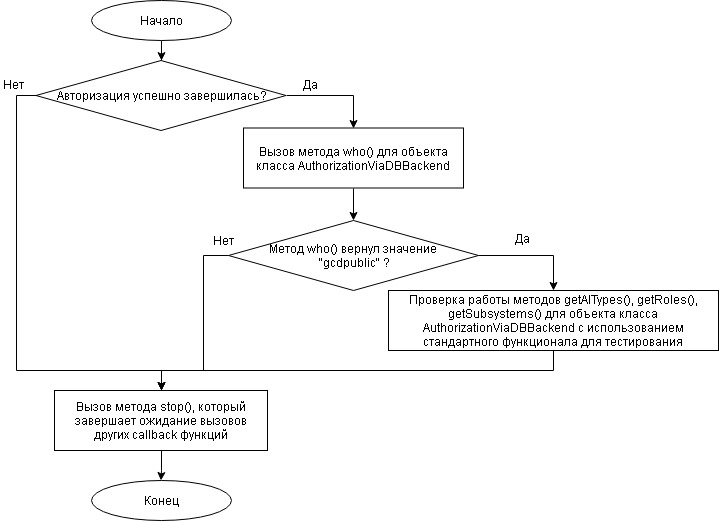


Рис. 5 Блок-схема callback функции, вызов которой происходит после авторизации пользователя с использованием объекта класса AuthorizationViaDBBackend

Особенностью данной программной реализации для тестирования функций авторизации на сервере приложений является применение callback функций, которые позволяют реализовать ожидание ответа от севера без использования нежелательных пассивных циклов. Под пассивными циклами понимаются циклы, внутри которых не происходит обработки информации, их основной задачей является генерация временной задержки необходимой для того, чтобы от сервера пришел ответ на запрос, отправленный со стороны пользователя.

Представленный алгоритм был реализован на языке программирования C++, так как сервер приложений и соответствующий функционал для авторизации на сервере написан на данном языке программирования.

1. **ТЕСТИРОВАНИЕ И ОТЛАДКА**

С целью тестирования и отладки разработанного алгоритма была проверена работа методов isAuthorized() - должен возвращать true при успешной авторизации; who() - должен возвращать "gcdpublic" при авторизации с логином gcdpublic и пустую строку при некорректном логине/пароле; getAITypes() - должен возвращать {"COMAPS", "TEST", "META3", "SOCACS", "SOCARI", "SOCLEA", "SOCTCH", "SOCANL"} при авторизации с логином gcdpublic; getRoles() - должен возвращать {"server robot", "student", "teacher"} при авторизации с логином gcdpublic; getSubsystems() - должен возвращать {"SERVER", "RNDSOC\_TCH", "RNDSOC\_TCH", "RNSSOC\_STU"} при авторизации с логином gcdpublic. Для тестирования необходимо было запустить метод authorize() для объектов класса типа AuthorizationViaNode и AuthorizationViaDBBackend с передачей в данный метод корректных и некорректных логина и пароля. Результат работы разработанного теста для корректных логина и пароля представлены на   
рисунке 6.

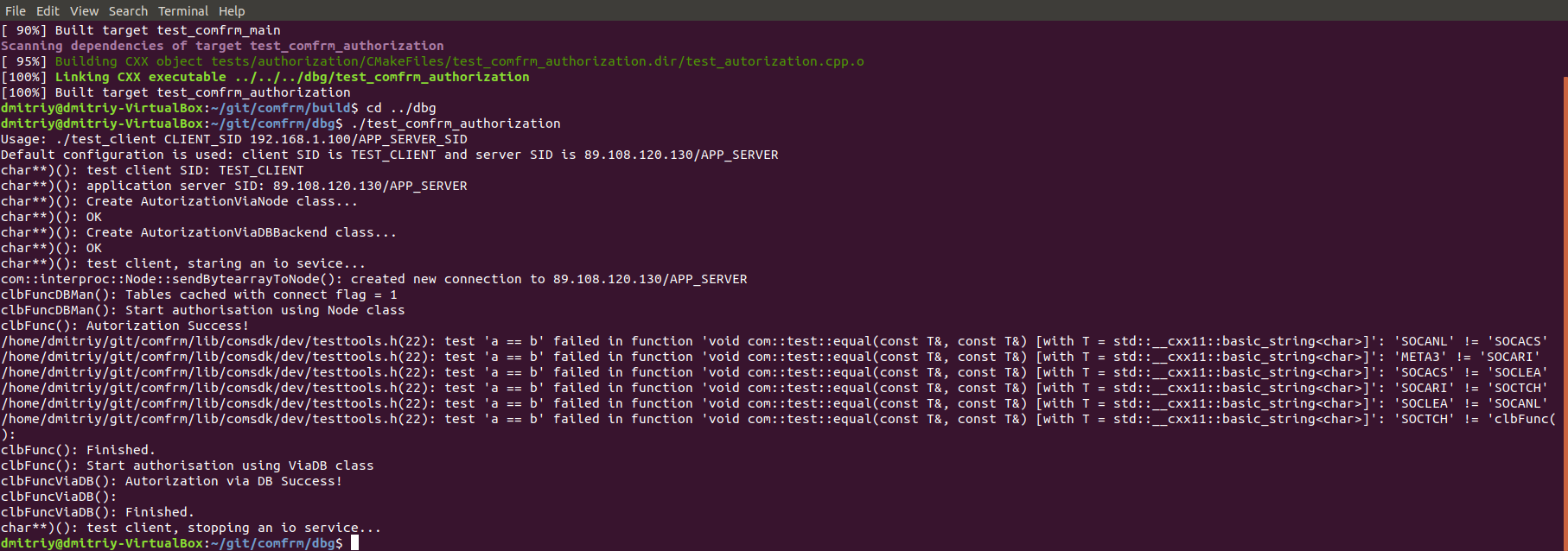


Рис. 6 Результат работы разработанного теста при передаче корректных логина и пароля в функцию authorize()

Результат работы разработанного теста для некорректных логина и пароля представлены на рисунке 7.

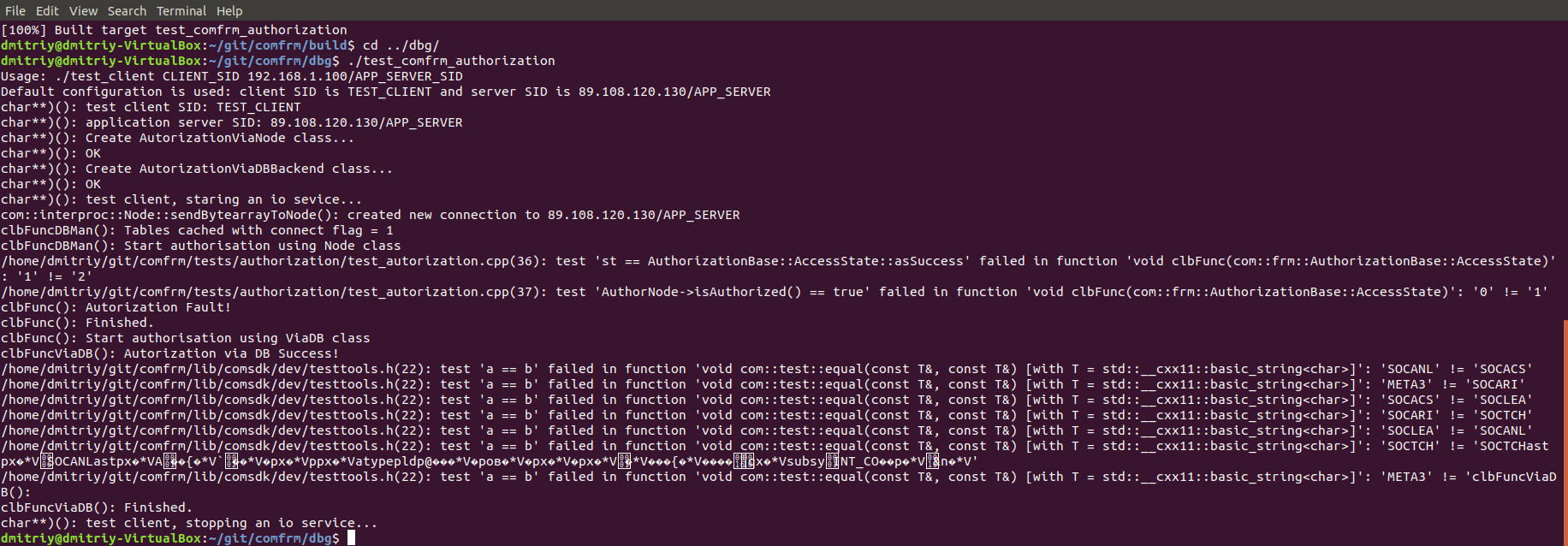


Рис. 7 Результат работы разработанного теста при передаче некорректных логина и пароля в функцию authorize()

Таким образом, с использованием разработанного алгоритма для тестирования функций для авторизации пользователей на сервере приложений были выявлены следующие ошибки в работе:

* для объекта класса AuthorizationViaNode метод getAITypes() и метод getSubsystems() возвращают список строковых значений, который не совпадает со списком, обговоренным в постановке задачи;
* для объекта класса AuthorizationViaDBBackend метод getAITypes() и метод getSubsystems() возвращают список строковых значений, который не совпадает со списком, обговоренным в постановке задачи;
* для объекта класса AuthorizationViaDBBackend метод authorize() завершается успешно в независимости от корректности передаваемых логина и пароля.

**ЗАКЛЮЧЕНИЕ**

В результате работы был проведен анализ работы текущей версии   
web-клиента распределенной вычислительной системы, целью которого было определение необходимых доработок платформы, которые позволяли бы оптимизировать работу при выгрузке больших объемов данных со стороны сервера на сторону клиента. По завершению анализа текущей версии   
web-клиента было определено, что необходимо расширить текущий функционал сервера приложений распределенной вычислительной системы. В следствии того, что некоторый функционал текущей версии сервера приложений находится на стадии тестирования, была поставлена задача реализации теста для функций, позволяющих проводить авторизацию пользователей на стороне сервера приложений. После реализации и запуска разработанного теста был выявлен ряд доработок, которые необходимо внести в существующий функционал по авторизации пользователей.