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

Специальность 09.02.07 Информационные системы и программирование

КУРСОВОЙ ПРОЕКТ

по ПМ.05 «Проектирование и разработка информационных систем»

Тема: Тестирование информационной системы «Доска объявлений»

Выполнил:

Студент группы ИС-33К Антипова О.А.

Руководитель:

Чаплыгин А.В.

Оценка за курсовой проект

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

\_\_\_\_\_\_\_\_\_ Чаплыгин А.В.

(подпись) (инициалы, фамилия)

Курск

2024

**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 | 8 |
| 2.1 Интеграционное тестирование | 8 |
| 2.2 Модульное тестирование | 18 |
| 2.3 Нагрузочное тестирование | 19 |
| Заключение | 13 |
| Список литературы | 22 |
| Приложение А Интерфейс среды Android Studio | 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 "Доска объявлений”. С развитием технологий всё больше людей предпочитают пользоваться интернетом для поиска информации, услуг и товаров. Поэтому создание приложения, которое предоставляет возможность разместить и найти различные объявления, становится актуальным и востребованным на рынке.

Экономия времени и ресурсов: благодаря быстрому поиску нужной информации на доске объявлений, пользователи экономят время и усилия, необходимые для поиска нужного товара или услуги в других источниках. Это инструмент для эффективного продвижения товаров и услуг, поиска информации и коммуникации с потенциальными клиентами.

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:

* тестирование 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: 500 Гб;
* 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, Android Studio, Apache    JMeter.
* операционная система Windows 10 Pro;
* средства разработки: Microsoft Visual Studio Code, Android Studio.
* Django Rest Framework 3.15.0

## 1.2 Описание функциональной структуры

Разрабатываемый программный продукт предназначен обычных пользователей игры.

Функциональные возможности программного обеспечения:

* Поиск данных;
* Добавление данных;
* Редактирование данных;
* Удаление данных.
* Регистрация и авторизация пользователя

## 1.3 Характеристика программных комплексов

Для тестирования курсового проекта был выбран язык Kotlin. Главная причина выбор Kotlin кроется в огромном количестве программных комплексов, доступных для разработки на Kotlin. Этот язык программирования предлагает широкий спектр инструментов и библиотек, которые могут быть использованы для создания различных типов приложений.

Из всех сред разработки мой выбор пал на Android Studio — это одна из самых популярных сред разработки для создания приложений под Android. Почти любой разработчик с готовностью признает, что Android Studio — это мощный и удобный инструмент для создания качественных приложений.

Преимущества Android Studio:

* + Интеграция с инструментами Google. Android Studio предоставляет разработчикам доступ к инструментам и сервисам Google, таким как Google Play, Firebase и Google Cloud.
* Графическая среда разработки. Android Studio имеет графическую среду разработки, которая упрощает процесс создания приложений.
* Автоматизация. Android Studio предлагает автоматизацию многих задач, таких как сборка, тестирование и развёртывание приложений.
* Эти преимущества делают Android Studio отличным выбором для разработки приложений под Android, в том числе и курсового проекта на Kotlin.

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;

**2 Методы испытаний**

2.1 Интеграционное тестирование

Испытаниям подвергается автоматизированная информационная система «L9.GG»

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

Цель испытаний – проверка работоспособности автоматизированной информационной системы «Projectan».

Требования к программе – проверка работоспособности приложения при выполнении функций:

* авторизация пользователя;
* регистрация пользователя;
* поиск данных;
* фильтрация данных;
* добавление данных;
* редактирование данных;
* удаление данных.

Методы и сценарии испытаний представлены в Таблицах 1 – 16

Таблица 1. Общая информация о тестировании

|  |  |
| --- | --- |
| **Название проекта** | Projectan |
| **Рабочая версия** | 1.1v |
| **Имя тестирующего** | Антипова Ольга Андреевна |
| **Дата(ы) теста** | 05.02.2024 |
|  |  |

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** | 1. Ввод логина 2. Ввод пароля 3. Нажать кнопку «Войти» |
| **Тестовые данные** | Для полей страницы:  • Логин: 1;  • Пароль: 1; |
| **Ожидаемый результат** | Программа отобразит страницу «Меню» с кнопками «Добро пожаловать! Вы успешно вошли в систему!». |
| **Фактический результат** | Программа отобразила страницу «Меню» с кнопками и всплывающим сообщением «Добро пожаловать! Вы успешно вошли в систему!». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»; |
| **Постусловие** | Система готова к дальнейшей работе, с функционалом, доступным пользователю. |
| **Примечания/комментарии** |  |

Таблица 3 - Test Case #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AuthorizationTest\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Тестирование авторизации |
| **Краткое изложение теста** | Вход пользователя в систему |
| **Этапы теста** | 1. Ввод логина 2. Ввод пароля 3. Нажать кнопку «Войти» |
| **Тестовые данные** | Для полей страницы:  • Логин: 12;  • Пароль: 1; |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Ошибка!!! Вы ввели неправильные данные и (или) отсутствует подключение к Интернету». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Ошибка!!! Вы ввели неправильные данные и (или) отсутствует подключение к Интернету». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»; |
| **Постусловие** | Система готова к повторному вводу логина и пароля для дальнейшей работы. |
| **Примечания/комментарии** |  |

Таблица 4 - Test Case #3:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AuthorizationTest\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Тестирование авторизации |
| **Краткое изложение теста** | Вход пользователя в систему |
| **Этапы теста** | 1. Ввод логина 2. Ввод пароля 3. Нажать кнопку «Войти» |
| **Тестовые данные** | Для полей страницы:  • Логин: 1;  • Пароль: 12; |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Ошибка!!! Вы ввели неправильные данные и (или) отсутствует подключение к Интернету». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Ошибка!!! Вы ввели неправильные данные и (или) отсутствует подключение к Интернету». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»; |
| **Постусловие** | Система готова к повторному вводу логина и пароля для дальнейшей работы. |
| **Примечания/комментарии** |  |

Таблица 5 - Test Case #4:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AuthorizationTest\_4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Тестирование авторизации |
| **Краткое изложение теста** | Вход пользователя в систему |
| **Этапы теста** | 1. Ввод пароля 2. Нажать кнопку «Войти» |
| **Тестовые данные** | Для полей страницы:  • Пароль: 1; |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Ошибка! Вы не ввели все данные: Вы не ввели Логин». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Ошибка! Вы не ввели все данные: Вы не ввели Логин». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»; |
| **Постусловие** | Система готова к повторному вводу логина и пароля для дальнейшей работы. |
| **Примечания/комментарии** |  |

Таблица 6 - Test Case #5:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AuthorizationTest\_5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Тестирование авторизации |
| **Краткое изложение теста** | Вход пользователя в систему |
| **Этапы теста** | 1. Ввод логина 2. Нажать кнопку «Войти» |
| **Тестовые данные** | Для полей страницы:  • Логин: 1; |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Ошибка! Вы не ввели все данные: Вы не ввели Пароль». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Ошибка! Вы не ввели все данные: Вы не ввели Пароль». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»; |
| **Постусловие** | Система готова к повторному вводу логина и пароля для дальнейшей работы. |
| **Примечания/комментарии** |  |

Таблица 7 - Test Case #6:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AuthorizationTest\_6 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Тестирование авторизации |
| **Краткое изложение теста** | Вход пользователя в систему |
| **Этапы теста** | 1. Нажать кнопку «Войти» |
| **Тестовые данные** | Для полей страницы: - |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Ошибка! Вы не ввели все данные: Вы не ввели Логин Вы не ввели Пароль». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Ошибка! Вы не ввели все данные: Вы не ввели Логин Вы не ввели Пароль». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»; |
| **Постусловие** | Система готова к повторному вводу логина и пароля для дальнейшей работы. |
| **Примечания/комментарии** |  |

Таблица 8 - Test Case #7:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Edit\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование редактирования сведений о пользователе |
| **Краткое изложение теста** | Редактирование сведений о существующем пользователе |
| **Этапы теста** | 1. Нажать кнопку «Редактировать» 2. Изменить фамилию пользователя 3. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Для полей страницы:  • ФИО:  • Тип объявления:  • Цена:  • Рейтинг:  • Описание. |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Изменения внесены успешно!». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Изменения внесены успешно!». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»;  • Прохождение авторизации;  • Нажатие кнопки «Добавить объявление»; |
| **Постусловие** | Система готова к дальнейшей работе с измененными сведениями о клиенте. |
| **Примечания/комментарии** |  |

Таблица 9 - Test Case #8:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Edit\_2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование редактирования сведений о клиенте |
| **Краткое изложение теста** | Редактирование сведений о существующем клиенте |
| **Этапы теста** | 1. Нажать кнопку «Редактировать» 2. Изменить фамилию пользователя 3. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Для полей страницы:  • ФИО:  • Тип объявления:  • Цена:  • Рейтинг:  • Описание. |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Изменения внесены успешно!». В списке выводимых клиентов имя клиента «Владимир» будет заменено на «Олег». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Изменения внесены успешно!». В списке выводимых клиентов имя клиента «Владимир» было заменено на «Олег». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»;  • Прохождение авторизации. |
| **Постусловие** | Система готова к дальнейшей работе с измененными сведениями о клиенте. |
| **Примечания/комментарии** |  |

Таблица 10 - Test Case #9:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Edit\_3 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование редактирования сведений о пользователе |
| **Краткое изложение теста** | Редактирование сведений о существующем пользователе |
| **Этапы теста** | 1. Нажать кнопку «Редактировать» 2. Изменить данные 3. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Для полей страницы: |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Изменения внесены успешно!». |
| **Фактический результат** | Программа отобразит всплывающее сообщение «Изменения внесены успешно!». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»;  • Прохождение авторизации. |
| **Постусловие** | Система готова к дальнейшей работе с измененными сведениями о пользователе |
| **Примечания/комментарии** |  |

Таблица 11 - Test Case #10:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Users\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование добавления сведений о пользователе |
| **Краткое изложение теста** | Добавление сведений о новом персонаже |
| **Этапы теста** | 1. Нажать кнопку «Редактировать» 2. Изменить данные 3. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Для полей страницы:  • ФИО:  • Тип объявления:  • Цена:  • Рейтинг:  • Описание. |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Добавление прошло успешно!». |
| **Фактический результат** | Программа отобразила всплывающее сообщение «Добавление прошло успешно!» |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»;  • Прохождение авторизации; |
| **Постусловие** | Система готова к дальнейшей работе с добавленными сведениями о пользователе |
| **Примечания/комментарии** |  |

Таблица 12 - Test Case #11:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UsersEdit\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование добавления сведений о пользователе |
| **Краткое изложение теста** | Добавление сведений о новом персонаже |
| **Этапы теста** | 1. Нажать кнопку «Редактировать» 2. Изменить данные 3. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Для полей страницы:  • ФИО:  • Тип объявления:  • Цена:  • Рейтинг:  • Описание. |
| **Ожидаемый результат** | Программа отобразит всплывающее сообщение «Ошибка! Произошла ошибка! Причины могут быть в следующем Не ввели имя! Не ввели Пароль! Не ввели Логин! Не ввели номер телефона». |
| **Фактический результат** | Программа отобразит всплывающее сообщение «Ошибка! Произошла ошибка! Причины могут быть в следующем Не ввели имя! Не ввели Пароль! Не ввели Логин! Не ввели номер телефона». |
| **Статус** | Зачет |
| **Предварительное условие** | • Запуск приложения;  • Нажатие кнопки «Вход»;  • Прохождение авторизации; |
| **Постусловие** | Система готова к дальнейшей работе для корректного добавления сведений о новом |
| **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 Android Studio и план тестирования. Результаты тестирования представлены на Рисунках 1-3.

Рисунок 1

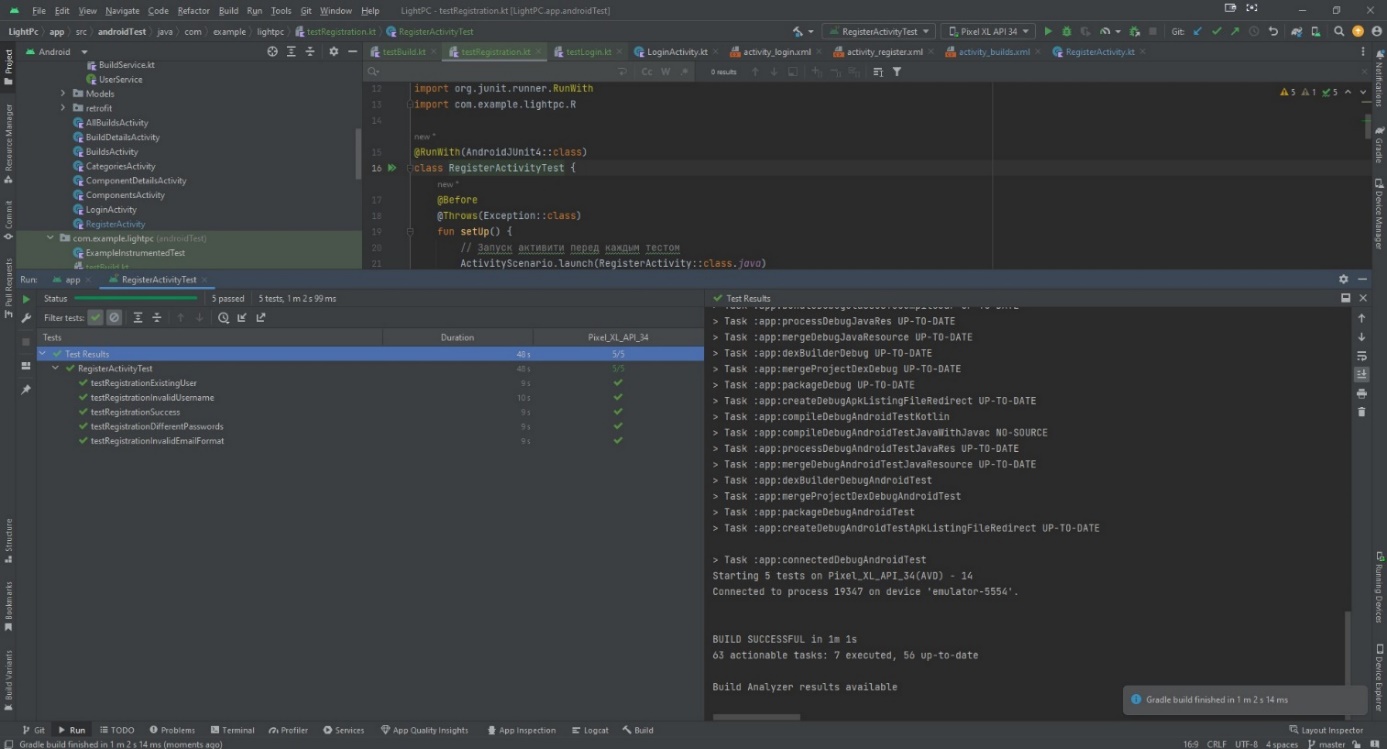


Рисунок 2

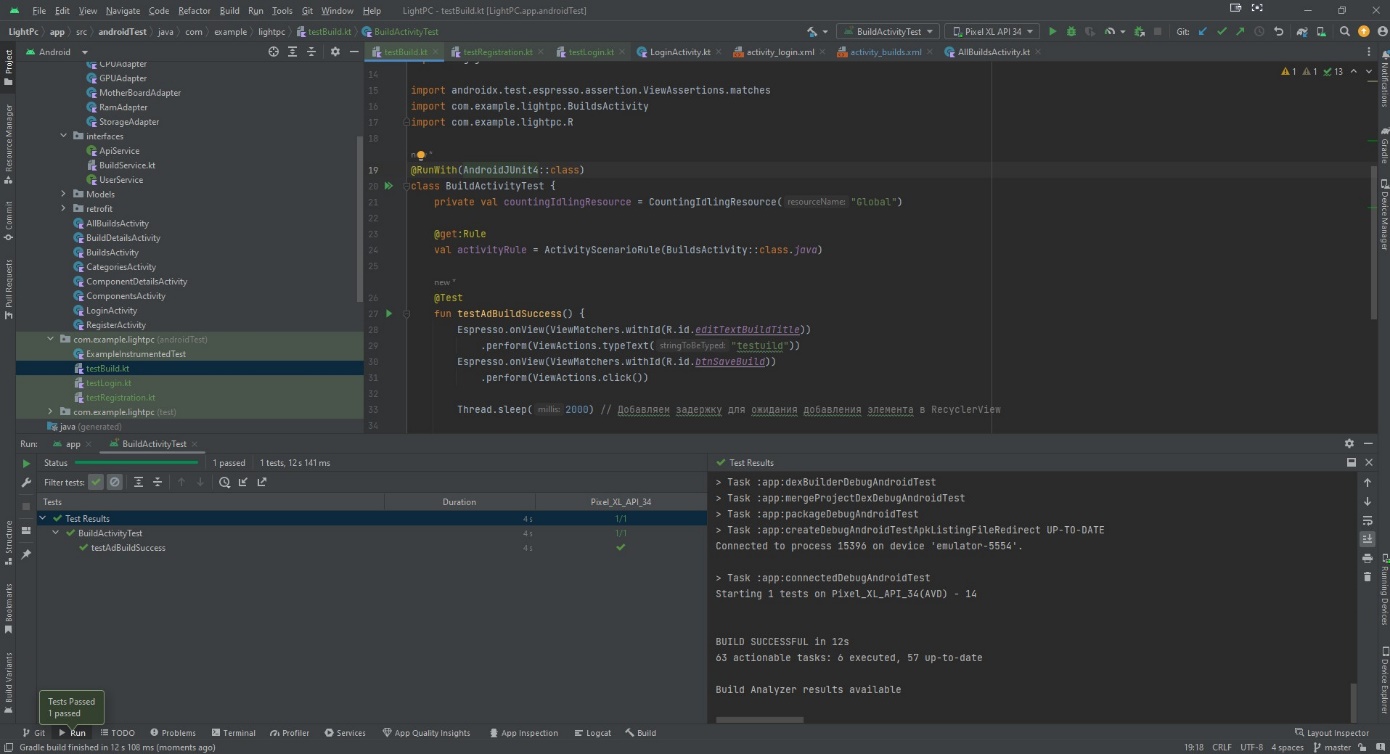
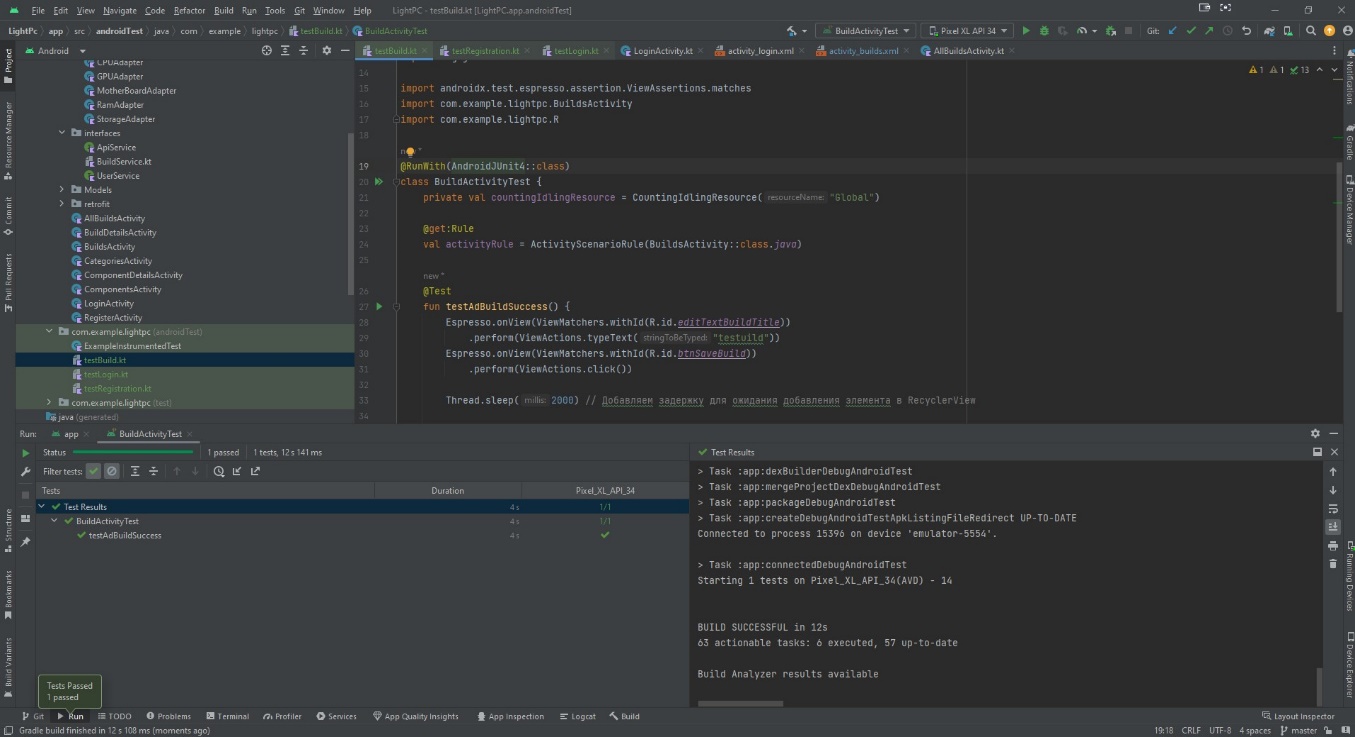


Рисунок 3



## 2.3 Нагрузочное тестирование

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

План тестирования заключался в отправке тысяч запросов апи, происходивших продолжительное время. Результаты тестирования предоставляются на Рисунках 4-6.

Рисунок 4

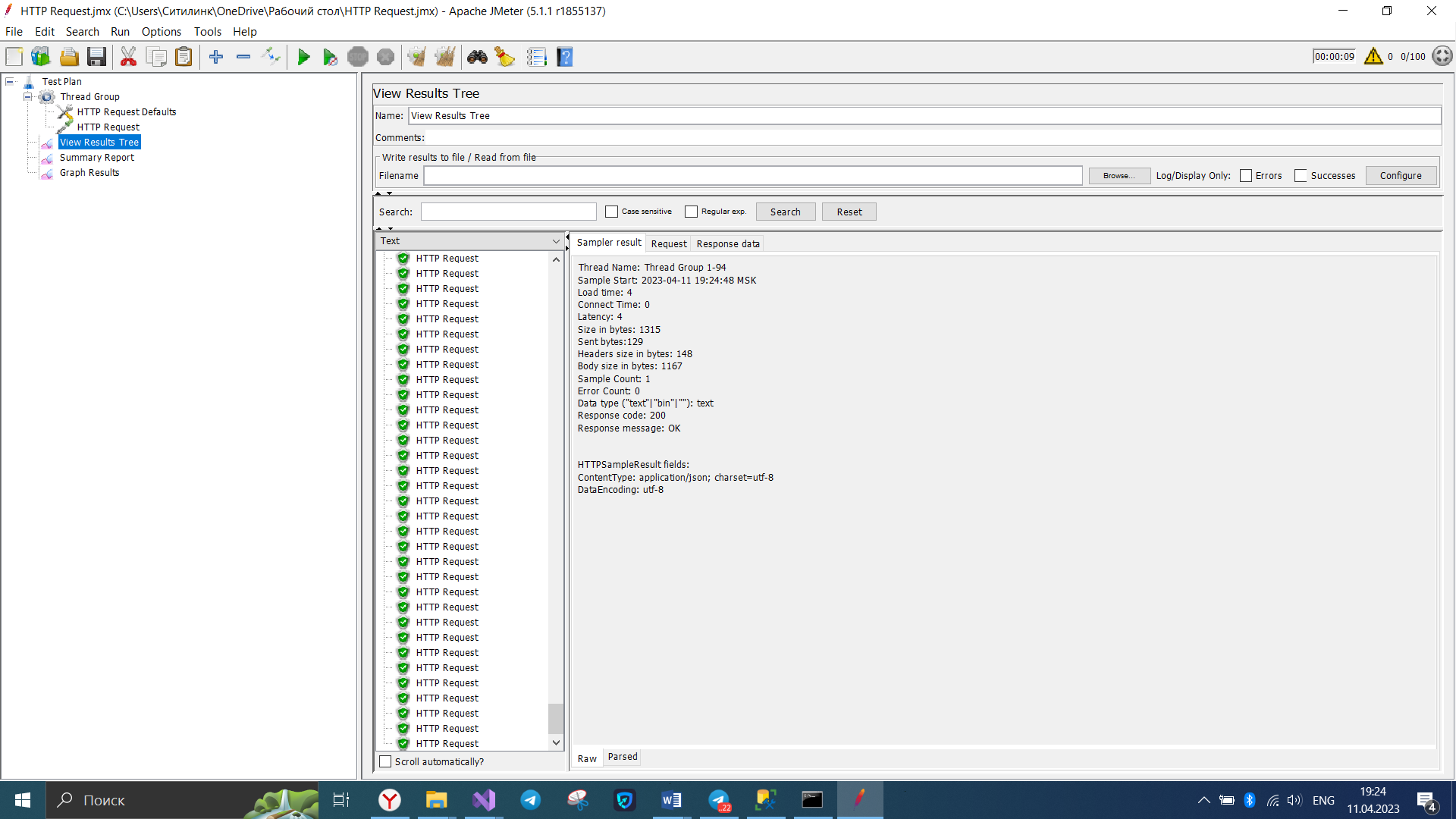


Рисунок 2. View Results Tree

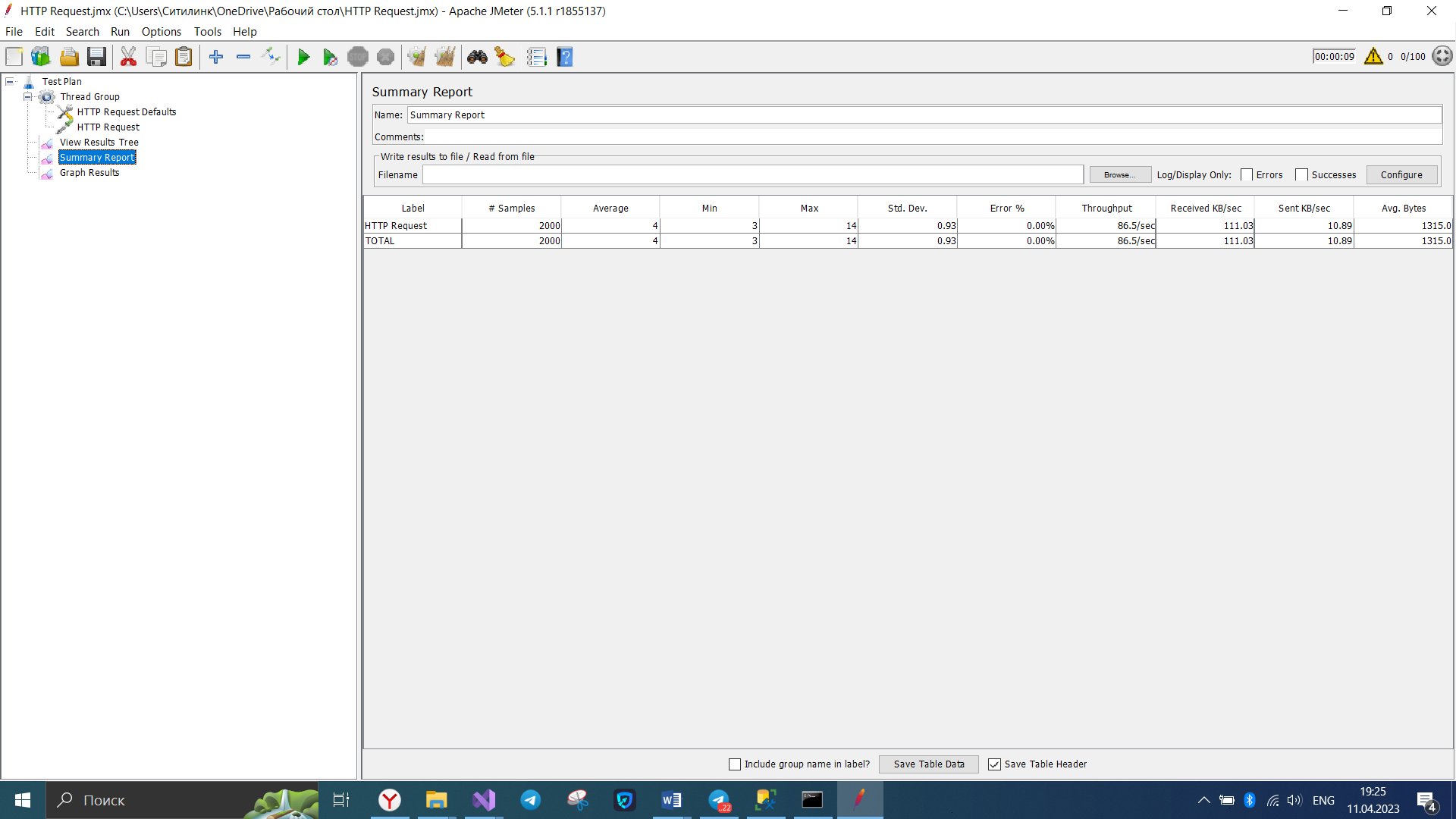
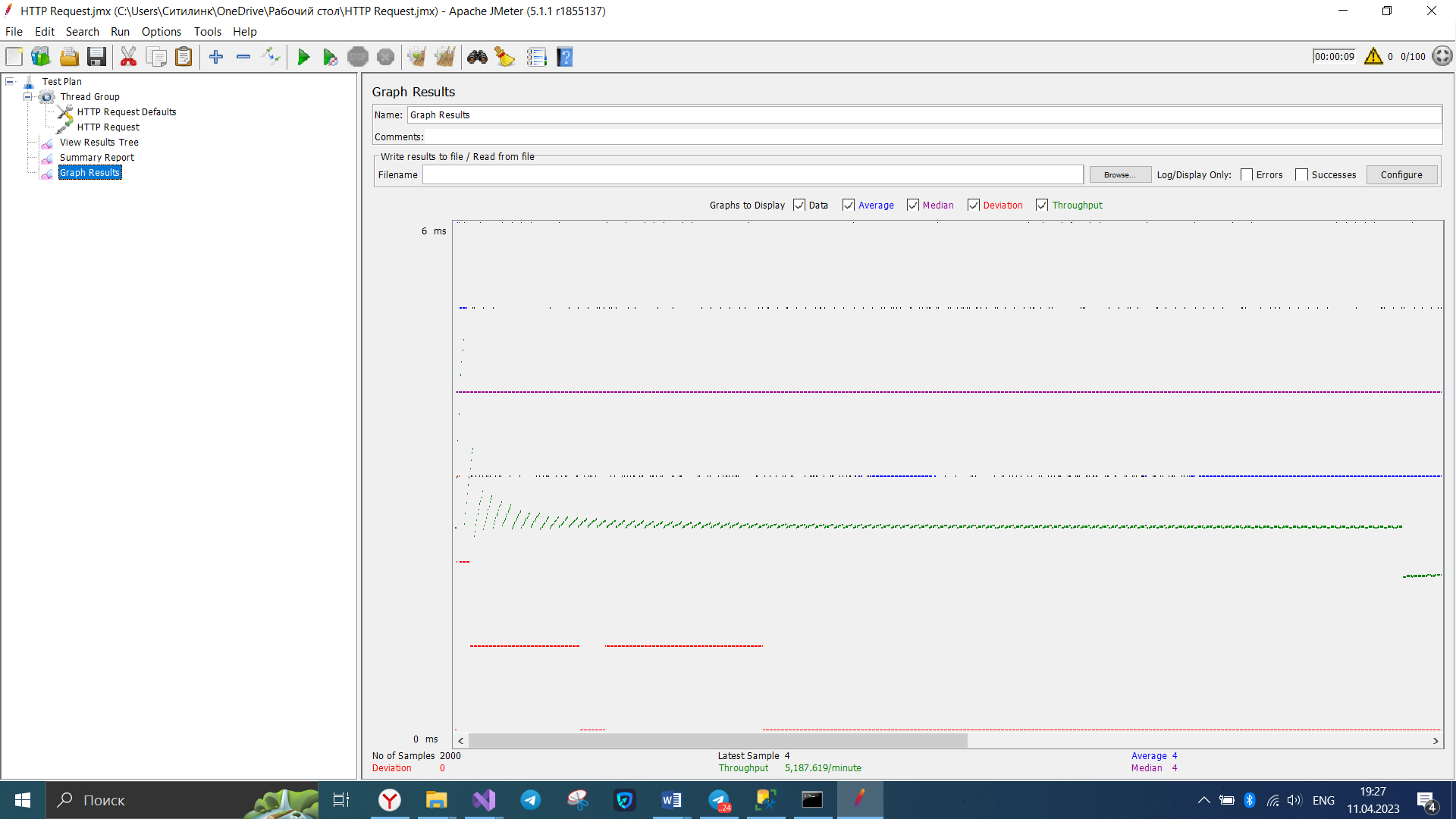


Рисунок 3. Summary Report



## Заключение

В ходе тестирования курсового проекта были решены все поставленные задачи:

* Мы разработали оптимальные структуры данных, обеспечивающие эффективное хранение и обработку информации в программе.
* Мы подобрали оптимальные критерии для выбора алгоритмов, что обеспечило высокую производительность и стабильность нашего программного продукта.
* Процесс интеграции модулей был успешно завершен, что позволило нам получить ценные данные для дальнейшего улучшения программы.
* База данных программы успешно прошла нагрузочное тестирование, что позволило нам оптимизировать ее производительность и надежность.
* Наш план тестирования был разработан и реализован, а результаты тестов были оформлены в удобном для анализа виде.
* Мы тщательно протестировали все методы программы, выявили и устранили ошибки, обеспечив высокое качество и надежность продукта.
* Мы подготовили подробные отчеты о тестировании, содержащие информацию о процессе, результатах и рекомендациях для дальнейших улучшений.

В заключении, выполнение всех этапов работы по курсовой позволило нам создать качественное программное решение, готовое к использованию в реальном бизнесе.

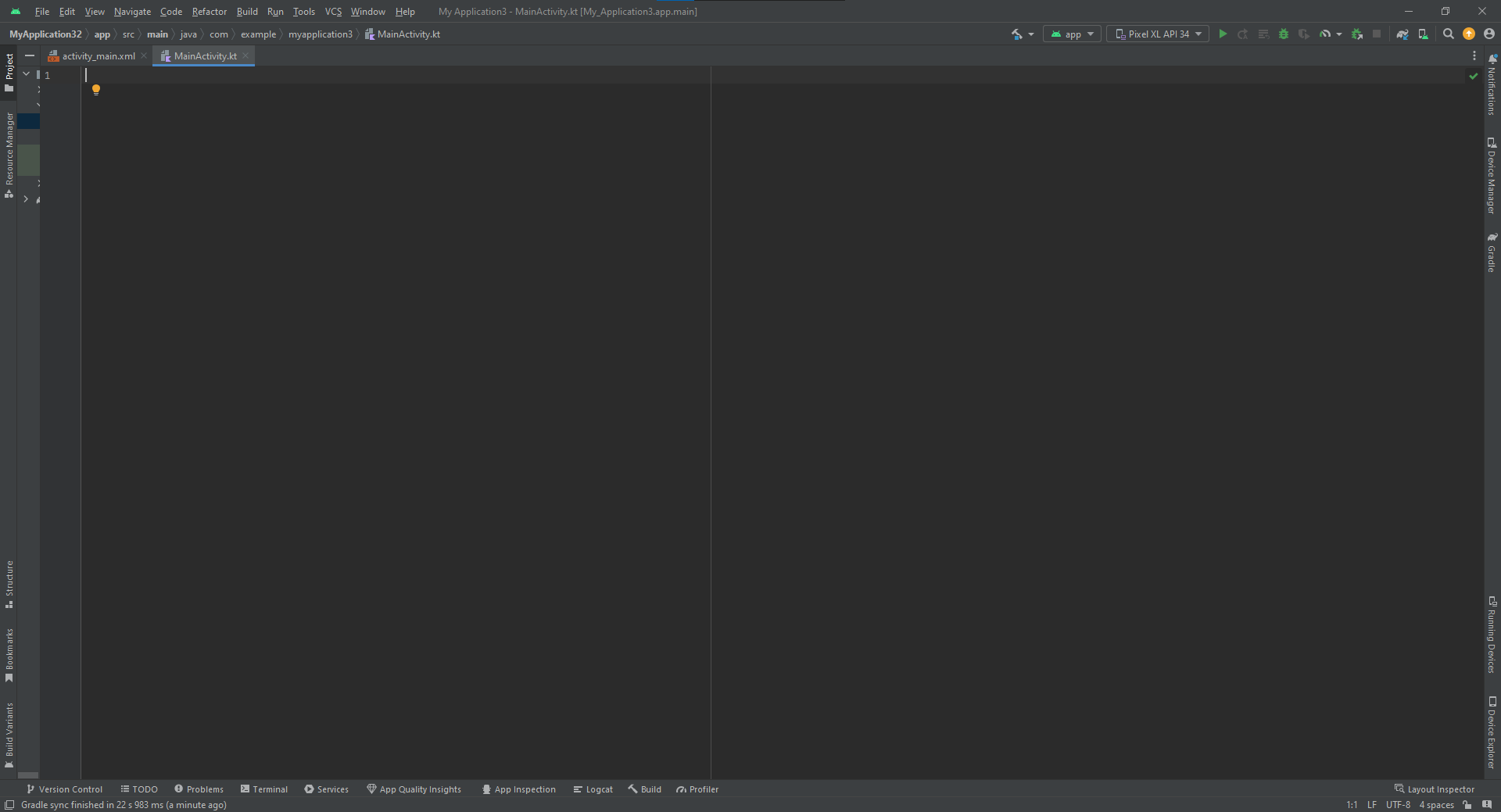
# Список использованной литературы:

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. Пьер-Ив Симон, «Волшебство Kotlin», 2020. - 418 с.
3. Пьер-Оливье Лоранс, Аманда Хинчман-Домингес, Дж.Блейк Мик, Майк Данн, «Программирование на Kotlin для Android», издание - БХВ, 2024. - 645 с.
4. Дон Гриффитс, Дэвид Гриффитс. «Head First. Kotlin», 2019. - 527 с.
5. Ияну Аделекан, «Котлин. Программирование на примерах», 2020. - 479 с.
6. Пол Дейтел, Харви Дейтел, Александер Уолд, «Android для разработчиков», 3-е издание, 2020. - 563с.
7. Зэев Фрайман, «Создание приложений для смартфонов и планшетов под ОС Android», 2023. - 387 с.
8. Денис Колисниченко, «Программирование для Android», 3-е издание, 2022. - 501 с.
9. Ян Дарвин, «Android. Сборник рецептов: задачи и решения для разработчиков приложений», 2022. - 309 с.
10. Джон Хортон, «Разработка Android-приложений с нуля», 3-е издание, 2021. - 576 с.

# 

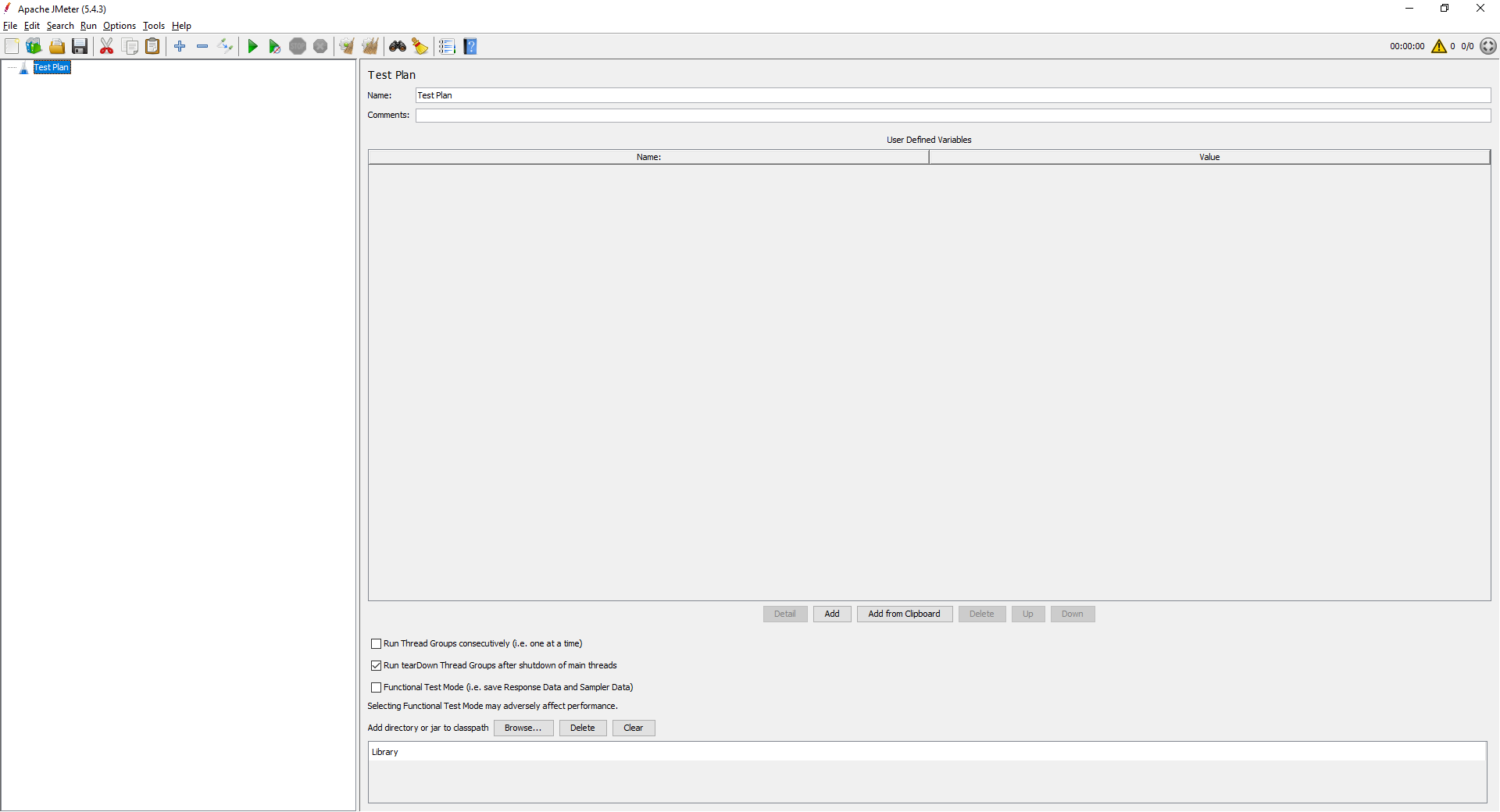
Приложение А

Интерфейс среды Android Studio



Приложение Б

Интерфейс среды Apache JMeter



Приложение В

# Листинг модульных тестов

@RunWith(AndroidJUnit4::class)  
class RegisterActivityTest {  
 @Before  
 @Throws(Exception::class)  
 fun setUp() {  
 // Запуск активити перед каждым тестом  
 ActivityScenario.launch(RegisterActivity::class.*java*)  
 }  
  
 @Test  
 fun testRegistrationSuccess() {  
 Espresso.onView(ViewMatchers.withId(R.id.nickname))  
 .perform(ViewActions.typeText("TestUser"))  
 Espresso.onView(ViewMatchers.withId(R.id.email))  
 .perform(ViewActions.typeText("test1@gmail.com"))  
 Espresso.onView(ViewMatchers.withId(R.id.username))  
 .perform(ViewActions.typeText("Test FullName1"))  
 Espresso.onView(ViewMatchers.withId(R.id.password))  
 .perform(ViewActions.typeText("light2d\_"))  
 Espresso.onView(ViewMatchers.withId(R.id.passwordConfirm))  
 .perform(ViewActions.typeText("light2d\_"), ViewActions.closeSoftKeyboard())  
 Espresso.onView(ViewMatchers.withId(R.id.radioButtonMale)).perform(ViewActions.click());  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister)).perform(ViewActions.click())  
  
 Thread.sleep(2000)  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
  
 @Test  
 fun testRegistrationExistingUser() {  
 Espresso.onView(ViewMatchers.withId(R.id.nickname))  
 .perform(ViewActions.typeText("light2d\_"))  
 Espresso.onView(ViewMatchers.withId(R.id.email))  
 .perform(ViewActions.typeText("test1@gmail.com"))  
 Espresso.onView(ViewMatchers.withId(R.id.username))  
 .perform(ViewActions.typeText("Test FullName1"))  
 Espresso.onView(ViewMatchers.withId(R.id.password))  
 .perform(ViewActions.typeText("light2d\_"))  
 Espresso.onView(ViewMatchers.withId(R.id.passwordConfirm))  
 .perform(ViewActions.typeText("light2d\_"), ViewActions.closeSoftKeyboard())  
 Espresso.onView(ViewMatchers.withId(R.id.radioButtonMale)).perform(ViewActions.click());  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister)).perform(ViewActions.click())  
  
 Thread.sleep(2000)  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
 @Test  
 fun testRegistrationInvalidEmailFormat() {  
 Espresso.onView(ViewMatchers.withId(R.id.nickname))  
 .perform(ViewActions.typeText("TestUser2"))  
 Espresso.onView(ViewMatchers.withId(R.id.email))  
 .perform(ViewActions.typeText("test2.com"))  
 Espresso.onView(ViewMatchers.withId(R.id.username))  
 .perform(ViewActions.typeText("Test FullName1"))  
 Espresso.onView(ViewMatchers.withId(R.id.password))  
 .perform(ViewActions.typeText("light2d\_"))  
 Espresso.onView(ViewMatchers.withId(R.id.passwordConfirm))  
 .perform(ViewActions.typeText("light2d\_"), ViewActions.closeSoftKeyboard())  
 Espresso.onView(ViewMatchers.withId(R.id.radioButtonMale)).perform(ViewActions.click());  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister)).perform(ViewActions.click())  
  
 Thread.sleep(2000)  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
 @Test  
 fun testRegistrationInvalidUsername() {  
 Espresso.onView(ViewMatchers.withId(R.id.nickname))  
 .perform(ViewActions.typeText("TestUser3"))  
 Espresso.onView(ViewMatchers.withId(R.id.email))  
 .perform(ViewActions.typeText("test3@gmail.com"))  
 Espresso.onView(ViewMatchers.withId(R.id.username))  
 .perform(ViewActions.typeText("Test FullName!;&?"))  
 Espresso.onView(ViewMatchers.withId(R.id.password))  
 .perform(ViewActions.typeText("light2d\_"))  
 Espresso.onView(ViewMatchers.withId(R.id.passwordConfirm))  
 .perform(ViewActions.typeText("light2d\_"), ViewActions.closeSoftKeyboard())  
 Espresso.onView(ViewMatchers.withId(R.id.radioButtonMale)).perform(ViewActions.click());  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister)).perform(ViewActions.click())  
  
 Thread.sleep(2000)  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
 @Test  
 fun testRegistrationDifferentPasswords() {  
 Espresso.onView(ViewMatchers.withId(R.id.nickname))  
 .perform(ViewActions.typeText("TestUser4"))  
 Espresso.onView(ViewMatchers.withId(R.id.email))  
 .perform(ViewActions.typeText("test4@gmail.com"))  
 Espresso.onView(ViewMatchers.withId(R.id.username))  
 .perform(ViewActions.typeText("Test FullName1"))  
 Espresso.onView(ViewMatchers.withId(R.id.password))  
 .perform(ViewActions.typeText("light2d\_"))  
 Espresso.onView(ViewMatchers.withId(R.id.passwordConfirm))  
 .perform(ViewActions.typeText("light2d"), ViewActions.closeSoftKeyboard())  
 Espresso.onView(ViewMatchers.withId(R.id.radioButtonMale)).perform(ViewActions.click());  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister)).perform(ViewActions.click())  
  
 Thread.sleep(2000)  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnRegister))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
}

@RunWith(AndroidJUnit4::class)  
class LoginActivityTest {  
 private val countingIdlingResource = CountingIdlingResource("Global")  
  
 @get:Rule  
 val activityRule = ActivityScenarioRule(LoginActivity::class.*java*)  
  
 @Test  
 fun testLoginSuccess() {  
 Espresso.onView(ViewMatchers.withId(R.id.loginUsername))  
 .perform(ViewActions.typeText("light2d\_"))  
 Espresso.onView(ViewMatchers.withId(R.id.loginPassword))  
 .perform(ViewActions.typeText("light2d!"))  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .perform(ViewActions.click())  
  
 IdlingRegistry.getInstance().register(countingIdlingResource)  
  
 Thread.sleep(3000)  
  
 Espresso.onView(ViewMatchers.withId(R.id.loginImage))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 IdlingRegistry.getInstance().unregister(countingIdlingResource)  
 }  
  
 fun testLoginFailureLogin() {  
 Espresso.onView(ViewMatchers.withId(R.id.loginUsername))  
 .perform(ViewActions.typeText("wrong\_user"))  
 Espresso.onView(ViewMatchers.withId(R.id.loginPassword))  
 .perform(ViewActions.typeText("light2d\_"))  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .perform(ViewActions.click())  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
 }  
  
 fun testLoginFailurePassword() {  
 Espresso.onView(ViewMatchers.withId(R.id.loginUsername))  
 .perform(ViewActions.typeText("light"))  
 Espresso.onView(ViewMatchers.withId(R.id.loginPassword))  
 .perform(ViewActions.typeText("wrong\_password"))  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .perform(ViewActions.click())  
  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
  
 @Test  
 fun testLoginFailure() {  
 // Вводим неправильные имя пользователя и пароль  
 Espresso.onView(ViewMatchers.withId(R.id.loginUsername))  
 .perform(ViewActions.typeText("wrong\_user"))  
 Espresso.onView(ViewMatchers.withId(R.id.loginPassword))  
 .perform(ViewActions.typeText("wrong\_password"))  
  
 // Нажимаем на кнопку входа  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .perform(ViewActions.click())  
  
 // Проверяем  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
  
 @Test  
 fun testLoginEmptyLogin() {  
 // Вводим неправильные имя пользователя и пароль  
 Espresso.onView(ViewMatchers.withId(R.id.loginPassword))  
 .perform(ViewActions.typeText("wrong\_password"))  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .perform(ViewActions.click())  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
  
 @Test  
 fun testLoginEmptyPassword() {  
 Espresso.onView(ViewMatchers.withId(R.id.loginUsername))  
 .perform(ViewActions.typeText("light"))  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .perform(ViewActions.click())  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
  
 @Test  
 fun testLoginEmpty() {  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .perform(ViewActions.click())  
  
 Espresso.onView(ViewMatchers.withId(R.id.btnLogin))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 }  
}

@RunWith(AndroidJUnit4::class)  
class BuildActivityTest {  
 private val countingIdlingResource = CountingIdlingResource("Global")  
  
 @get:Rule  
 val activityRule = ActivityScenarioRule(BuildsActivity::class.*java*)  
  
 @Test  
 fun testAdBuildSuccess() {  
 Espresso.onView(ViewMatchers.withId(R.id.editTextBuildTitle))  
 .perform(ViewActions.typeText("testuild"))  
 Espresso.onView(ViewMatchers.withId(R.id.btnSaveBuild))  
 .perform(ViewActions.click())  
  
 Thread.sleep(2000) // Добавляем задержку для ожидания добавления элемента в RecyclerView  
  
 // Проверяем, что элемент с указанным текстом действительно отображается в RecyclerView  
 Espresso.onView(ViewMatchers.withText("testuild")).check(matches(ViewMatchers.isDisplayed()))  
  
 IdlingRegistry.getInstance().register(countingIdlingResource)  
  
 Espresso.onView(ViewMatchers.withId(R.id.buildsLayout))  
 .check(ViewAssertions.matches(ViewMatchers.isDisplayed()))  
  
 IdlingRegistry.getInstance().unregister(countingIdlingResource)  
 }  
}