ГУАП
КАФЕДРА № 51

ПРЕПОДАВАТЕЛЬ

доцент, к.т.н.		Е.М. Линский
должность , уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

Электронная ведомость

по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	5721		А.Е. Ковалева
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2019

Содержание

1 Функциональная спецификация

1.1 Цель работы

Создание электронной ведомости для двух типов пользователей.

1.2 Функциональность приложения

- Поддержка двух вариантов пользователей: учитель может просматривать,редактир и удалять ведомость, ученик только просматривать
- Возможность иметь несколько ведомостей под разные предметы
- Возможность динамически задавать количество строк и столбцов

1.3 Особенности реализации

- Для передачи данных между клиентом и сервером используется протокол HTTP.
- Для реализации web-приложения использован Apache Tomcat.
- Проект собирается с помощью Maven.
- Логины и пароли пользователей хранятся в файле.

2 Руководство пользователя

2.1 Авторизация и регистрация нового пользователя

Прежде чем начать пользоваться электронными ведомостями в полной мере необходимо авторизироваться. Для этого необходимо в строке «Enter your username» ввести категорию (учитель/ученик) пользователя, а в строке "Enter your password"ввести пароль.

Рис. 1: Окно ввода данных для авторизации пользователя.

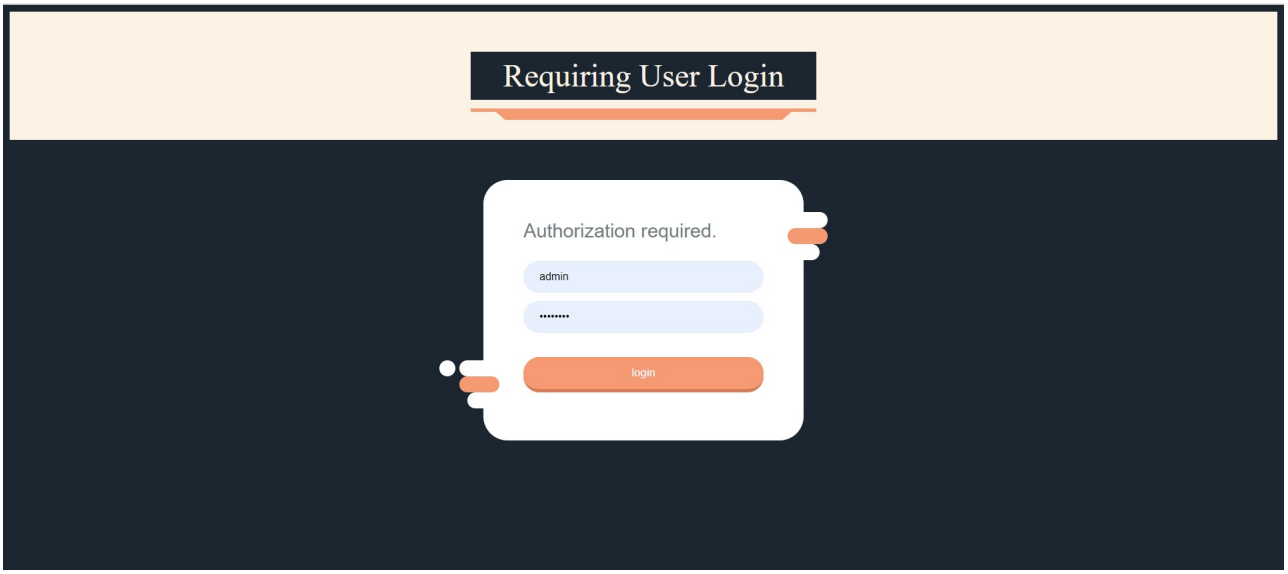

Рис. 2: Авторизация пользователя.

При неправильном логине или пароле выводится сообщение об ошибке с просьбой повторить попытку авторизации: "Error! No such user was found or the password was entered incorrectly. Try again."

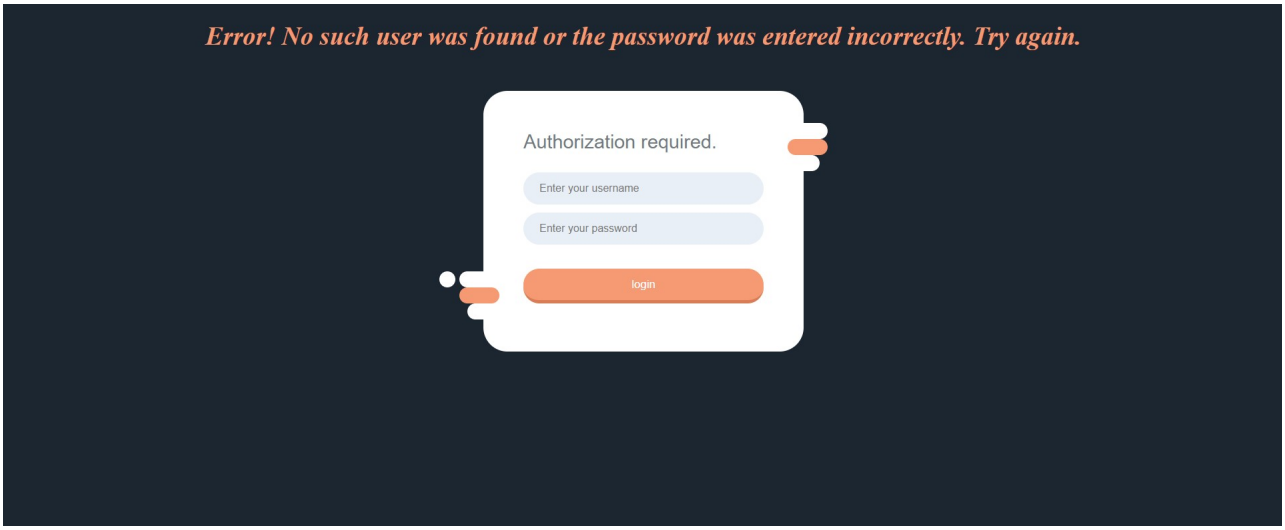

Рис. 3: Ошибка авторизации.

2.2 Создание электронной ведомости. Авторизация учителя

После успешной авторизации учителя, появляется возможность создать новую ведомость и отредактировать или удалить уже существующие(Рис. 4). Для того, чтобы создать новую ведомость учителю необходимо нажать на ссылку "Add new grade journal". После чего появится возможность ввести заголовок таблицы "Title ширину "weight"и задать высоту "Height". После нажатия кнопки "Add"создастся таблица с соответствующими значениями (Рис. 5). На странице будет выведено название ведомости, время создания и последнего изменения,сама ведомость и ссылки "Add string"и "Add column с помощью которых можно динамически задавать количество строк и столбцов (Рис. 6).

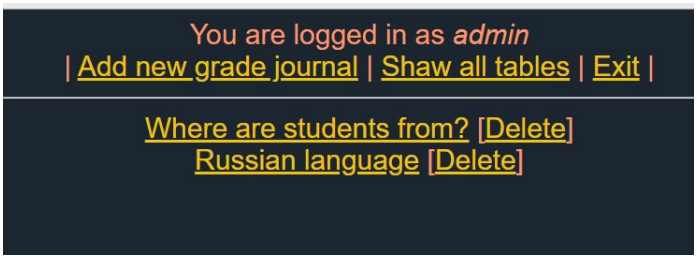

Рис. 4: Главная страница после авторизации

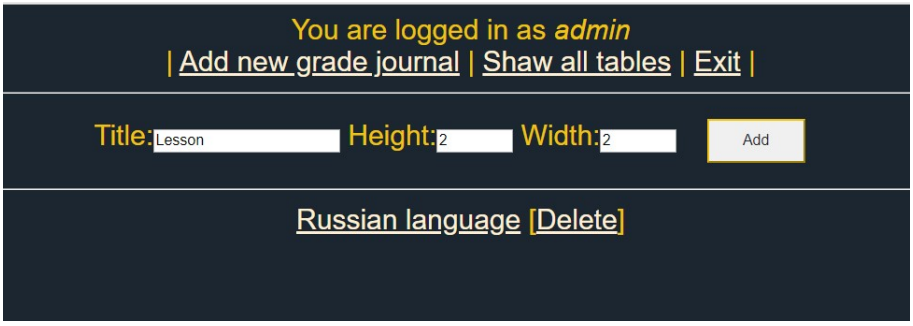

Рис. 5: Создание новой ведомости

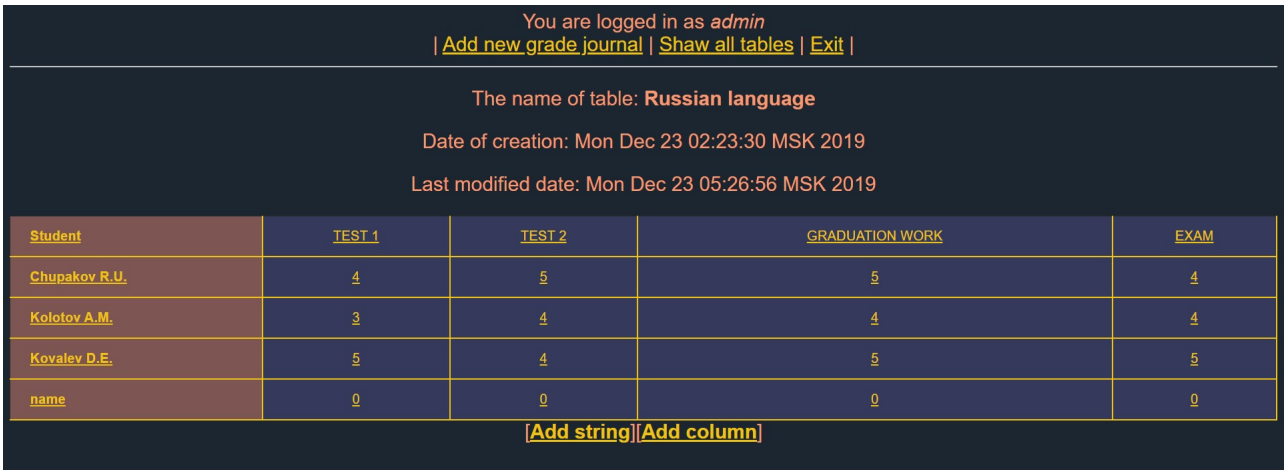

Рис. 6: Добавление столбцов и строк к существующей ведомости

Помимо добавления строк и столбцов, учитель может изменять названия колонок и данные в ячейках. Для этого необходимо нажать на ссылку в ячейке, после чего откроется страница для введения необходимого значения (Рис. 7).После нажатия кнопки "Change"текст в ячейке будет изменен (Рис. 8).

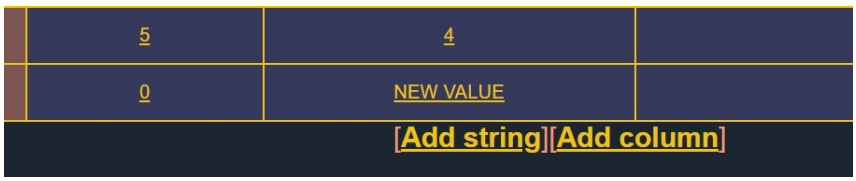

Рис. 7: Введение значения

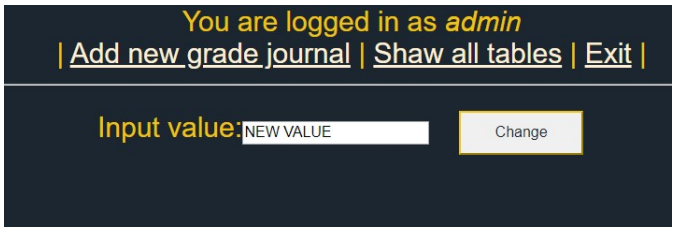

Рис. 8: Изменение значения в ведомости

Закончив работу с ведомостью учителю предоставляется возможность вернуться к списку всех ведомостей, добавить новую ведомость или выйти из системы (Рис. 9). На странице со списком всех ведомостей учителю предоставляеся возможность удалить ведомость нажав на кнопку "Delete"(Рис. 10),(Рис. 11).

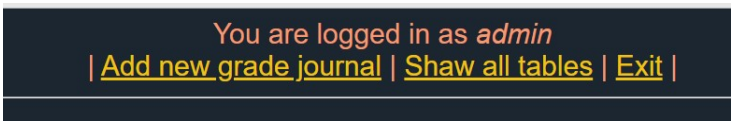

Рис. 9: Возможности сайта

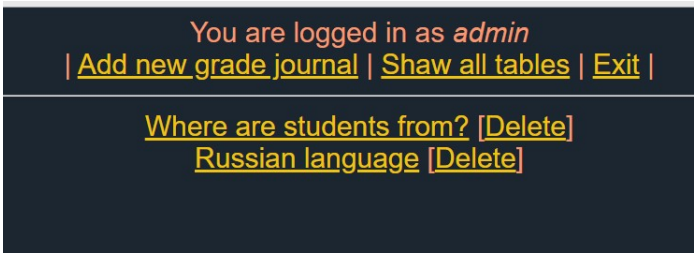

Рис. 10: До удаления ведомости

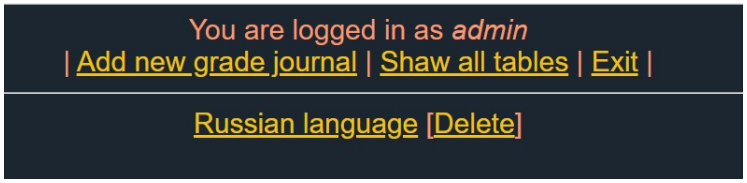

Рис. 11: После удаления ведомости

2.3 Просмотр ведомостей. Авторизация ученика

После авторизации ученику на главной странице выводится список всех ведомостей(Рис. 12), нажав на название которых, открывается таблица без возможности редактирования (Рис. 13). На странице имеются ссылки для возврата к меню со списком всех страниц и на выход из системы(Рис. 14).

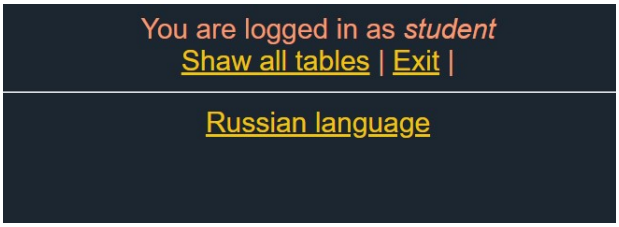

Рис. 12: Главная страница

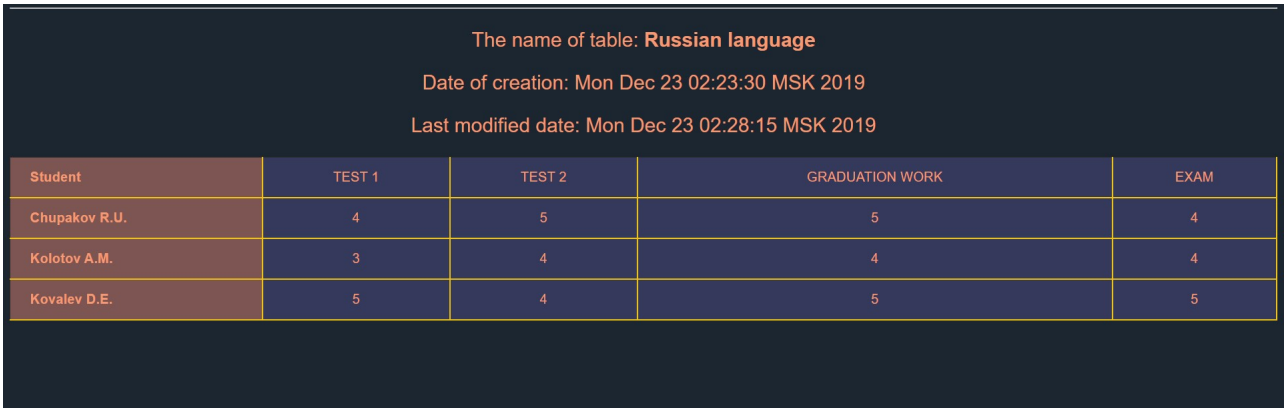

Рис. 13: Просмотр ведомости

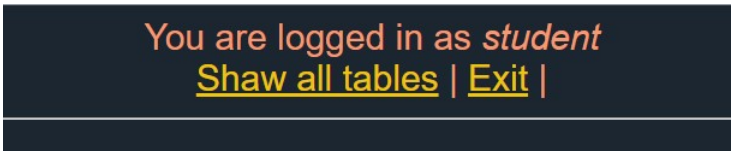

Рис. 14: Варианты работы со страницей

3 Архитектура программы

Архитектура программы соответствует двухзвенной архитектуре (two-tier, 2-tier),представленной на рисунке 15, где есть необходимость распределения трех базовых компонентов между двумя узлами (клиентом и сервером).

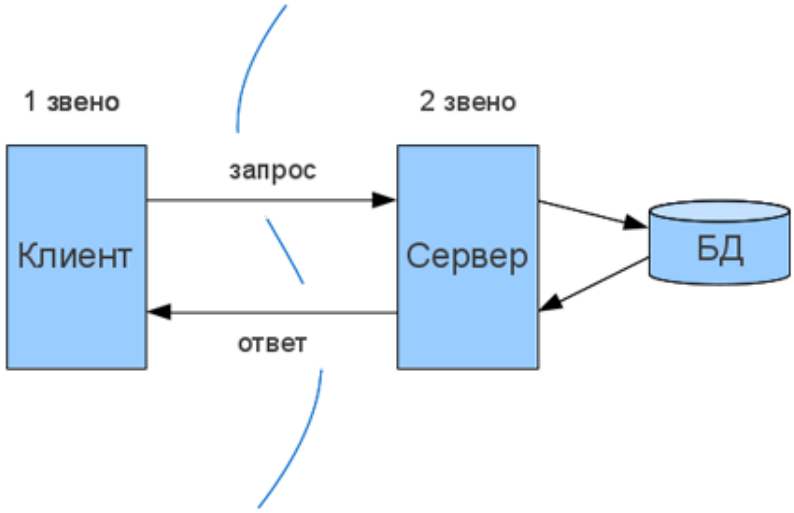

Рис. 15: Суть "клиент-серверной архитектуры приложения".

Сервлет взаимодействует с клиентами посредством принципа запрос-ответ:

- То есть клиент обращается к серверу посредством отправки HTTP-запроса.
- Сервер формирует необходимые данные (получает их из базы данных) и возвращает клиенту.

Проект состоит из частей:

- web
- Electronic table

3.1 Electronic table

Реализация логической части программы, определяющей работу с базами данных и хэширование пароля пользователей.

3.2 Web

Реализация графической части программы, отображающей интрефейс программы и вид электронных ведомостей.

4 Основные классы и методы

4.1 Electronic table

Основным классом является класс Main, содержащий всю информацию о передаваемых данных серверу от клиента. Содержит классы Exit и Login, в которых реализовано взаимодействие клиентов через сервер. Содержит методы, описывающие работу с электронной ведомостью, например:

- public void createNewTable () — создать новую таблицу
- public void addColumn()— добавить колонок в таблицу
- public void change() — изменить данные в таблице
- protected void doGet() - получаем список всех пользователей и т.д.

4.2 Web

Содержит реализацию клиентской стороны и реализацию графической части программы. Для создания клиентской стороны были использованы JSP-страницы. Внутри JSP-страниц будут используются HTML-теги <form/>. А для отправки данных из <form/> могут быть использованы только HTTP-запросы GET и POST. На каждой из страниц указан метод, который необходимо выполнить. Например, на стартовой странице "/login"указано, что после введения никнейма и пароля пользователя при нажатии на кнопку "login"будет выполнен method="post". На странице "/add отвечающей за создание новой ведомости, при нажатии на кнопку "Add будет выполнен method="post". Оставшуюся часть составляет обычная статическая HTML-разметка

5 Вспомогательные инструменты

5.1 Базы данных

В требованиях к исходному и исполняемому коду прописано, что под базой данных понимается текстовый файл. Он хранится на сервере. Для записи мы переводим объект в последовательность байтов, по которой затем его можно полностью восстановить, т.е. используем сериализацию, так как она позволяет«дать жизнь» объекту между запусками программы. Так как механизм сериализации связан с базовой системой ввода/вывода и переводит объект в поток байтов, для его выполнения необходимо создан выходной поток OutputStream, упакованный в ObjectOutputStream, и вызван метод writeObject(). Для восстановления был упакован InputStream в ObjectInputStream и вызван метод readObject().

5.2 MAVEN

Проект был собран в jar формат с помощью Maven.

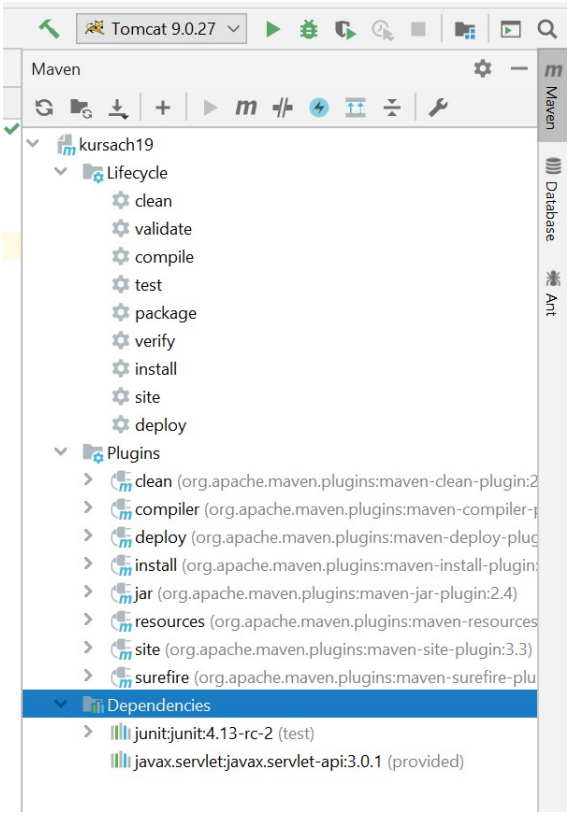

Рис. 16: Maven

5.3 JUnit

Для тестирования модели была использована библиотека JUnit. На рисунке 17 представлен результат тестирования классов.
Для подключения библиотеки JUnit используется maven.
Класс Controller послужил основой для написания юнит тестов. Тестирующие классы находятся в пакете test.
В IntelliJ IDEA тестирующие классы были созданы автоматически. Для этого нужно было нажать alt + enter на классе и выбрать «Create test». Далее были выбраны методы, которые нужно было протестировать.
В результате был создан класс Maintest с четырьмя выбранными методами: TestSize(),TestIndex (),TestName(),Testcontains(). Эти методы были реализованы самостоятельно.

- public void getSizeTest () .
  В данном методе осуществляется вызов метода getSize() и сравнивание его значения со значением "0".

- public void TestIndex ().
  В данном методе осуществляется тестирование метода public int returnIndex(String name) класса database. Данным методом создается ведомость "abc"и вызывается соответствующий метод returnIndex("abc"). Происходит сравнение полученного значения со значением "0". После чего созданная таблица удаляется tableBase.delete("abc").

- public void TestName().
  Здесь осуществялется проверка метода public String getTableName (int index) из класса database. Аналогично методу TestIndex () здесь создается ведомость с именем "abc вызывается метод из класса database с параметром 0, а именно getTableName(0). Таким образом, происходит сравнение получившегося значения (строки) с названием ведомости - строкой "abc"(assertEquals("abc tableBase.getTableName(0));). После сравнения ведомость удаляется (tableBase.delete("abc");).

- public void Testcontains().
  данным методом происходит тестирование метода public boolean containsTable (String name) из класса database. Создается новая ведомость с другим названием, в данном коде "a11". Вызывается метод из класса database с этим переданным параметром. После чего происходит сравнение полученного результата с истиной, т.е. значением "true"(assertTrue(tableBase.containsTable("a11"))

После чего, аналогично предыдущим методам, по завершению сравнения происходит удаление созданной ведомости (tableBase.delete("a11");).

В классе для указания на тестирующие методы, перечисленные выше, была использована анотация @Test.
Фикстурой @Before задан момент создания объекта класса перед каждым тестом.
Такой аннотацией обозначен и метод:

- public void init ().
  Данная аннотация говорит о том, что этот метод выполняется один раз перед всеми тестами. Здесь происходит создание сущности database, где происходит создание тестовых ведомостей.

Подробнее об использованных методах класса Assert:

- Метод assertEquals() проверяет на равенство переданных значений с возможной погрешностью delta. При выполнении заданных условий сообщает об успешном завершении, в противном случае — об аварийном завершении теста. При аварийном завершении генерируется ошибка java.lang.AssertionError. Другими словами, если два значения в методе отличны - будет вызвано исключение.

- В методе Test contains() assertTrue — проверяет на истину значение, в данном случае - содержится ли необходимая ведомость.
  Все методы класса Assert в качестве возвращаемого значения имеют тип void.

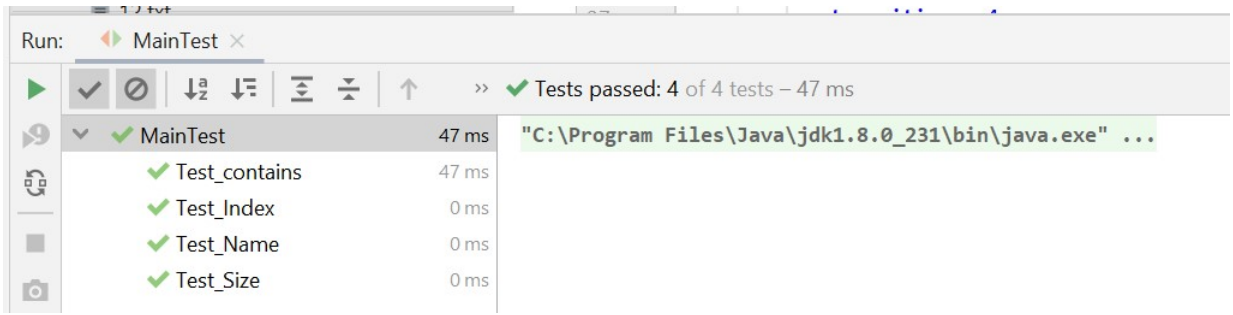

Рис. 17: Результат тестирования.

5.4 Github

Проект был расположен на Github.
Находится по ссылке https://github.com/Kova99999/Electronic-table

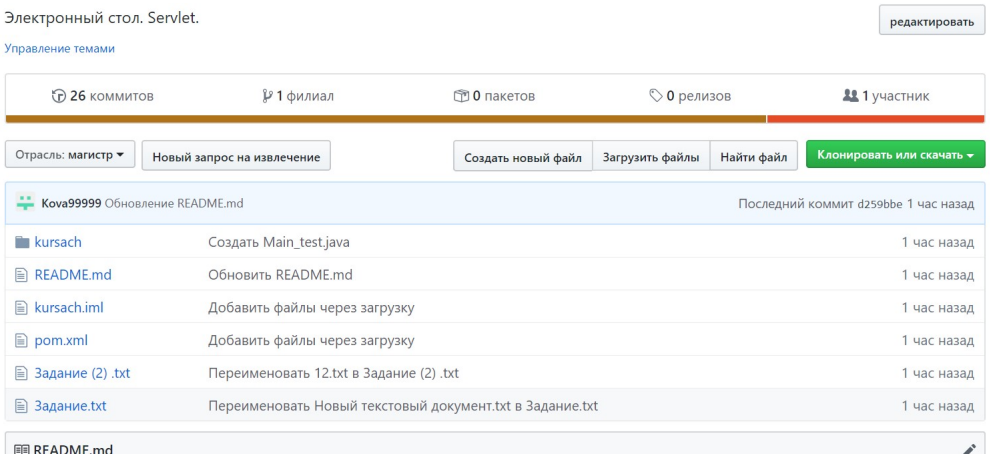

Рис. 18: Проект на GitHub.

6 Заключение

В результате выполнения курсовой работы была реализована электронная ведомость. Для передачи данных между клиентом и сервером используется протокол HTTP. Для реализации web-приложения использован Apache Tomcat. Проект собирается с помощью Maven.Логины и пароли пользователей хранятся в файле. Осуществлено два варианта работы с электронной ведомостью (в качестве ученика и студента), присутствует возможность динамического увеличения числа столбцов и строк,а так же поддержка нескольких ведомостей одновременно.