Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Легалов
подпись инициалы, фамилия
«___» _____ 2016 г.

БАКЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специализации

Модернизация интерпретатора обработки запросов для системы автоматизации выполнения лабораторных работ по курсу «Операционные системы»
тема

Пояснительная записка

Руководитель	_____	ст.преподаватель	Л. В. Макуха
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Д. К. Шадрин
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	к.т.н., доцент	В. И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2016

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И Легалов
подпись инициалы, фамилия
«___» _____ 2016 г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы
бакалаврской работы, дипломного проекта, дипломной работы, магистерской диссертации

Студенту _____ Шадрину Денису Константиновичу_____
фамилия, имя, отчество

Группа ___КИ12-09Б___ Направление (специальность) ___09.03.01.05 _____
номер код

Высокопроизводительные вычислительные системы на базе больших ЭВМ
наименование

Тема выпускной квалификационной работы
Модернизация интерпретатора обработки запросов для системы автоматизации выполнения лабораторных работ по курсу «Операционные системы»
Утверждена приказом по университету № _____ от _____

Руководитель ВКР __Макуха Л.В., старший преподаватель кафедры ВТ____
фамилия, инициалы, должность, учёное звание и место работы

Исходные данные для ВКР лабораторные работы по предмету_____
Операционные системы_____
_____

Перечень разделов ВКР анализ задания на проектирование, разработка_____
архитектуры и основных технических решений._____
_____

Перечень графического материала демонстрация Power Point._____

Руководитель ВКР _____ Макуха Л.В.
подпись фамилия, инициалы

Задание принял к исполнению _____ Шадрин Д. К.
подпись фамилия, инициалы

«___» _____ 2016 г.

РЕФЕРАТ

Тема данной выпускной квалификационной работы (ВКР) «Модернизация интерпретатора обработки запросов для системы автоматизации выполнения лабораторных работ по курсу «Операционные системы»». Пояснительная записка содержит 32 страницы текстового документа, 11 иллюстраций и 6 использованных источников.

Объект работы — симулятор для выполнения лабораторных работ, разработанный год назад.

Цель работы: модификация существующей системы проверки знаний по дисциплине «Операционные системы», добавив функцию симулирования работы в текстовом редакторе.

Результатом работы станет возможность многострочного последовательного ввода различных команд студентом при выполнении такого задания, как вызов текстового редактора.

В итоге в симуляторе появилось новое задание, а также появилась возможность обучиться работе с текстовым редактором в режиме командной строки.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день симуляционные формы обучения становятся все более востребованными в современном обучении. Понятие «симулятор» подразумевает под собой такой процесс, как имитация реального процесса, которая зачастую происходит через компьютер или же другие технологические устройства, для приобретения опыта [1].

Симуляционный подход — это подход учебного процесса, при котором студент действует в игровой ситуации и знает об этом.

В ходе изучения студентом различных предметных областей, усваивается достаточно большой объем учебного материала. Но при этом полного понимания материала не будет. Поэтому требуется практическое применение для полного усвоения той или иной предметной области. Практическое применение можно получить как в реальности, так и в симуляторе.

Исключением не является и такая дисциплина, как «Операционные системы». Так как большинство лабораторных работ представляют собой одинаковые задания, то наличие такого симулятора однозначно ускорит процесс выполнения и сдачи лабораторных работ, а также снизит нагрузку, которая накладывается на преподавателя.

На текущий момент существует симулятор по дисциплине. Но данная среда обучения требует улучшений. Отсутствие симуляции текстового редактора не позволяет в полной мере изучить возможности командной строки.

1 Анализ технического задания

Целью данной бакалаврской работы является модификация существующей системы проверки знаний по дисциплине «Операционные системы», добавив функцию симулирования работы в текстовом редакторе.

Из поставленной цели были выявлены следующие задачи:

- Изучить возможности существующего симулятора;
- Выявление элементов, нуждающихся в модификации;
- Модернизация функционала симулятора и тестовая эксплуатация.

1.1 Анализ существующего симулятора

На данный момент времени уже существует симулятор, отвечающий за автоматизацию выполнения лабораторных работ по дисциплине «Операционные системы». В данном симуляторе уж есть набор определенных функций, реализующий сдачу лабораторных работ, симулирующий такие операционные системы, как Windows и Linux.

Симулятор состоит из различных составляющих, таких как система управления базами данных (СУБД), средства разработки, отвечающие за работу логики сервера, а также средства, отвечающие за разработку работы клиентской части приложения и веб-интерфейса.

Существует API-функционал для обеспечения работоспособности сервера. Например, файл ***create_and_edit_labwork.php*** отвечает за администрирование лабораторных работ, существующих на данный момент. Наиболее важные методы, находящиеся в этом файле это:

- ***createLabwork***, отвечающий за создание новой лабораторной работы;
- ***createProblem***, который позволяет создать новое задание для выбранной лабораторной работы;
- ***getLabwork***, позволяющий получить список доступных на текущий момент лабораторных работ;

• ***loadProblem***, который возвращает список заданий, необходимых для выполнения той или иной лабораторной работы.

Также немаловажен файл ***run_session.php***. Данный файл необходим для разработки API-функционала для работы с сессиями.

Данный файл включает в себя наиболее важные функции для непосредственной работы при выполнении студентом лабораторной работы.

Например, функция ***getExecuteResult*** принимает от клиента введенные данные для сравнения введенного с эталоном команд. При выполнении данной функции сервер обращается к файлу ***parser.php*** для выполнения функции ***compareCommand***, которая в свою очередь анализирует введенные данные пользователем с тем, что хранится в базе данных. Результатом работы функции является верное или неверное выполнение того или иного задания.

Для того чтобы API функционал, написанный на PHP работал с PHP Data Objects (PDO) предлагает единые методы для работы с различными базами данных, хотя текст запросов может немного отличаться. Так как многие СУБД реализуют свой диалект SQL, который в той или иной мере поддерживает стандарты ANSI и ISO, то при использовании простых запросов можно добиться совместимости между различными языками. На практике это означает, что можно достаточно легко перейти на другую СУБД, при этом не меняя или частично изменяя код программы. PDO не использует абстрактных слоёв для подключения к БД, наподобие ODBC, а использует для разных БД их «родные» драйверы, что позволяет добиться высокой производительности. В настоящее время для PDO существуют драйверы практически ко всем общеизвестным СУБД и интерфейсам. Так же позволяет работать сразу с несколькими базами данных одновременно.

Написанная логика сервера использует расширение PDO почти в каждом методе.

Также у данного симулятора присутствует режим администрирования. В этом режиме преподаватель может создавать/изменять лабораторные работы, а

также просматривать статистику выполнения лабораторных работ студентами, указав период, за который ему необходимо просмотреть статистику.

На рисунке 1 отображена панель создания/изменения лабораторных работ. В данном меню преподаватель может создавать лабораторные работы, добавляя туда новые задания, либо изменять текущие лабораторные работы, меняя задания или добавляя новые.

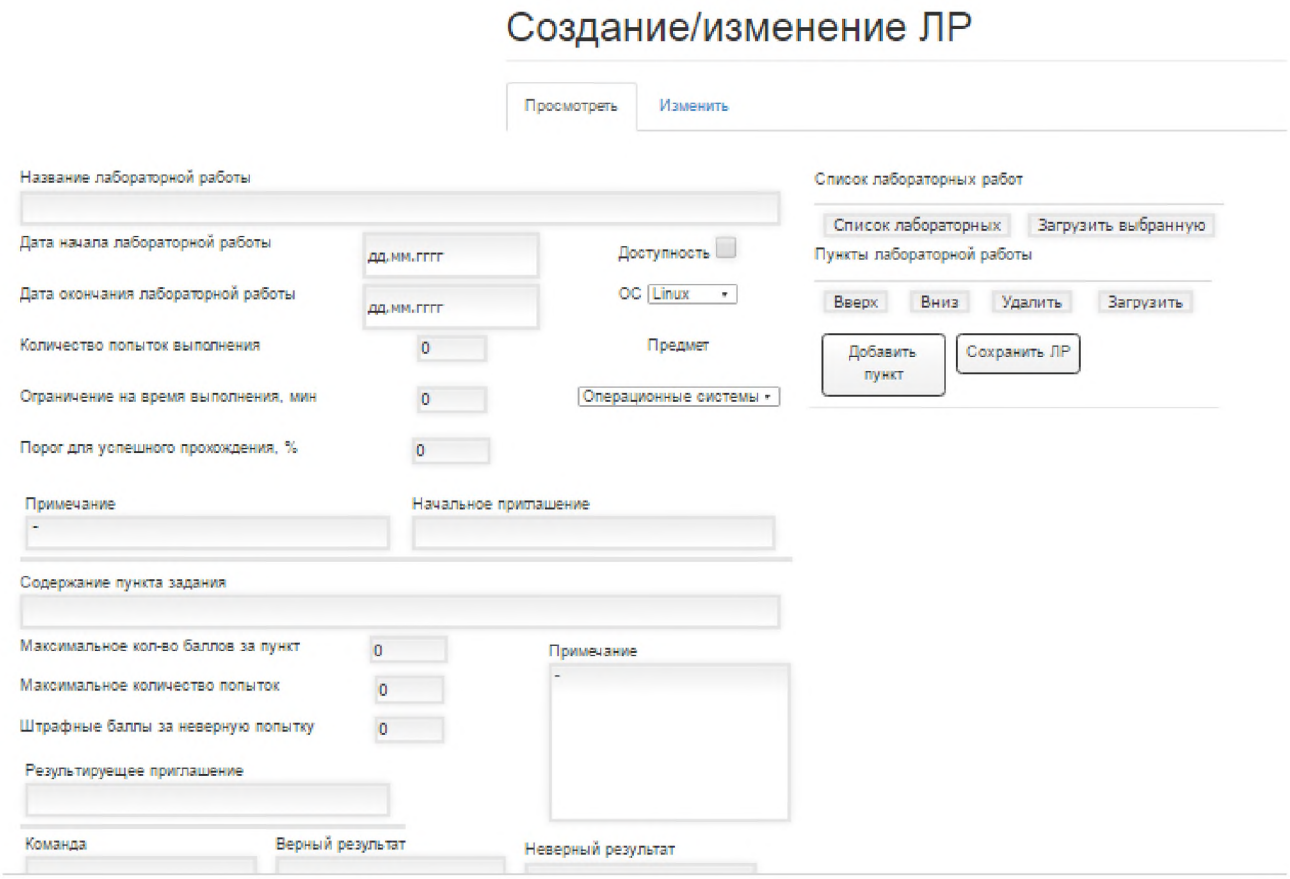

Рисунок 1 — Режим «создание/изменение» лабораторных работ

На рисунке 2 показана статистика выполнения лабораторных работ. В данных статистики содержится id сессии, ФИО студента, тема лабораторной работы, время, затраченное на выполнение, сдана ли лабораторная.

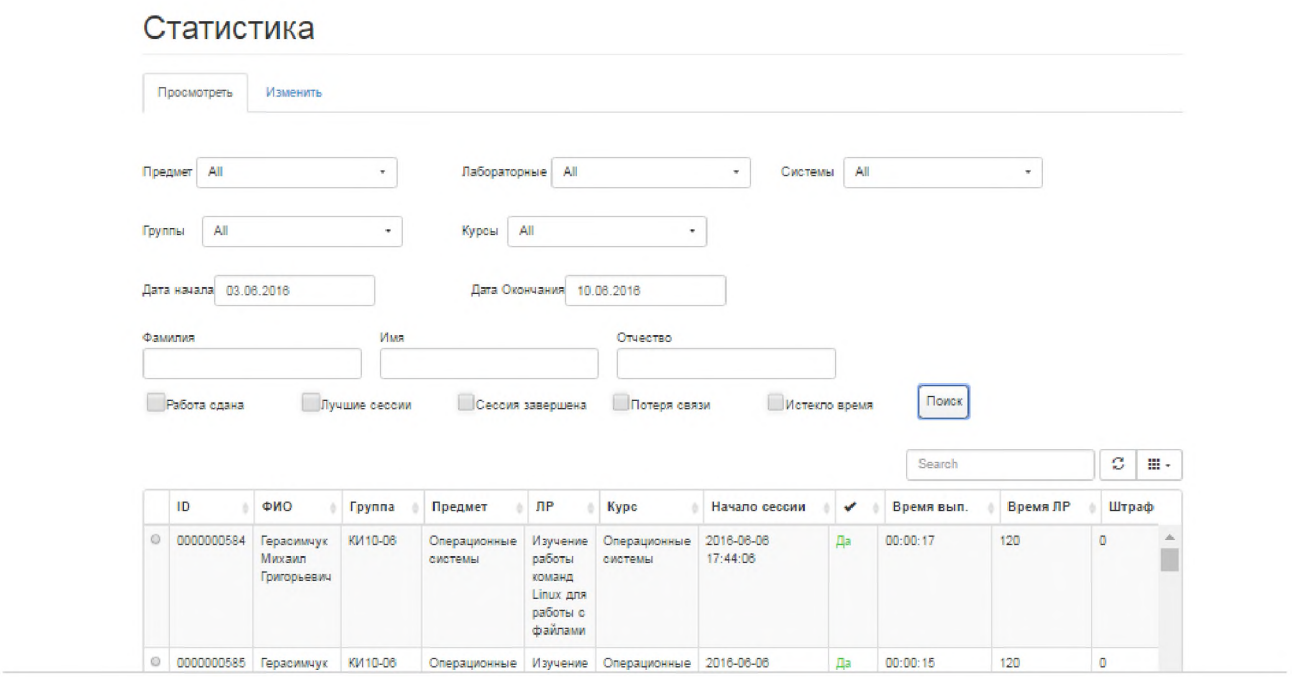

Рисунок 2 — Меню статистики выполнения лабораторных работ

При запуске лабораторной работы создается сессия при помощи функции ***createSession***. Данной сессии присваивается свой индивидуальный id. Также существует функция ***getInfoProblem***, отвечающая за вызов из базы данных текущего задания. Наиболее важной на стороне клиента является функция ***getExecuteResult***. В данную функцию отправляется, строка, введенная пользователем, id текущей сессии и приветственное окно, которое отвечает за то, что будет отображено в следующем задании при выполнении лабораторной работы. Затем вызывается функция, на стороне сервера, созданная для сравнения введенной пользователем строки с эталоном команды, взятым из базы данных. После сравнения идет проверка, является ли данный пункт конечным при выполнении данной работы. Если да, то выводится итоговый результат сдачи лабораторной работы. Если нет, то происходит проверка, закончились ли попытки для выполнения данного задания в текущей работе. После проверки, при условии, что введенная строка верна выводится приветственное окно для выполнения последующих заданий лабораторной работы. Если же введенная строка не верна, но попытки еще есть, то студенты предлагается повторно ввести ответ на

задание. Если попытки завершены и строка введена неверно, то выводится приветственное окно для следующего задания.

За выполнение каждого пункта лабораторной работы студенту, в зависимости от количества верно введенных команд, начисляются баллы, которые переводятся в проценты выполнения лабораторной работы.

По окончании выполнения лабораторной работы подводится итог. Чтобы выполнить лабораторную работу, необходимо набрать 70% и более от всей лабораторной работы.

Работа данного симулятора заключается в проверке определенных знаний и навыков, приобретаемых студентом в процессе обучения.

Первоначальная задача студента выбрать лабораторную работу для ее сдачи. После выбора соответствующей лабораторной работы для сдачи перед студентом появляется задание, которое ему необходимо выполнить. Затем, студенту необходимо ввести верные данные (команду) для успешного выполнения задания. На выполнение задания студенты выдается ограниченное количество попыток. Если студент не управляется за количество попыток, отведенное на выполнение текущего задания, то задание считается проваленным, и осуществляется переход к следующему заданию.

По мере выполнения заданий, за верно выполненные задания студенту начисляются баллы. После завершения выполнения лабораторной работы сдаваемому выводится итог, а также считается ли текущая лабораторная работа сданной или нет.

Пример выполнения лабораторной работы по операционной системе Windows продемонстрировано на рисунке 3.

Текущее задание: Создать папку TEST в C:\USERS	4/7	3
Примечание: (mkdir, md) (C:\Users\Test, Test);	01:59:11	

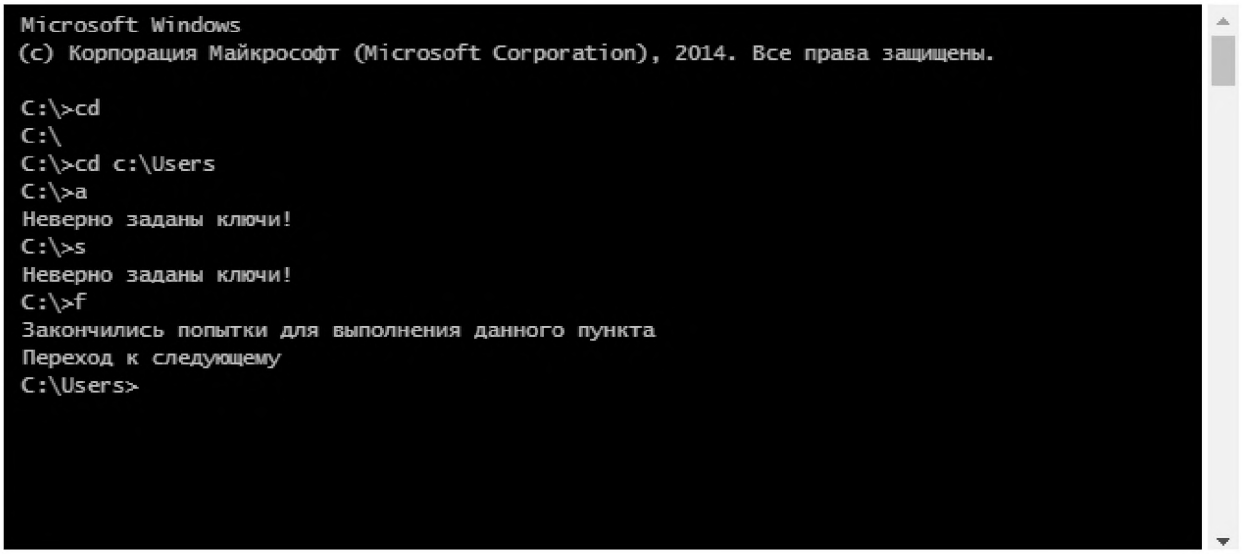

Рисунок 3 — Пример выполнения лабораторной работы по операционной системе Windows

Итоги лабораторной работы показаны на рисунке 4.

Текущее задание: Отобразить атрибуты скопированных файлов	7/7	2
Примечание: (ATTRIB); (ATTRIB) (*.txt, c:\users\test*.txt);	01:59:28	

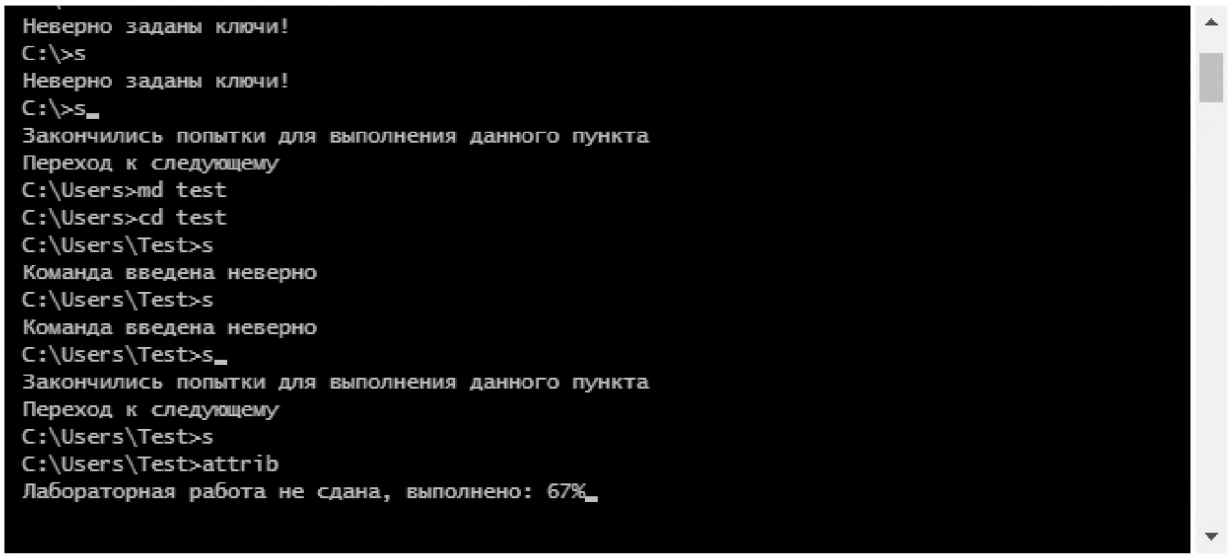

Рисунок 4 — Итоги выполненной лабораторной работы

В данном случае лабораторная работа считается невыполненной, поскольку не был преодолен порог, необходимый для сдачи данной лабораторной работы.

У данного симулятора имеется ряд достоинств:

- Для сдачи лабораторной работы по той или иной теме не обязательно наличие выполненных всех верно пунктов лабораторной работы. Необходимо лишь преодолеть порог;
- Лабораторную работу можно выполнить, не находясь в аудитории, т.е. ее можно выполнить дистанционно и студенту не обязательно находиться в аудитории при ее выполнении;
- Наличие такого симулятора заметно снижает нагрузку, которая ложится на преподавателя.

Но у данного симулятора также имеется и недостаток:

- У студента на выполнение задания имеется ограниченное количество попыток, когда в реальной операционной системе он может вводить команду, отвечающие за правильность выполнения ровно столько раз, сколько нужно студенту для выполнения задания.

Подводя итоги, можно сказать, что данный продукт снижает нагрузку с преподавателя. К тому же, он позволяет выполнять лабораторные работы студентам дистанционно, что также снижает нагрузку со студентов, позволяя выполнять лабораторные работы дома, не приходя для сдачи данных работ на практические занятия.

1.2 Обзор существующих средств разработки

Для реализации данного симулятора, созданного Герасимчуком М. Г., была использована СУБД, которая позволила хранить данные о лабораторной работе, т.е. её текущие задания для выполнения работы. Также были использо-

ваны средства разработки, которые взаимодействуют с СУБД, т.е. отвечают за логику сервера. К тому же был сделан веб-интерфейс для обучающегося и обработку данных, вводимых пользователем. Для реализации этих целей были использованы такие средства разработки как:

- СУБД MySQL;
- PHP, отвечающий за разработку логики сервера;
- HTML, отвечающий за разработку веб-интерфейса для симулятора;
- JavaScript для обработки вводимых пользователем данных;
- CSS для отображения консоли.

1.3 Техническое задание

1.3.1 Наименование системы

Полное наименование системы:

Симулятор командного интерфейса для обучения и контроля знаний студентов по предмету Операционные системы.

Краткое наименование системы:

Симулятор командного интерфейса.

1.3.2 Назначение и цель модернизации системы

1.3.2.1 Назначение системы

Симулятор командного интерфейса — система, предназначенная для:

- Симуляции консоли конфигурирования операционных систем Windows и Linux;
- Проверки и контроля знаний студентов по предмету Операционные системы;
- Возможности многострочного последовательного ввода команд;

• Организации процесса дистанционного выполнения лабораторных работ студентами дистанционно без участия преподавателя для проверки результатов выполнения работ;

• Сбора подробной статистики результата работы с симулятором по каждому тестируемому, и записи результатов в базу данных.

1.3.2.2 Цель модернизации системы

Основной целью модернизации текущей системы является симуляция текстового редактора, который бы позволил студенту реализовывать выполнение лабораторной работы под ОС Linux, симулируя ввод многострочного сообщения, которое интерпретируется в зависимости от поставленной задачи.

1.3.3 Требования к системе

1.3.3.1 Требование к структуре и функционалу системы

Функциональная структура системы должна включать в себя модули выполняющие задачи администрирования системы, симуляции среды, сбора и анализа статистических данных, а также обеспечивающие «подмодули», позволяющие реализовать функционал основных модулей системы.

Функциональная часть текстового редактора подразумевает то, что введенный текст будет сравниваться с эталоном, находящемся в базе данных.

1.3.3.2 Требования к способам и средствам связи для информационного обмена между компонентами системы

Информационный обмен между модулями логики сервера системы и веб-интерфейсами системы должен осуществляться через всемирную систему объединённых компьютерных сетей для хранения и передачи информации — ин-

тернет посредством использования стандартизированных протоколов и форматов обмена данными.

1.3.3.3 Требования к совместимости со сторонними системами

Веб-интерфейсы системы должны иметь возможность интеграции в системы управления обучением, например, такие как Moodle.

1.3.4 Требования к видам обеспечения

1.3.4.1 Требования к лингвистическому обеспечению

Языки программирования

Разработка программного обеспечения должна вестись с использованием языка программирования PHP, HTML5 и СУБД MySQL.

Языки взаимодействия пользователей и системы

Основным языком взаимодействия пользователей и системы является русский язык. Взаимодействие пользователя с ПК должно осуществляться на русском языке.

1.3.4.2 Требования к программному обеспечению

Программное обеспечение должно быть представлено в виде комплекта файлов необходимых для развертывания логики сервера системы, в том числе файлов содержащих SQL-запросы для верного функционирования, а также дополнения записей в соответствующие места таблиц базы данных. Веб-интерфейс клиентской части должен верно отображать данные, хранящиеся в базе данных.

1.4 Итоги анализа

В результате анализа задания на бакалаврскую работу была сформирована текущая цель для модернизации текущего состояния данной системы и требования к ее разработке.

Актуальность данной системы в том, что она не только снизит нагрузку с преподавателя, но и сделает процесс сдачи лабораторных работ гораздо удобней для студентов, нежели было до этого.

В соответствии с заданием был выбран тот же набор средств разработки, что и при разработке изначальной версии симулятора.

2 Модернизация функционала разработанной системы

2.1 Задачи для модернизации приложения

Оценив текущие возможности данного симулятора, было выявлено, что на данный момент возможен лишь ввод покомандно. В следствие этого было решено расширить функционал симулятора, добавив возможность многострочного последовательного ввода команд при помощи симуляции текстового редактора. Для реализации данной возможности были поставлены следующие задачи:

- Добавить соответствующее задание в БД;
- Написать процедуру, реализующую выбор эталона для лабораторной работы;
- Модернизировать серверную часть приложения;
- Модернизировать клиентскую часть приложения.

2.2 Добавление задания

Проанализировав поставленную задачу, было решено, что новые таблицы в базе данных не нужны. Необходимо добавить задание через интерфейс, созданный для преподавателя новое задание, созданное для операционной систему Linux. Оно отображено на рисунке 5.

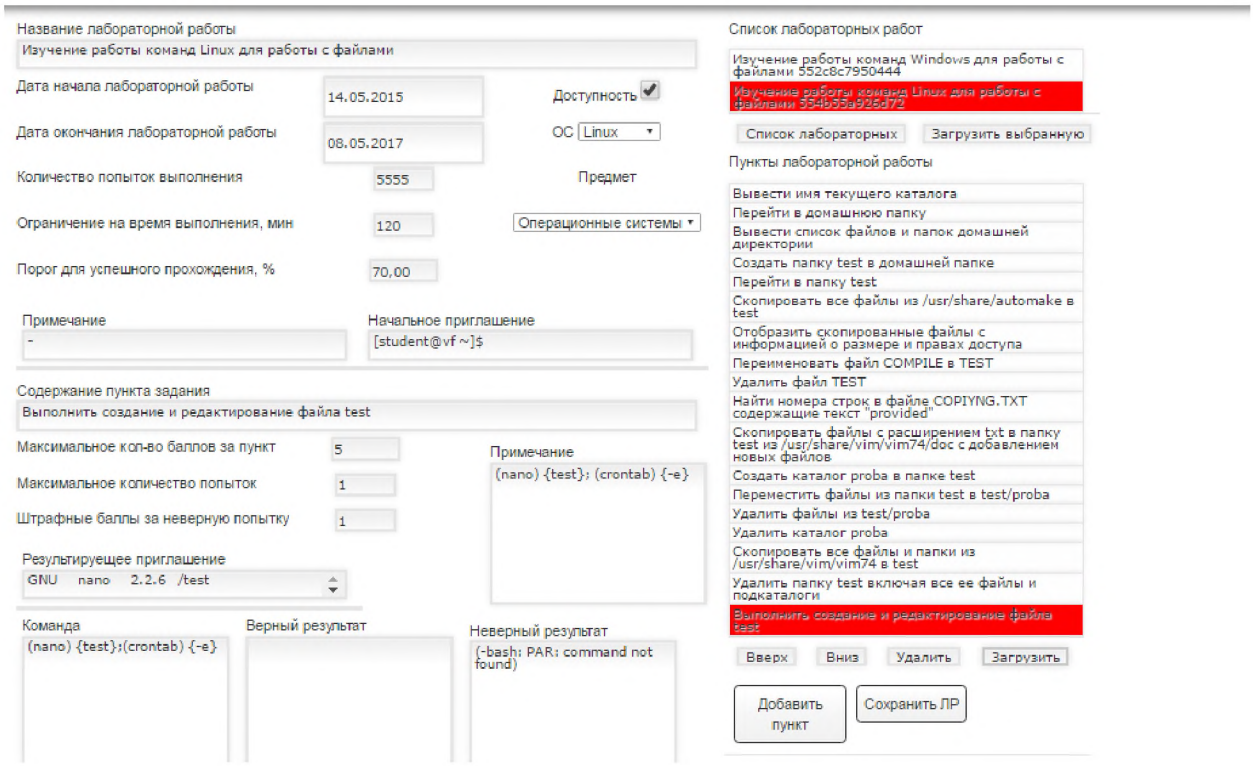

Рисунок 5 — Добавленная запись в таблицу «problem»

Данная запись отвечает за такое задание, как запуск текстового редактора с его отображением в дальнейшем.

В содержание пункта задания было добавлено описание самого задания. В колонку команда были добавлены команды, при вводе которых студент попадет в режим текстового редактора. В примечание также был добавлен эталон, подходящие для решения текущей задачи для удобства тестирования, т.к. эти данные отображаются на странице обучающегося. В колонку результирующего приглашения были помещены верхнее и нижнее подменю для того, чтобы в случае верно введенной команды отобразить текстовый редактор.

2.3 Добавление процедуры в СУБД

Для того, чтобы обеспечить возврат эталона на клиентскую сторону приложения необходимо написать процедуру, которая будет реализовывать соответствующие запросы к БД, и, при этом, будет универсальной, т.е. id лабораторной работы будет браться из id текущей сессии.

Ниже представлен листинг процедуры.

```
CREATE DEFINER=`sayu`@`%` PROCEDURE `get_info_fornano`(IN `pid_session` INT)
    DETERMINISTIC
BEGIN
DECLARE pid_lab INT;
SET pid_lab=(SELECT id_labwork FROM session WHERE id_session=pid_session);
SELECT problem.problem_command FROM problem WHERE problem.id_labwork=pid_lab;
END
```

2.4 Модернизированный API-функционал системы.

Программный модуль, реализованный до этого, отвечает за API-функционал системы, а также за связь между базой данных и клиентской стороной приложения.

В данном случае необходим метод, реализующий возврат на клиентскую часть приложения списка эталона. При выполнении заданий, созданных до этого, данные отправлялись на сервер и уже там проверялись на правильность. А для проверки ввода данных, написанных внутри текстового редактора, данные будут проверяться на стороне клиента.

На рисунке 6 отображена модернизированная схема API-функционала системы

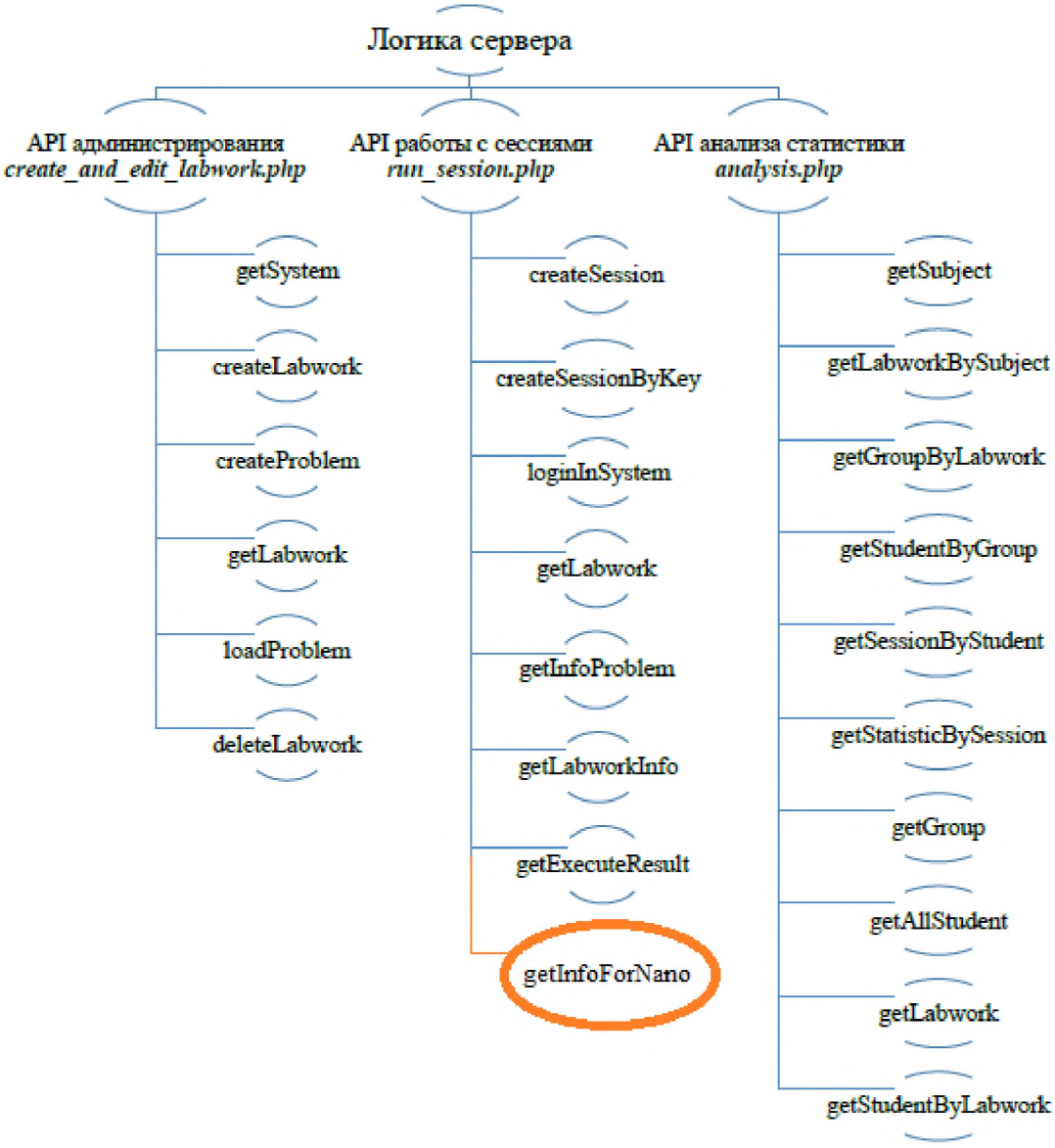

Рисунок 6 — Модернизированная схема API-функционала

К созданному API-функционалу ранее в файл ***run_session.php*** была добавлена следующая функция: ***getInfoForNano ($id_session)***;

@param int $id_session - идентификатор сессии;

@return TABLE_ROW - информация по текущему заданию, которая требуется ля клиента.

Данная функция предназначена для того, чтобы вернуть клиенту эталон, взятые из таблицы ***problem*** [2][3].

Ниже приведен код данной функции:

```
function getInfoForNano($id_session)
{
        $db=connect();
        $stmt = $db->query("SET @p0 = '$id_session';");
        $stmt = $db->query("CALL `get_info_fornano`(@p0);");// вызов процедуры «get_info_fornano»
        $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);// запись полученных значений
        echo jsonRemoveUnicodeSequences($rows);
        $db=NULL;
}
```

Алгоритм функции следующий. При вызове данной функции происходит подключение к базе данных. Затем выполняется SQL-запрос для того, чтобы «вытащить» из базы данных информацию об эталоне для выполнения заданий. В конце данные записываются в ассоциативный массив. Переменной для подключения к базе данных присваивается значение “NULL”.

Также, для работоспособности данной функции, он был добавлен в выбор API в файле ***run_session.php***. Это позволит при вызове данного метода со стороны клиента выполнить его. Код добавления данного метода приведен ниже.

```
switch ($act)
{
    case "createSession":
        $login=$_GET['login'];
        $pass=$_GET['pass'];
        $id_lr=(int)$_GET['id_lr'];
        createSession($login,$pass,$id_lr);
        break;
        …
--**********************************получение эталона из таблицы problem
    case "getInfoForNano":
        $id_session=(int)$_GET['id_session'];
        getInfoForNano($id_session);
        break;
```

```
--**********************************
        case "getLabworkInfo":
            $lw_key=$_GET['lw_key'];
            getLabworkInfo($lw_key);
            break;
            …
             case "isHelp":
            $command=$_GET['command'];
            $system=$_GET['system'];
            isHelp($command, $system);
            break;
}
```

2.5 Модернизация клиентской стороны симулятора

Клиентская часть приложения написана на языке программирования JavaScript. В ней были реализованы следующие функции:

- ***function CheckOS()*** — данная функция предназначена для определения ОС, В которой будет выполняться лабораторная работа;
- ***function createSession ()*** — данная функция создает сессию для выполнения лабораторной работы;
- ***function getInfoProblem (pid_session)*** — данная функция получает информацию о текущем задании. В функцию передается текущий id сессии;
- ***function getExecuteResult(pid_session, pcommand, pwel)*** — данная функция отправляет на сервер данные для проверки правильности введенной команды и, в зависимости от введенного, выдает результат. В функцию передается id текущей сессии, данные, введенные студентом и приветственное окно для выполнения следующего задания;
- ***function changeInfoTable (InfoProblem)*** — данная функция строит таблицу, отображающую текущее задание, примечание и количество попыток. В функцию передается ассоциативный массив, в котором хранятся данные для текущего задания лабораторной работы;

• ***function TimeLab()*** — данная функция позволяет отобразить время, которое осталось на выполнение лабораторной работы;

• ***function DisplayConsole()*** — данная функция отображает командную строку, в которую студент будет вводить данные, необходимые для выполнения лабораторной работы;

• ***function NewLine(n, prwel)*** — Данная функция осуществляет перевод курсора на n новых строк. В функцию передается количество строк, на которое необходимо сместить курсор и приветственное окно;

• ***function WriteText(string)*** — данная функция отображает текст на экране. В функцию передается строка, которую необходимо отобразить на экране;

• ***function ChangeCursor()*** — данная функция изменяет состояние курсора;

• ***function DisableCursor()*** — данная функция отвечает за гашение курсора;

• ***function HandleKeypress(evt)*** — данная функция отвечает за обработку клавиш. В функцию передается событие, т.е. нажатие клавиши;

• ***function HandleBackspace()*** — данная функция отвечает за обработку клавиши «BackSpace»;

• ***function HandleForwardArrow()*** — данная функция отвечает за обработку клавиши «стрелка вперед»;

• ***function HandleBackArrow()*** — данная функция отвечает за обработку клавиши «стрелка назад»;

• ***function HandleUpArrow()*** — данная функция отвечает за обработку клавиши «стрелка вверх»;

• ***function HandleDownArrow()*** — данная функция отвечает за обработку клавиши «стрелка вниз»;

• ***function HandleSpecialKeypress(evt)*** — данная функция отвечает за обработку «специальных клавиш»;

• ***function DisableHotKey()*** — данная функция отвечает за перехват «горячих» клавиш;

На рисунке 7, приведена блок-схема работы симулятора при сдачи лабораторной работы по ОС Linux.

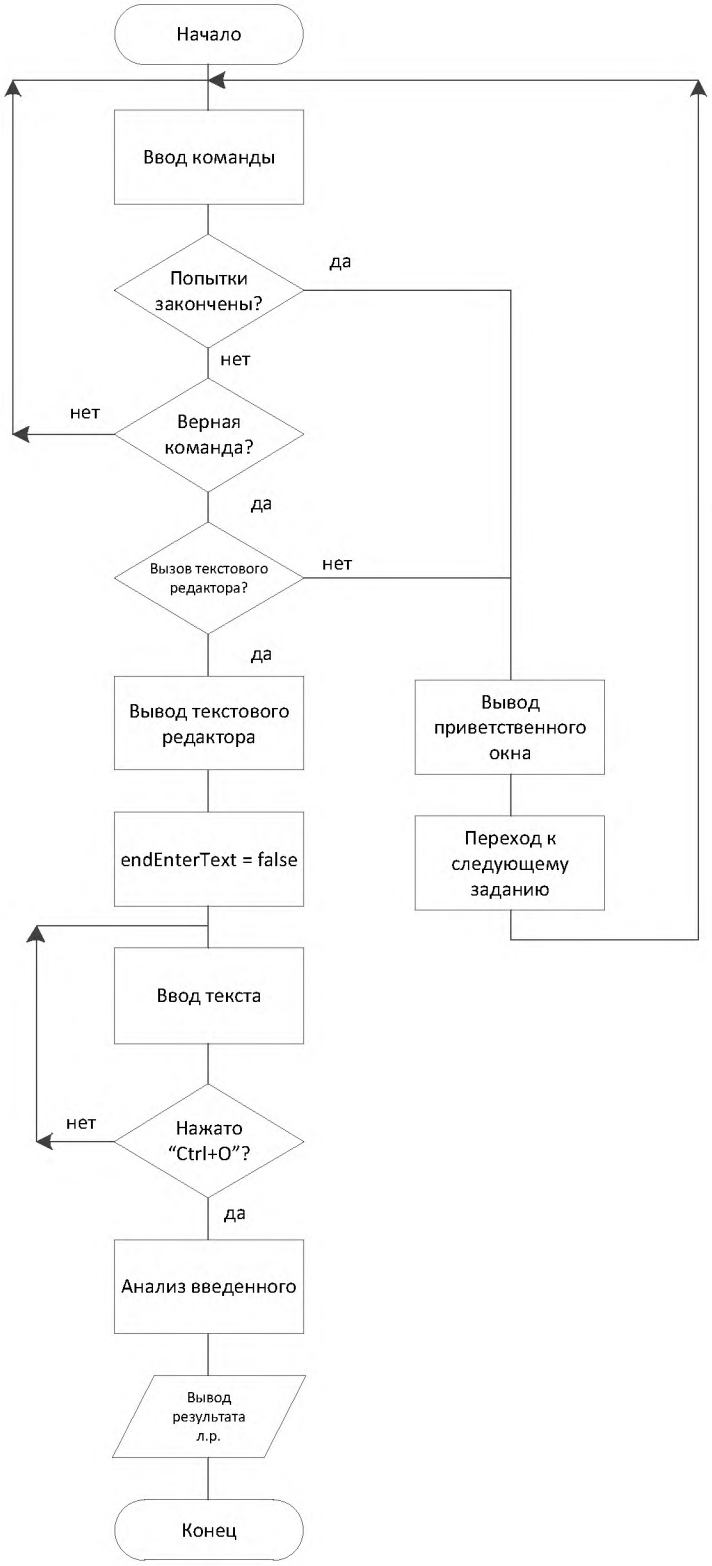

Рисунок 7 — Блок-схема алгоритма работы симулятора

Во время ввода пользователю доступно также нажатие специальных клавиш, таких как переместить курсор вверх, вниз, вправо, влево, а также клавиша «BackSpace».

Для начала необходимо разработать функцию, которая бы обращалась с запросом эталона для текущих заданий, хранящихся в базе данных для операционной системы Linux[5][6].

```
function getInfoForNano (pid_session) //получение информации о лабораторной (получает id сесиии)
{
    var XHR = window.XDomainRequest || window.XMLHttpRequest
    var xhr = new XHR();
    xhr.open("GET",host + 'act=getInfoForNano&id_session='+pid_session, true);// Запрос в БД вызывающий выполнение API «getInfoForNano»
    xhr.onload = function() {
    InfoForNano = eval( '(' + xhr.responseText + ')' );
        var regexp = /[a-zA-Z-~$*/]+/g
        for(var i=0; i<InfoForNano.length;i++)// Цикл, обеспечивающий распарсивание эталона, избавляя их от ненужных знаков
        {
            var clear_cmd = InfoForNano[i].problem_command;
            clear_cmd = clear_cmd.split(";");
            clear_cmd.forEach(function(el,ind,arr){
                var tmpStr = el.match(regexp)
                if(tmpStr)// если есть совпадение по переменной regexp
                {
                    tmpStr=tmpStr.join(" ");// соединение по пробельным вставкам
                    arr[ind] = tmpStr.trim();//очищение пробелов по краям
                }
            })
            InfoForNano[i].problem_command = clear_cmd;
        }
}
```

```
        xhr.onerror = function() {
        alert("Error")
        }
        xhr.send()
    }
```

Данная функция берет данные, выполняя API «*getInfoForNano*», откуда забирает эталон для операционной системы Linux. После того, как данные были получены, она освобождает эталон от лишних знаков, после чего записывает данные обратно в массив объектов.

Затем, необходимо было написать новую функцию, реализующую вывод на экран приглашения, по случаю нового задания [5][6]:

```
function func1(pr_wel, pcommand)// функция вывода приветственного окна
{
        var ncol=0;
        var mas_temp=[];
        var jj=pr_wel.split(/[\n\r]/);// Разбиение строк по enter'ам
        console.log(jj);
        for(var i=0;i<jj.length;i++)// Цикл для вывода приветственного окна
        {
            NewLine(1,jj[i]);
            if(jj[i]==" ")
                ncol++;
            var ss = jj[i].split(' ');
            for(var j=0; j<ss.length;j++)// цикл, заполняющий массив количеством слов в приветственном окне
            {
                if(mas_temp.length<3)
                {
                    if(ss[j]!="")
                        mas_temp.push(ss[j]);
                }
                else break;
            }
        }
```

```
if(mas_temp.length==3)// Проверка на то, была введена команда текстового редактора
                {
                        {
                                flag_nano = true;
                                count = ncol;
                                cursX = 10; cursY -= 15 * (ncol)
                                str="";
                                endEnterText = false;// флаг режима текстового редактора
                                var scrolling = document.getElementById('can');
                                        scrolling.scrollTop = cursY - 28;
                        }
                }
        }
```

Симуляция текстового редактора показана на рисунке 8.

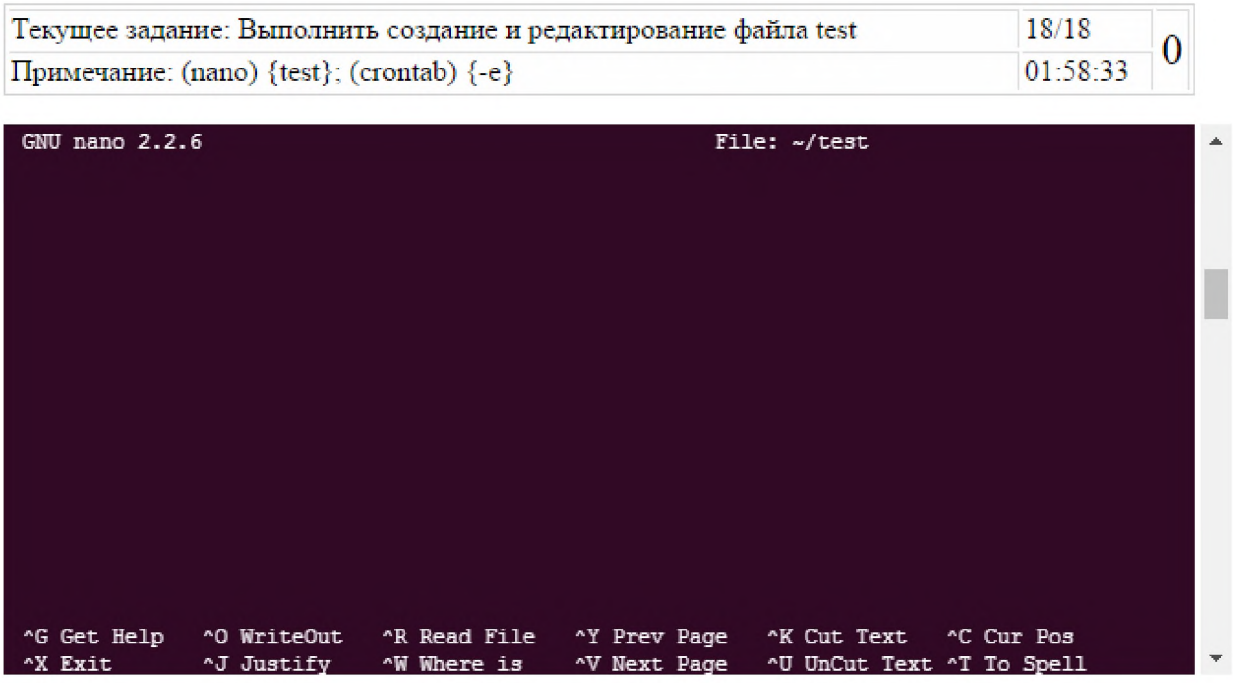

Рисунок 8 — Симуляция текстового редактора nano

Затем необходимо модернизировать две функции: HandleKeypress(evt) и DisableHotKey(), где evt — переменная, определяющая код нажатой клавиши.

Функция HandleKeypress отвечает за обработку нажатия обычной клавиши, а DisableHotKey() — за перехват «горячих клавиш».

Переменная endEnterText отвечает за перехват пользователем команды вызова текстового редактора.

Когда встретилась команда «crontab -e» или «nano test», то необходимо опустить флаг переменной endEnterText, что будет служить знаком, что был произведен вход в текстовый редактор.

Далее можно будет выполнять ввод строк, сколько их необходимо ввести пользователю. Наглядный пример ввода продемонстрирован на рисунке 9.

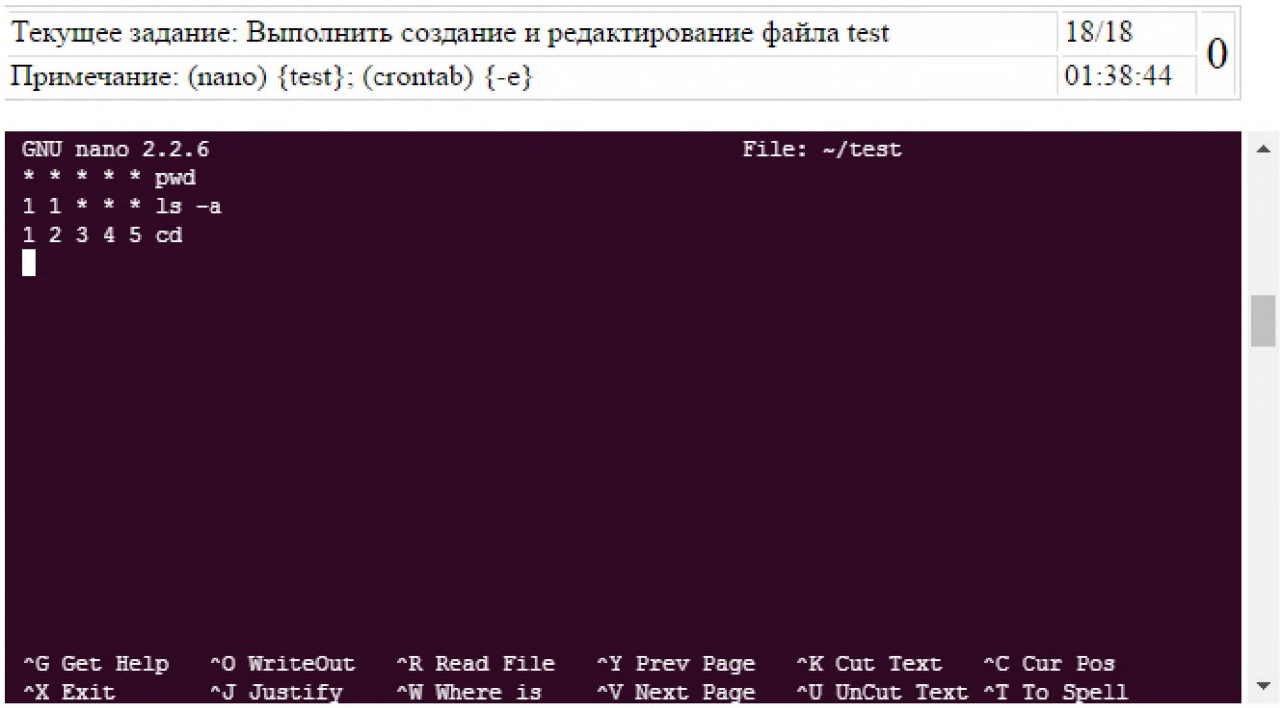

Рисунок 9 — Демонстрация возможности ввода нескольких строк за время выполнения одной команды

По окончанию ввода команд необходимо зажать такое сочетание клавиш как “ctrl+O”. После этого введенные пользователем команды будут отправлены на проверку при помощи функции for_nano (pcommand), где pcommand текущая введенная команда пользователем [5][6].

```
function for_nano(pcommand)
{
        console.log(InfoForNano);
        for(var I = 0; i<InfoForNano.length;i++)// цикл для сравнения эталона
        {
                for(var j = 0; j<InfoForNano[i].problem_command.length;j++)// цикл для
сравнения эталона, если вариантов несколько
                {
                        console.log(InfoForNano[i].problem_command[j],” “, pcommand)
                        if(InfoForNano[i].problem_command[j]==pcommand)
                        {
                                return true;
                        }
                }
        }
                return false;
}
```

Данная функция вызывается ровно столько раз, сколько новых строк ввел пользователем. После, происходит проверка введенной команды с эталоном команд. Если такая команда встречается, то функция возвращает true. Иначе — false.

Затем, происходит проверка на сравнение правильность введенных команд с количеством введенных строк. За каждую неверно введенную команду студент получает штрафной балл.

Демонстрация верно введенных команд продемонстрирована на рисунке 10, а неверных на рисунке 11.

Текущее задание: Выполнить создание и редактирование файла test	18/18	0
Примечание: (nano) {test}: (crontab) {-e}	01:38:22	

```
Задание решено верно!
Лабораторная работа не сдана, выполнено: 11%
```

Рисунок 10 — Пользователь ввел данные корректно

Текущее задание: Выполнить создание и редактирование файла test	18/18	0
Примечание: (nano) {test}: (crontab) {-e}	01:59:34	

```
Задание решено неверно!
Лабораторная работа не сдана, выполнено: 4%
```

Рисунок 11 — Пользователь ввел данные некорректно

2.6 Итоги разработки

В итоге разработки удалось реализовать симуляцию текстового редактора. Данная возможность позволит студентам вводить несколько команд сразу и мгновенно узнать результат. Однако, студент при вводе нескольких команд сразу должен быть чрезвычайно аккуратен, ведь малейшая невнимательность может привести к получению штрафного балла, и, как результат, неверному выполнению задания.

ЗАКЛЮЧЕНИЕ

На сегодняшний день наличие симуляторов просто необходимо для автоматизации учебного процесса обучающихся. Это позволит студентам выполнять задания дистанционно, без вмешательства преподавателя в контроль выполнения заданий.

Плюс данной системы в том, что ее клиентскую часть можно интегрировать в различные системы обучения. Также у симулятора есть возможность адаптации под различные дисциплины.

СПИСОК СОКРАЩЕНИЙ

СУБД — система управления базами данных

PHP — Hypertext Preprocessor

ОС — Операционная система

SQL — Structured Query Language

MS DOS — Microsoft Disk Operating System

ODBC — Open Database Connectivity

HTML — HyperText Markup Language

AJAX — Asynchronous Javascript And Xml

ВКР — Выпускная квалификационная работа

PDO — PHP Data Objects

БД — База данных

ФИО — Фамилия, имя, отчество

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Потенциал учебных симуляторов [Электронный ресурс] — Режим доступа: http://psyfactor.org/lib/naumov4.htm

2 Руководство по PHP [Электронный ресурс] — Режим доступа: http://php.net/manual/ru/index.php

3 Beginners PHP [Электронный доступ] — Режим доступа: http://www.homeandlearn.co.uk/php/php.html

4 Кроудер, Д. Создание web-сайта для чайников: 3-е издание. Д. Кроудер, — Москва: Диалектика, 2009. — 352 с.

5 Современный учебник JavaScript [Электронный ресурс] — Режим доступа: https://learn.javascript.ru/

6 The JavaScript Tutorial [Электронный ресурс] — Режим доступа: http://javascript.info/