КОМИТЕТ ОБРАЗОВАНИЯ И НАУКИ КУРСКОЙ ОБЛАСТИ  
областное бюджетное профессиональное образовательное учреждение  
«Курский государственный политехнический колледж»

Курск

2023

Специальность 09.02.07 Информационные системы и программирование

КУРСОВОЙ ПРОЕКТ

**По МДК 05.03 «Тестирование информационных систем»**

**Тема:** **Тестирование информационной системы «Кинотеатр»**

Выполнил:

Студент группы ИС-31 Глушков А.В.

Руководитель:

Красников М.В.

Оценка за курсовой проект

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_ Красников М.В.

(подпись) (инициалы, фамилия)

**Содержание**

|  |  |
| --- | --- |
| Введение | 3 |
| 1 Техническое задание | 5 |
| 1.1 Обоснование требований к комплексу технических средств | 5 |
| 1.2 Описание функциональной структуры | 5 |
| 1.3 Характеристика программных комплексов для тестирования информационной системы | 6 |
| 2 Методы испытаний | 7 |
| 2.1 Интеграционное тестирование | 7 |
| 2.2 Модульное тестирование | 18 |
| 2.3 Нагрузочное тестирование | 19 |
| Заключение | 21 |
| Список литературы | 22 |
| Приложение А Интерфейс среды Visual Studio 2019 | 23 |
| Приложение Б Интерфейс среды Apace JMeter | 24 |
| Приложение В Листинг модульных тестов | 25 |

# ****Введение****

В нашем современном мире ни один человек не может обойтись без сети “интернет”. Он упрощает жизнь многим людям, позволяет совершать покупки, не выходя из дома, пользоваться разного рода услугами. Моей темой для данного проекта послужила “Система учета фильмов”. Данная Информационная Система разрабатывалась мной для помощи кинотеатру, для внесения и отслеживания фильмов, билетов, заказов, кинокомпаний и работников внесённых в базу данных данной компании.

Объектом исследования курсового проекта является тестирование основных функций, имеющихся в программе.

Предметом исследования является методика тестирования систем, подсистем и компонентов программы.

Основной целью курсового проекта является тестирование всех функций программного проекта, с последующем устранением найденных ошибок.

Для достижения данной цели необходимо решить следующие частные задачи:

* разработка структур данных;
* автоматизация процесса работы предприятия;
* выбор эффективных алгоритмов с учётом их точности, устойчивости, сходимости и т.д.;
* интеграция программных модулей, анализ полученных результатов работы модулей;
* нагрузочное тестирование базы данных программы;
* разработка плана тестирования;
* оформление результатов модульных тестов в TestCase-таблицы;
* тестирование всех методов программы;
* устранение ошибок, выявленных в процессе тестирования;
* тестирование программного обеспечения с целью поиска уязвимостей;
* составление отчётов о тестировании.

Источниками данных для тестирования ИС является исходный код программы, базы данных и планы тестирования.

## Техническое задание

## 1.1 Обоснование требований к комплексу технических средств

Для разработки программного обеспечения использовались следующие программно-технические средства:

Персональный компьютер с характеристиками:

* объем оперативной памяти: 16,0 Гб;
* объем накопителя: 1 Тб;
* 6-х ядерный процессор с тактовой частотой 3.60GHz;

Требования к составу и параметрам периферийных устройств: клавиатура, мышь.

Требования к программному обеспечению:

* операционная система Windows 10 Профессиональная;
* средства разработки: Microsoft Windows, Microsoft Visual Studio, Microsoft SQL Server Management Studio, Apache JMeter.
* .NET Framework 6.0;

## 1.2 Описание функциональной структуры

Программное обеспечение предназначено для автоматизации деятельности кинотеатра. Программный продукт позволяет вести автоматизированную обработку данных о фильмах, билетах и товарах.

Функциональные возможности программного обеспечения:

* формирование отчётов;
* добавление данных;
* редактирование данных;
* удаление данных.

## 1.3 Характеристика программных комплексов

Для разработки курсового проекта был выбран язык C# и .Net Framework. Главная причина выбор C# кроется в огромном количестве инструментов и фреймворков, которые поддерживает этот язык.

Из всех сред разработки мой выбор пал на MS Visual Studio. Почти любой разработчик с готовностью признает, что Visual Studio является одной из самых многофункциональных и мощных сред разработки на рынке. Среди её преимуществ особенно важны для меня следующие аспекты:

* подсветка синтаксиса;
* встроенный компилятор и отладчик;

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Из функциональных возможностей Visual Studio был использован набор средств для создания классического Windows приложения – Windows Presentation Foundation, т.к. он отвечает всем заявленным требованиям программного продукта.

Интерфейс Visual Studio 2022 представлен в приложении А.

Apache JMeter - инструмент для проведения нагрузочного тестирования, разрабатываемый Apache Software Foubdation.

– Легко установить;

– GUI и non-GUI режим работы (из консоли);

– Кроссплатформенность;

– Поддержка протоколов (HTTP(S), SOAP, FTP);

– Возможность интеграций с Selenium;

– Возможность автоматически генерировать тестовые скрипты;

Интерфейс Apache JMeter представлен в приложении Б.

**2 Методы испытаний**

2.1 Интеграционное тестирование

Испытаниям подвергается информационная система «Giga Cinema».

Область применения – электронный метод оформления и обработки заявок на мойку. Обозначение программы – «GlushkovCinemaApp.exe».

Цель испытаний – проверка работоспособности автоматизированной информационной системы «Giga Cinema».

Требования к программе – проверка работоспособности приложения при выполнении функций:

* авторизация пользователя;
* добавление данных;
* редактирование данных;
* удаление данных.

Методы и сценарии испытаний представлены в Таблицах 1 – 16

Таблица 1 - Общая информация о тестировании

|  |  |
| --- | --- |
| **Название проекта** | GlushkovCinemaApp |
| **Рабочая версия** | 1.0 v |
| **Имя тестирующего** | Глушков Александр Викторович |
| **Дата(ы) теста** | 05.04.2023 |
|  |  |

Таблица 2 - Test Case #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AuthorizationTest\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование авторизации |
| **Краткое изложение теста** | Вход пользователя в систему |
| **Этапы теста** | Для корректных полей страницы :  • Ввод поля : Логин  • Ввод поля : Пароль |
| **Тестовые данные** | Для полей страницы:  • Логин: admin;  • Пароль: 111; |
| **Ожидаемый результат** | Программа успешно выполнит авторизацию пользователя с надписью “Авторизация успешна” |
| **Фактический результат** | Программа успешно выполнила авторизацию пользователя с надписью “Авторизация успешна” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 3 - Test Case #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_MoviesButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование перехода на страницу фильмов |
| **Краткое изложение теста** | Переход на страницу фильмов |
| **Этапы теста** | 1. Нажать на кнопку “Фильмы” |
| **Ожидаемый результат** | Программа должна перейти на страницу фильмов и успешно вывести все данные |
| **Фактический результат** | Программа перешла на страницу фильмов и успешно вывела все данные |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 4 - Test Case #3:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_OrderButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование перехода на страницу заказов |
| **Краткое изложение теста** | Переход на страницу заказов |
| **Этапы теста** | 1. Нажать на кнопку “Заказы” |
| **Ожидаемый результат** | Программа должна перейти на страницу заказов и успешно вывести все данные |
| **Фактический результат** | Программа перешла на страницу заказов и успешно вывела все данные |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 5 - Test Case #4:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_TicketsButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование перехода на страницу билетов |
| **Краткое изложение теста** | Переход на страницу билетов |
| **Этапы теста** | 1. Нажать на кнопку “Билеты” |
| **Ожидаемый результат** | Программа должна перейти на страницу билетов и успешно вывести все данные |
| **Фактический результат** | Программа перешла на страницу билетов и успешно вывела все данные |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 6 - Test Case #5:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_EmployeesButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование перехода на страницу работников |
| **Краткое изложение теста** | Переход на страницу работников |
| **Этапы теста** | 1. Нажать на кнопку “Работники” |
| **Ожидаемый результат** | Программа должна перейти на работников заказы и успешно вывести все данные |
| **Фактический результат** | Программа перешла на страницу работников и успешно вывела все данные |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 7 - Test Case #6:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_BuffetButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование перехода на страницу буфета |
| **Краткое изложение теста** | Переход на страницу буфета |
| **Этапы теста** | 1. Нажать на кнопку “Буфет” |
| **Ожидаемый результат** | Программа должна перейти на страницу буфета и успешно вывести все данные |
| **Фактический результат** | Программа перешла на страницу буфета и успешно вывела все данные |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 8 - Test Case #7:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_MoviesDeleteButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование удаления фильмов из ItemTemplate |
| **Краткое изложение теста** | Удаление записи Movies |
| **Этапы теста** | 1. Нажать на кнопку “Фильмы”  2. Выбрать запись  3. Нажать на кнопку “Удалить”  4. Подтвердить выполнение удаления |
| **Ожидаемый результат** | Программа должна вывести диалоговое окно с подтверждением удаления, после чего успешно его выполнить, выведя при этом сообщение “Удаление успешно” |
| **Фактический результат** | Программа до вывела диалоговое окно с подтверждением удаления, после чего успешно его выполнила, выведя при этом сообщение “Удаление успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу фильмов; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 9 - Test Case #8:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_OrderDeleteButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование удаления заказов из ItemTemplate |
| **Краткое изложение теста** | Удаление записи Orders |
| **Этапы теста** | 1. Нажать на кнопку “Заказы”  2. Выбрать запись  3. Нажать на кнопку “Удалить”  4. Подтвердить выполнение удаления |
| **Ожидаемый результат** | Программа должна вывести диалоговое окно с подтверждением удаления, после чего успешно его выполнить, выведя при этом сообщение “Удаление успешно” |
| **Фактический результат** | Программа до вывела диалоговое окно с подтверждением удаления, после чего успешно его выполнила, выведя при этом сообщение “Удаление успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу заказов; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 10 - Test Case #9:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_EmployeesDeleteButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование удаления работников из ItemTemplate |
| **Краткое изложение теста** | Удаление записи Employees |
| **Этапы теста** | 1. Нажать на кнопку “Работники”  2. Выбрать запись  3. Нажать на кнопку “Удалить”  4. Подтвердить выполнение удаления |
| **Ожидаемый результат** | Программа должна вывести диалоговое окно с подтверждением удаления, после чего успешно его выполнить, выведя при этом сообщение “Удаление успешно” |
| **Фактический результат** | Программа до вывела диалоговое окно с подтверждением удаления, после чего успешно его выполнила, выведя при этом сообщение “Удаление успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу работников; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 11 - Test Case #10:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_TicketsDeleteButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование удаления билетов из ItemTemplate |
| **Краткое изложение теста** | Удаление записи Tickets |
| **Этапы теста** | 1. Нажать на кнопку “Билеты”  2. Выбрать запись  3. Нажать на кнопку “Удалить”  4. Подтвердить выполнение удаления |
| **Ожидаемый результат** | Программа должна вывести диалоговое окно с подтверждением удаления, после чего успешно его выполнить, выведя при этом сообщение “Удаление успешно” |
| **Фактический результат** | Программа до вывела диалоговое окно с подтверждением удаления, после чего успешно его выполнила, выведя при этом сообщение “Удаление успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу билетов; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 12 - Test Case #11:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_OrderDeleteButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование удаления заказов из ItemTemplate |
| **Краткое изложение теста** | Удаление записи Orders |
| **Этапы теста** | 1. Нажать на кнопку “Заказы”  2. Выбрать запись  3. Нажать на кнопку “Удалить”  4. Подтвердить выполнение удаления |
| **Ожидаемый результат** | Программа должна вывести диалоговое окно с подтверждением удаления, после чего успешно его выполнить, выведя при этом сообщение “Удаление успешно” |
| **Фактический результат** | Программа до вывела диалоговое окно с подтверждением удаления, после чего успешно его выполнила, выведя при этом сообщение “Удаление успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу заказов; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 13 - Test Case #12:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_MoviesEditButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование редактирования данных страницы Movies |
| **Краткое изложение теста** | Редактирование записи Movies |
| **Этапы теста** | 1. Нажать на кнопку “Фильмы”  2. Выбрать запись  3. Нажать на кнопку “Редактировать”  4. Ввести данные, согласно полям  5. Нажать кнопку “Сохранить” |
| **Ожидаемый результат** | Программа должна перейти на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа должна успешно провести редактирование данных и вывести сообщение об успешном сохранении “Сохранено успешно” |
| **Фактический результат** | Программа должна перешла на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа успешно провела редактирование данных и вывела сообщение об успешном сохранении “Сохранено успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу фильмов; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 14 - Test Case #13:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_OrderEditButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование редактирования данных страницы Orders |
| **Краткое изложение теста** | Редактирование записи Orders |
| **Этапы теста** | 1. Нажать на кнопку “Заказы”  2. Выбрать запись  3. Нажать на кнопку “Редактировать”  4. Ввести данные, согласно полям  5. Нажать кнопку “Сохранить” |
| **Ожидаемый результат** | Программа должна перейти на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа должна успешно провести редактирование данных и вывести сообщение об успешном сохранении “Сохранено успешно” |
| **Фактический результат** | Программа должна перешла на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа успешно провела редактирование данных и вывела сообщение об успешном сохранении “Сохранено успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу заказов; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 15 - Test Case #14:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_TicketsEditButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование редактирования данных страницы Tickets |
| **Краткое изложение теста** | Редактирование записи Tickets |
| **Этапы теста** | . Нажать на кнопку “Билеты”  2. Выбрать запись  3. Нажать на кнопку “Редактировать”  4. Ввести данные, согласно полям  5. Нажать кнопку “Сохранить |
| **Ожидаемый результат** | Программа должна перейти на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа должна успешно провести редактирование данных и вывести сообщение об успешном сохранении “Сохранено успешно” |
| **Фактический результат** | Программа должна перешла на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа успешно провела редактирование данных и вывела сообщение об успешном сохранении “Сохранено успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу билетов; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

Таблица 16 - Test Case #15:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_EmployeesEditButtonClicked\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование редактирования данных страницы Employees |
| **Краткое изложение теста** | Редактирование записи Employees |
| **Этапы теста** | . Нажать на кнопку “Работники”  2. Выбрать запись  3. Нажать на кнопку “Редактировать”  4. Ввести данные, согласно полям  5. Нажать кнопку “Сохранить |
| **Ожидаемый результат** | Программа должна перейти на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа должна успешно провести редактирование данных и вывести сообщение об успешном сохранении “Сохранено успешно” |
| **Фактический результат** | Программа должна перешла на страницу редактирования. После заполнения данных и нажатия кнопки “Сохранить”, программа успешно провела редактирование данных и вывела сообщение об успешном сохранении “Сохранено успешно” |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Авторизация;  • Переход на страницу работников; |
| **Постусловие** | Система готова к дальнейшей работе. |
| **Примечания/комментарии** |  |

## 2.2 Модульное тестирование

На данном этапе было произведено модульное тестирование отдельных методов и подсистем программы, с целью проверки полной работоспособности и отсутствия ошибок. Использовались стандартные средства Microsoft Visual Studio 2019 и план тестирования. Результаты тестирования представлены на Рисунке 1.

Листинг кода модульных тестов в Приложении В.

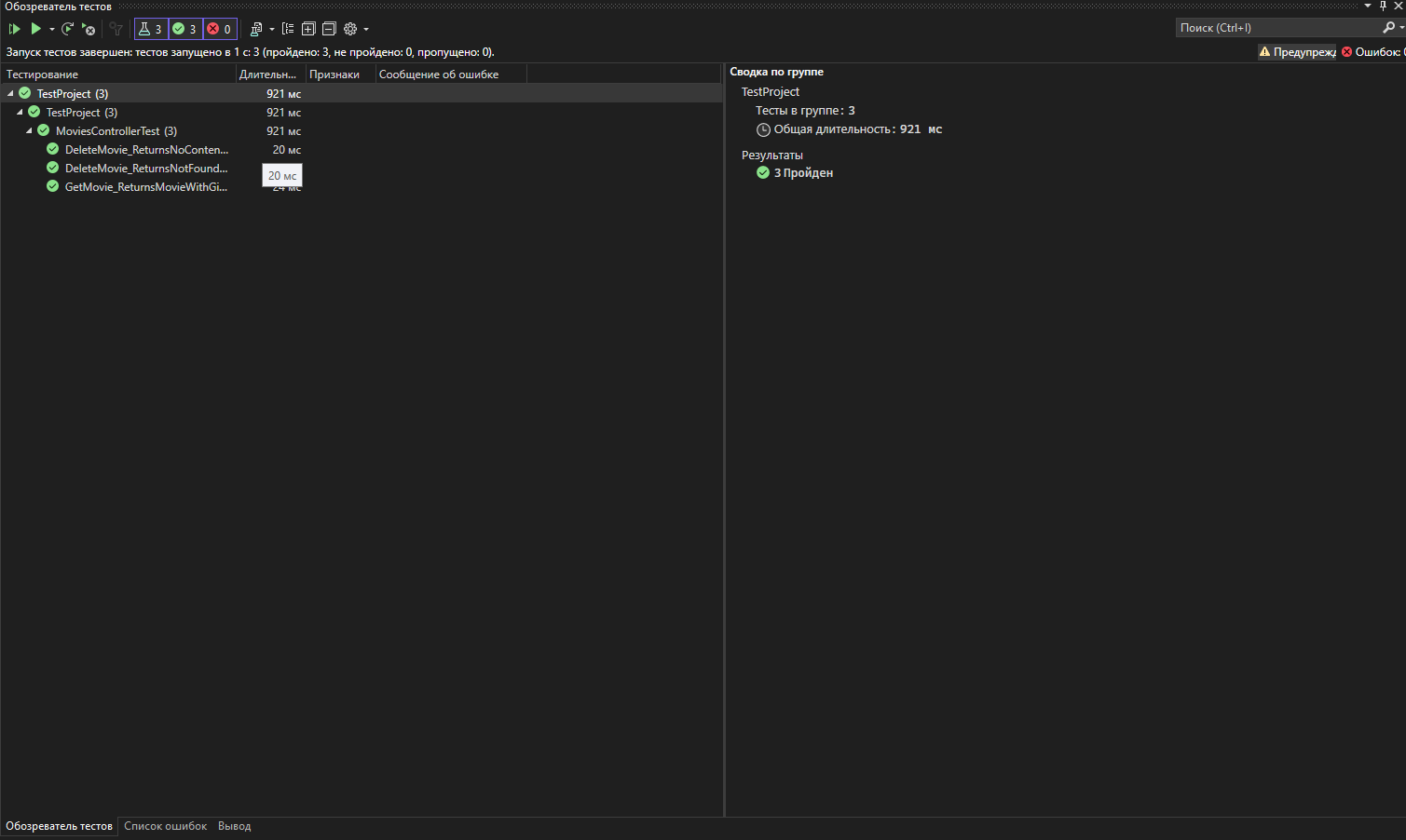


Рисунок 1 - Результаты модульного тестирования Web API

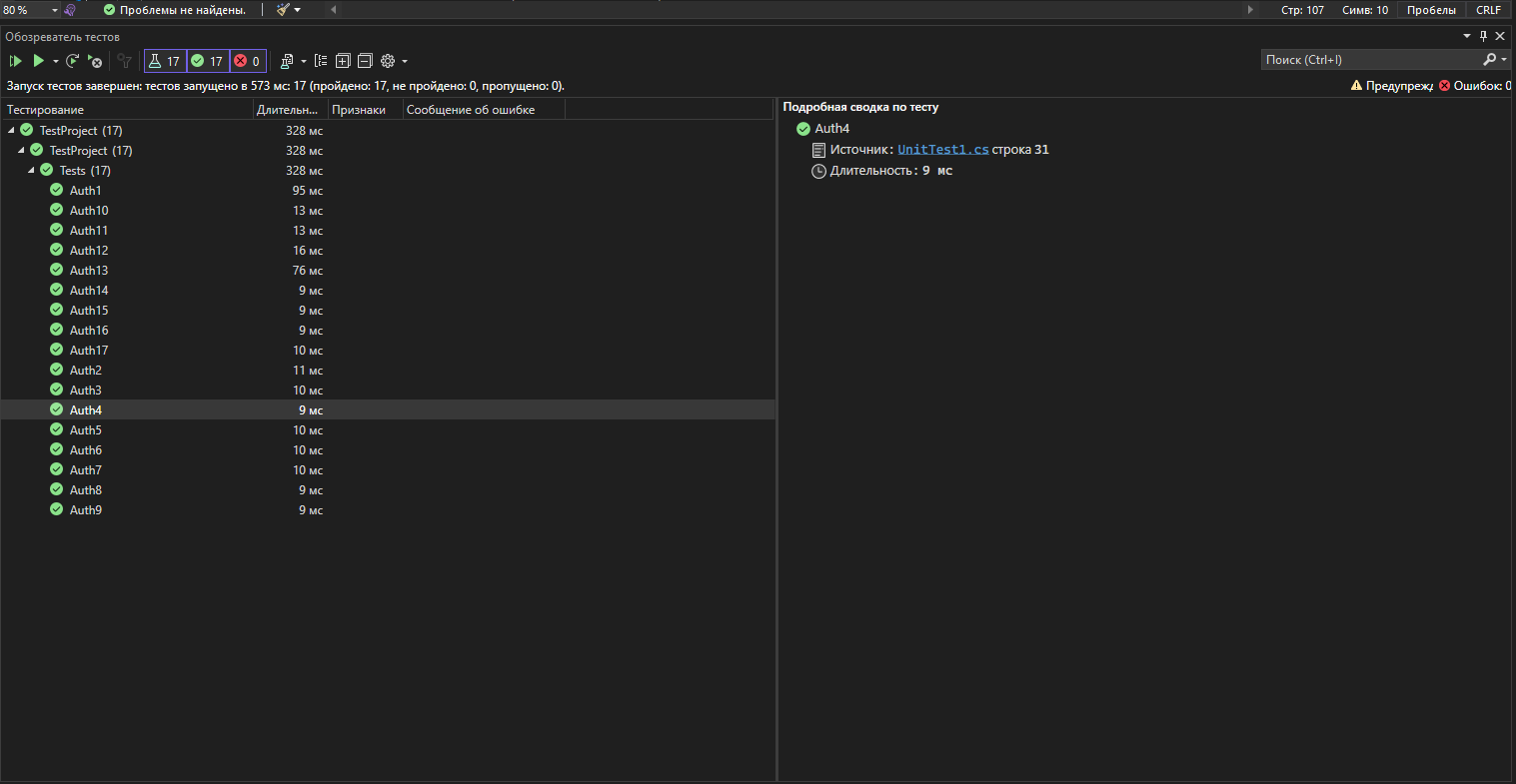
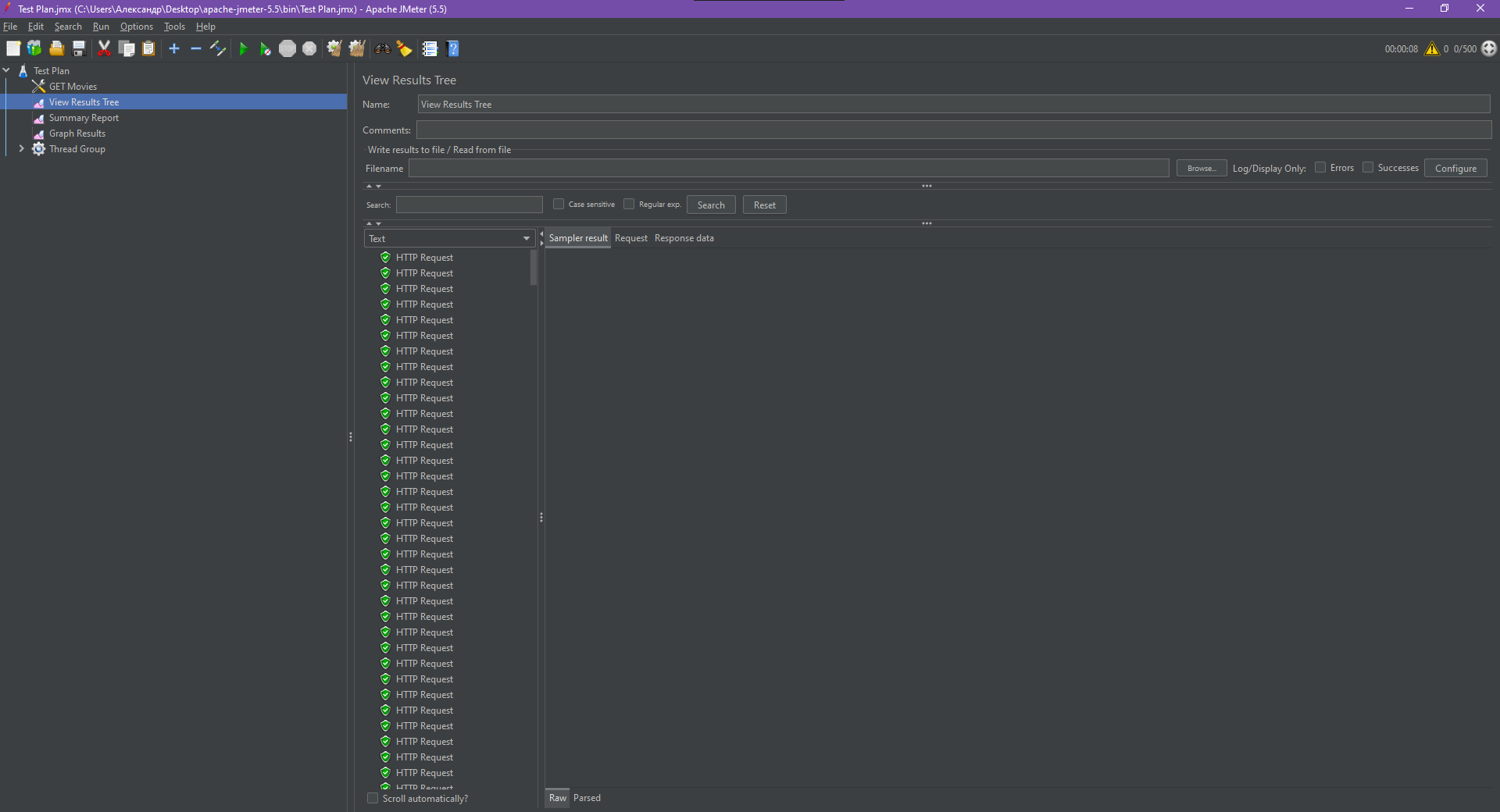


Рисунок 1.2 – Результаты модульного тестирования мобильного приложения

## 2.3 Нагрузочное тестирование

На данном этапе было произведено тестирование программного продукта в тяжёлых условиях, созданных искусственно. Был сымитирован сценарий большого количества одномоментных подключений Web API. В качестве программы для тестирования, использовалась бесплатная программа Apache JMeter.

План тестирования заключался в отправке тысяч запросов Web API, происходивших продолжительное время. Результаты тестирования предоставляются на Рисунках 2-4.

Рисунок 2. View Results Tree

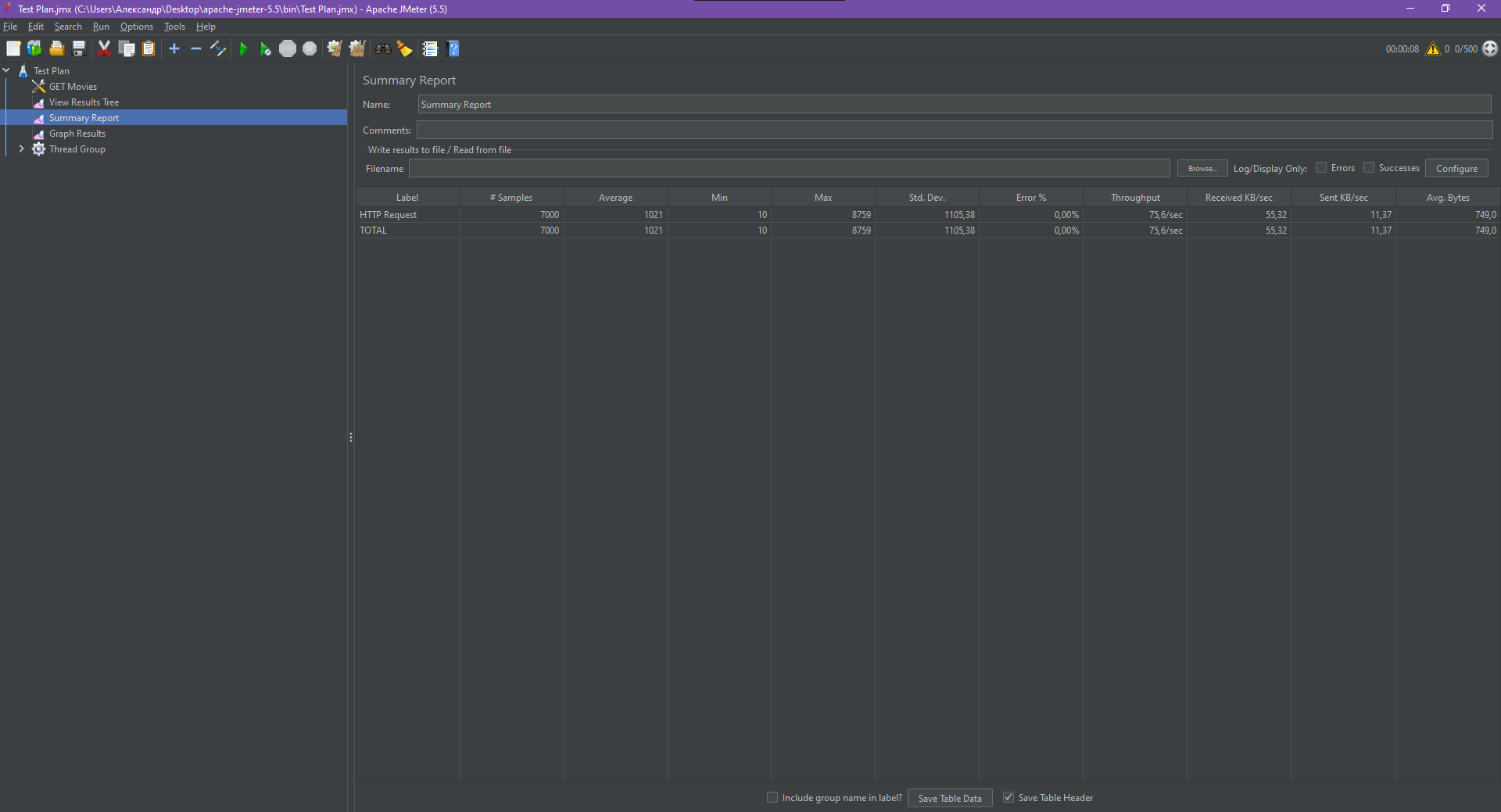


Рисунок 3. Summary Report

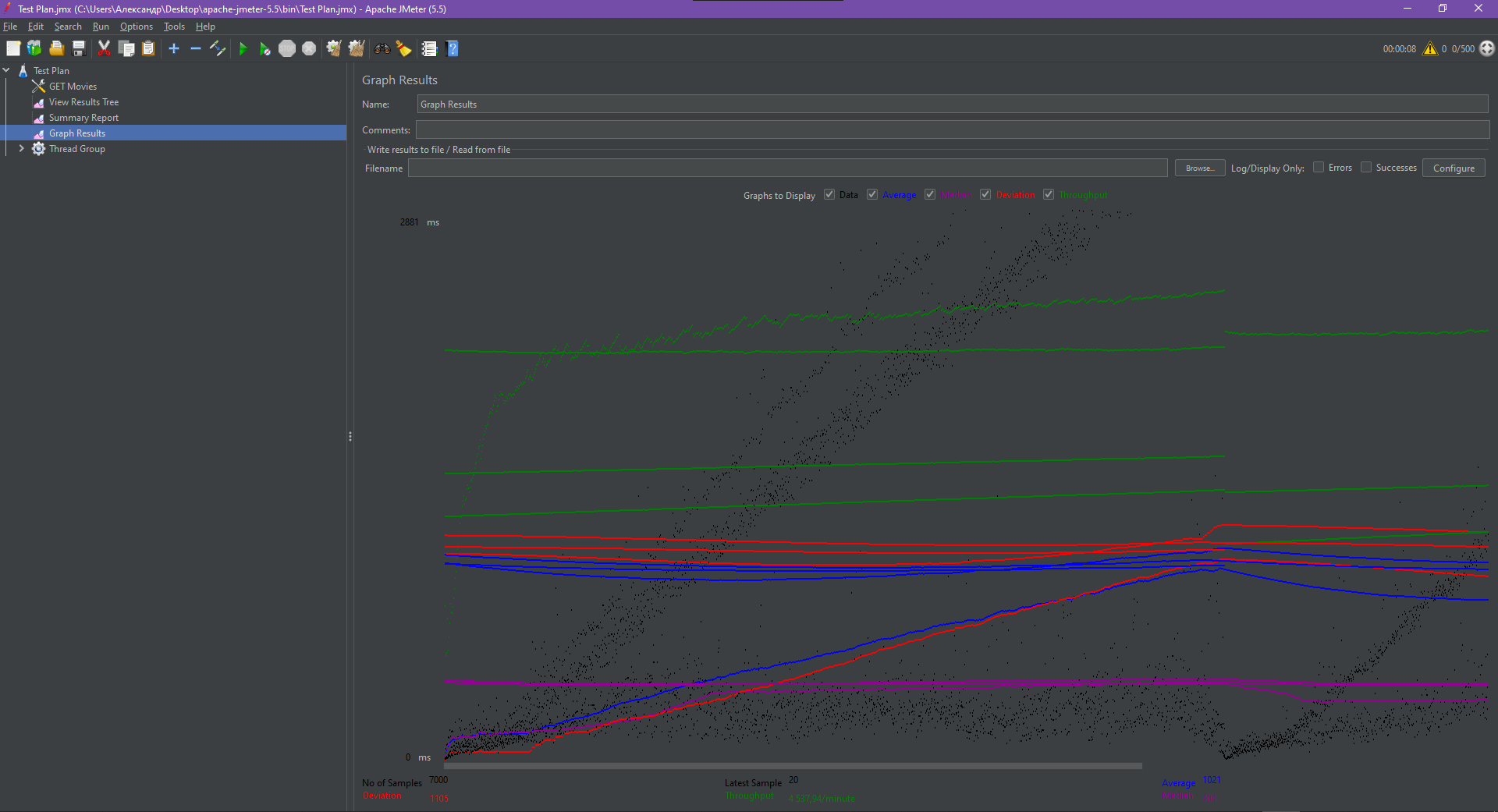


Рисунок 4. Graph Results

## Заключение

В ходе проектирования курсового проекта были решены все поставленные задачи:

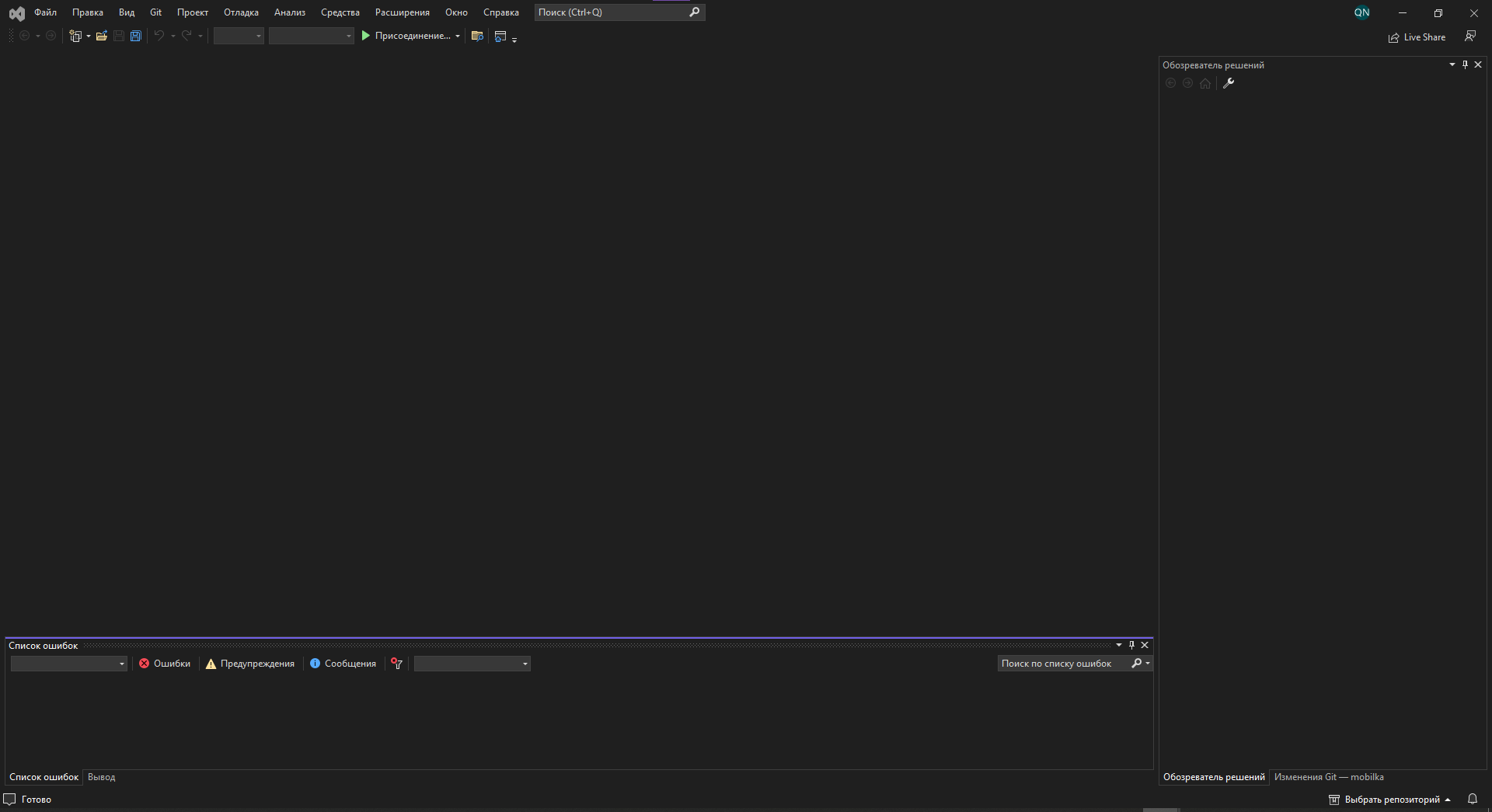
* изучение особенностей конкретной предметной области, относящихся к теме «Тестирование информационной системы «Кинотеатр»»
* анализ возможных подходов и методов решения с обоснованием выбранного метода, выбор или разработка модели (математической, структурной, информационной и т.д.), необходимой для достижения цели;
* выбор эффективных алгоритмов с учётом их точности, устойчивости, сходимости и т.д.;
* разработка структур данных;
* разработка программных модулей (авторизации, поиска и фильтрации данных, добавления, редактирования и удаления данных, формирования отчётной документации и т.д.);
* интеграция программных модулей, анализ полученных результатов работы модулей;
* тестирование программного обеспечения с целью найти уязвимости;
* разработка программной и эксплуатационной документации.

# Список использованной литературы

1. Ганенко А. П. и др. оформление текстовых и графических материалов при подготовке дипломных проектов и письменных экзаменационных работ (требования ЕСКД): Учеб. Пособие для нач. проф. Образования / А. П. Ганенко, Ю. В. Миловская, М. И. Лапсарь. – 2-е изд., стереотип. – М.: ИРПО; Изд. Центр «Академия», 2010. – 352 с.
2. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению
3. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению
4. ГОСТ 19.503-79 Руководство системного программиста. Требования к содержанию и оформлению
5. ГОСТ 19.504-79 Руководство программиста. Требования к содержанию и оформлению
6. ГОСТ 34.601-90 Автоматизированные системы стадии создания
7. ГОСТ 2.105-95 УДК 744:002:006.354 Группа Т52. МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ. Единая система конструкторской документации. ОБЩИЕ ТРЕБОВАНИЯ К ТЕКСТОВЫМ ДОКУМЕНТАМ.
8. ДИАГРАММЫ КОМПОНЕНТОВ https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema15/tema15\_2

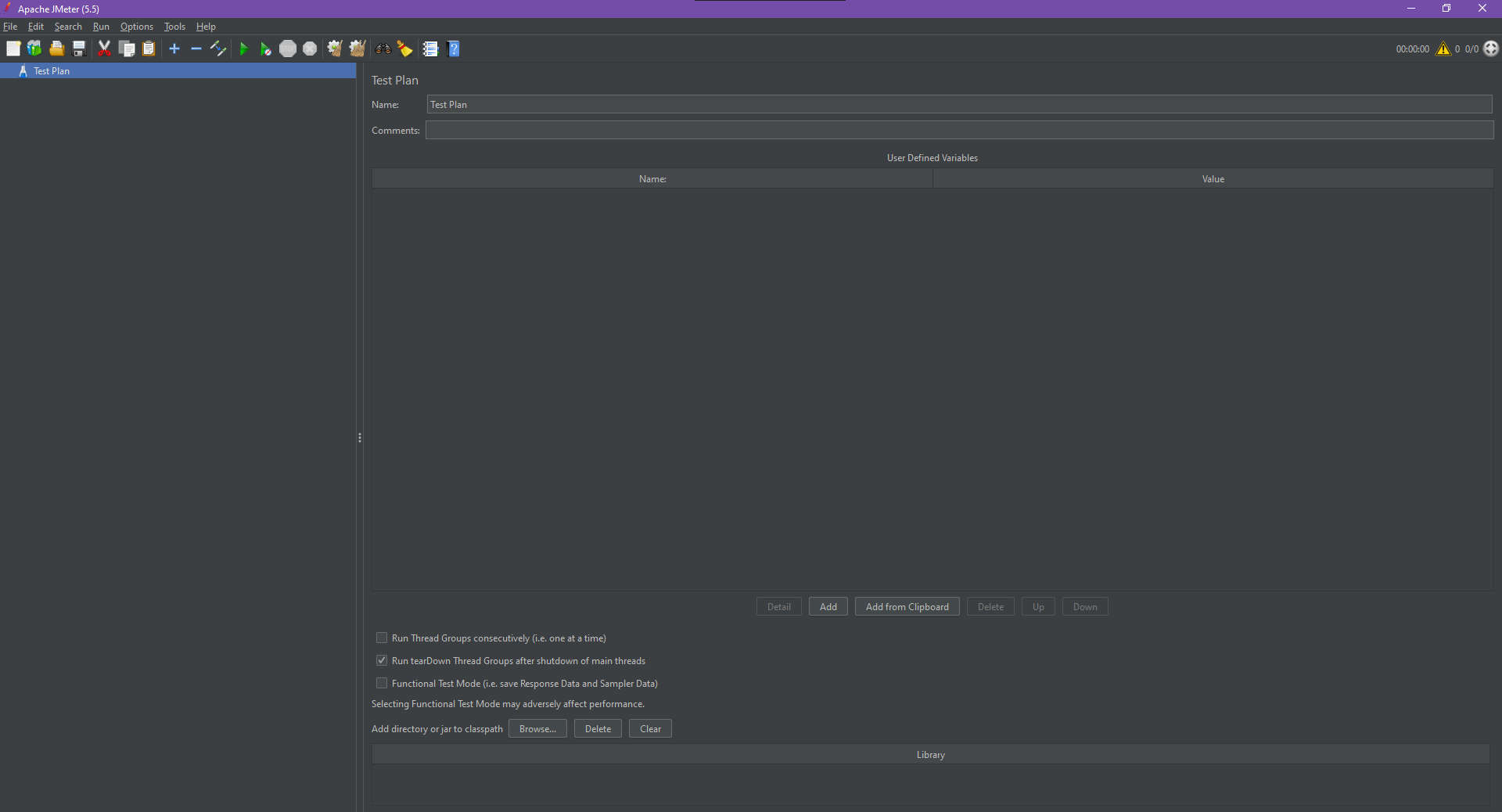
Приложение А

Интерфейс среды Visual Studio 2019



Приложение Б

Интерфейс среды Apache JMeter



Приложение В

# Листинг модульных тестов

Web API

[TestClass]

public class MoviesControllerTest

{

private MoviesController \_controller;

private DbContextOptionsBuilder<CinemaGlushkovContext> \_optionsBuilder;

private CinemaGlushkovContext \_dbContext;

[TestInitialize]

public void Setup()

{

\_optionsBuilder = new DbContextOptionsBuilder<CinemaGlushkovContext>()

.UseInMemoryDatabase(databaseName: "MoviesTestDb");

\_dbContext = new CinemaGlushkovContext(\_optionsBuilder.Options);

\_controller = new MoviesController(\_dbContext);

}

[TestMethod]

public async Task GetMovie\_ReturnsMovieWithGivenId()

{

// Arrange

var movie = new Movie { IdMovie = 6, MovieName = "Movie 6" };

\_dbContext.Movies.Add(movie);

await \_dbContext.SaveChangesAsync();

// Act

var result = await \_controller.GetMovie(movie.IdMovie);

// Assert

Assert.AreEqual(movie.IdMovie, result.Value.IdMovie);

}

[TestMethod]

public async Task DeleteMovie\_ReturnsNoContent\_WhenMovieIsDeleted()

{

// Arrange

var movie = new Movie { IdMovie = 7, MovieName = "Большой куш" };

\_dbContext.Movies.Add(movie);

await \_dbContext.SaveChangesAsync();

// Act

var result = await \_controller.DeleteMovie(movie.IdMovie);

// Assert

Assert.IsInstanceOfType(result, typeof(NoContentResult));

}

[TestMethod]

public async Task DeleteMovie\_ReturnsNotFound\_WhenMovieWithGivenIdDoesNotExist()

{

// Arrange

var movie = new Movie { IdMovie = 1, MovieName = "Большой куш" };

\_dbContext.Movies.Add(movie);

await \_dbContext.SaveChangesAsync();

// Act

var result = await \_controller.DeleteMovie(2);

// Assert

Assert.IsInstanceOfType(result, typeof(NotFoundResult));

}

}

Mobile App

[SetUp]

public void Setup()

{

}

[Test]

public async Task Auth1()

{

var ex = new List<Users> { new Users { password = "11", username = "failure" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth2()

{

var ex = new List<Users> { new Users { password = "111", username = "admin" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(true));

}

[Test]

public async Task Auth3()

{

var ex = new List<Users> { new Users { password = "1", username = "54" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth4()

{

var ex = new List<Users> { new Users { password = "121", username = "ex" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth5()

{

var ex = new List<Users> { new Users { password = "134", username = "fa" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth6()

{

var ex = new List<Users> { new Users { password = "211", username = "ure" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth7()

{

var ex = new List<Users> { new Users { password = "411", username = "lure" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth8()

{

var ex = new List<Users> { new Users { password = "6711", username = "ahah" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth9()

{

var ex = new List<Users> { new Users { password = "234", username = "testing is fun :^)" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth10()

{

var ex = new List<Users> { new Users { password = "11234", username = "e34" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth11()

{

var ex = new List<Users> { new Users { password = "151", username = "211" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth12()

{

var ex = new List<Users> { new Users { password = "1432", username = "f234i" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth13()

{

var ex = new List<Users> { new Users { password = "user1", username = "user1" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(true));

}

[Test]

public async Task Auth14()

{

var ex = new List<Users> { new Users { password = "12341", username = "fa2134" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth15()

{

var ex = new List<Users> { new Users { password = "1ww1", username = "fgfh" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth16()

{

var ex = new List<Users> { new Users { password = "osujdayu", username = "vtornikkk" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}

[Test]

public async Task Auth17()

{

var ex = new List<Users> { new Users { password = "xd", username = "xdddddd" } };

Assert.That(await AuthPage.UserPost(ex[0]), Is.EqualTo(false));

}