Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

У Т В Е Р Ж Д А Ю

**Зав. кафедрой ВММБ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Ю. Столбов

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

***ЗАДАНИЕ***

***НА КУРСОВУЮ РАБОТУ***

***по дисциплине***

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Репин Максим Евгеньевич; ИСТ-19-2б

(фамилия, имя, отчество студента; группа)

1. **Тема курсовой работы**

Программа для описания шагов тестирования и фиксирования их выполнения

2. **Срок сдачи студентом отчета:** « 13 » \_апреля\_\_ \_2023\_г.

3. **Содержание отчета:**

Сформулирована основная задача, описаны исходные и конечные данные, описан функционал программы.

Указаны сроки выполнения. Выделены основные этапы реализации программного продукта.

Разработаны логическая и физическая модели БД. Программа реализована в виде конечного продукта.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**КУРСОВАЯ РАБОТА**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Название темы курсовой работы»**

Выполнил:

студент гр.\_\_\_\_\_ИСТ-19-2б\_\_\_\_\_\_\_\_

\_\_\_\_\_Репин\_Максим\_Евгеньевич\_\_\_

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2021**

# Оглавление

[Оглавление 3](#_Toc132583714)

[Обоснование темы курсовой 4](#_Toc132583715)

[Задание 5](#_Toc132583716)

[Постановка задачи 5](#_Toc132583717)

[Этапы и сроки выполнения 5](#_Toc132583718)

[Средства реализации 6](#_Toc132583719)

[Структура базы данных 7](#_Toc132583720)

[Предметная область 7](#_Toc132583721)

[Логическая модель 7](#_Toc132583722)

[Физическая модель 7](#_Toc132583723)

[Программный продукт 9](#_Toc132583724)

[Авторизированный доступ 9](#_Toc132583725)

[Редактирование шагов в тест-кейсах 10](#_Toc132583726)

[Предоставление отчётности 11](#_Toc132583727)

[Виды отчётностей 12](#_Toc132583728)

[Обработка исключительных ситуаций 14](#_Toc132583729)

[Выводы 15](#_Toc132583730)

[Список литературы 16](#_Toc132583731)

# Обоснование темы курсовой

Аргументируем актуальность разрабатываемой программы. Компании, которые разрабатывают собственное программное обеспечение и выпускают его, перед релизом и между обновлениями тестируют разрабатываемое ПО [1]. Кроме этого, компании часто собирают сообщения о багах от пользователей с целью их исправления в следующих обновлениях.

Самый распространённый вид тестирования – это ручное тестирование [2]. Такое тестирование позволяет проверить важные части системы и убедиться, что они работают исправно. Когда тестируемая система становится большой и имеет много компонентов, нуждающихся в тестировании, часть компонентов покрывается автоматизированным тестированием[3].

Автоматизированное тестирование позволяет снять часть нагрузки с ручного тестирования, но никогда не стремится полностью заменить его. Ручное тестирование часто необходимо для тестирования свежих компонентов программы, так как они могут подвергаться изменениям.

Для оптимизации работы тестировщиков существуют программные решения, которые частично автоматизируют часть обязанностей тестировщика [4]. Например, существуют решения, фиксирующие необходимые шаги тестирования и результат тестирования. Именно такая программа разрабатывается в рамках данной курсовой.

Несмотря на то, что автоматизированное тестирование проходит значительно быстрее по сравнению с ручным тестированием, ручное тестирование должно оставаться основным в компании [5]. Не стоит забывать, что автоматизированное тестирование призвано покрывать тестами только те компоненты программы, которые меньше всего подвержены изменениям.

# Задание

В рамках курсовой работы сформулировано и согласовано задание, установлены требования и сроки этапов выполнения работы.

## Постановка задачи

Требуется написать программу, которая описывает шаги для прохождения ручного тестирования и фиксирует результат прохождения тестирования.

Необходимо разделить доступ к системе на пользователей и администраторов. Требуется предоставить администраторам возможность редактировать шаги тестирования. Остальной функционал доступен обеим группам пользователей.

Программа должна предоставлять возможность запускать тест-кейсы, по окончании прохождения шагов тестирования, тест-кейс помечается как успешно пройденный “Passed” или заблокированный “Blocked”. Тест-кейс помечается как “Blocked”, если успешному прохождению тест-кейса мешает, например, баг.

В программе должна присутствовать возможность предоставлять пять видов отчётности:

1. Тест-кейсы, пройденные за текущий run,
2. Процент успешного прохождения тест-кейсов по всем пользователям,
3. Среднее количество тест-кейсов за один run по всем пользователям,
4. Шаги для всех существующих тест-кейсов,
5. Список тест-кейсов, группированных по комплектам.

Каждый из отчётов можно вывести во внешний текстовый файл формата .txt или .csv.

В исключительном случае, программа должна оповестить пользователя о сбое, вывести на экран сообщение об ошибке и попросить исправить ошибку в вводе данных.

## Этапы и сроки выполнения

Плановые сроки приведены в таблице 1.

Таблица 1.

|  |  |  |
| --- | --- | --- |
| Этап | Срок выполнения | Форма отчётности |
| Проектирование хранилища | 26 февраля 2023 | Отчёт |
| Проектирование интерфейса | 12 марта 2023 | Отчёт |
| Разработка диаграммы классов | 26 марта 2023 | Отчёт |
| Программная реализация | 5 апреля 2023 | Отчёт и демонстрация заявленного функционала |
| Тестирование | 13 апреля 2023 | Отчёт |

# 

# Средства реализации

В качестве СУБД выбрана SQLite. SQLite проста в работе, так как у неё имеется всего 5 типов данных для полей и нет ограничений на длину текстовых и числовых полей. Также, SQLite сохраняет базу данных в один файл, а не разворачивает локальный сервер. Это удобно, потому что программу можно передавать между компьютерами, вместе с данными, сохранёнными в БД.

Для отображения интерфейса для пользователя выбран инструмент Windows Forms. Windows Form нативно поддерживается операционной системой Windows, хорошо оптимизирован. Окна Windows Forms адаптируются под версию операционной системы и отображаются в соответствии с настройками пользователя.

Код программы написан на языке C#, так как он поддерживает работу с Windows Forms без дополнительных настроек. Язык C# выбран, так это ООП язык, который удобен для описания классов сущностей, с которыми разработчик будет работать. Преимущество C# заключается в том, что этот язык позволяет быстро и просто написать программу от идеи до конечного продукта, легко связать код программы с интерфейсом Windows Forms и подключиться к БД.

Для изображения логический и физической моделей БД использовался сервис draw.io, так как инструменты сервиса позволяют быстро и качественно изобразить необходимые элементы.

# Структура базы данных

## Предметная область

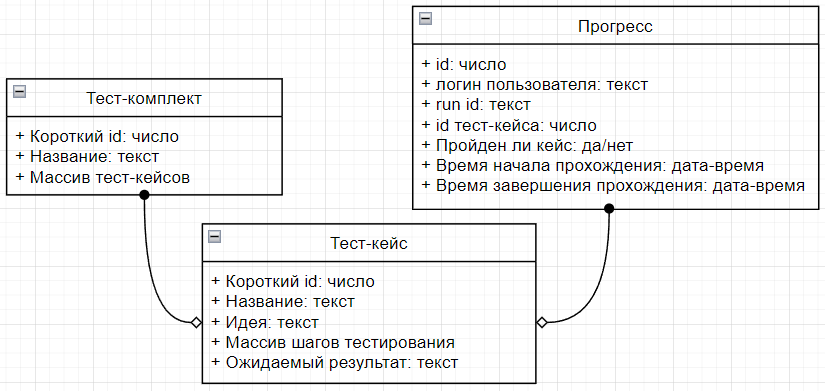
Основным элементом предметной области является тест-кейс. У тест-кейса есть его id, название, описание, ожидаемый результат и массив шагов для прохождения.

Тест-кейсы объединяются в комплекты. Комплект имеет id и название.

Пользователь идентифицируется в системе под login’ом и идентификатором run id. Для фиксации результата работы тестировщика используется его login, run id, id тест-кейса, время начала и завершения выполнения тестирования, результат тестирования “Passed” – пройдено, или “Blocked” – успешному прохождению тест-кейса мешает баг.

## Логическая модель

Изобразим логическую модель на основе предметной области на рисунке 1.

  
Рис. 1 – Логическая модель

Каждая из сущностей имеет id в виде числа.

Сущность Тест-кейс имеет текстовые поля: название, идею (описание), ожидаемый результат. Кроме этого, у сущностей Тест-кейс имеется массив шагов, шаги описаны в виде текста и отсортированы по порядку.

Сущность Тест-комплект имеет текстовое название и массив Тест-кейсов, которые в него входят.

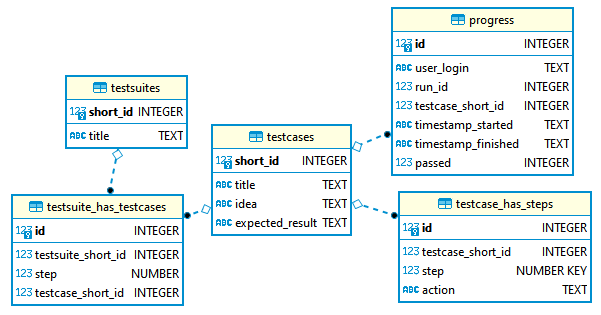
Сущность Прогресс имеет текстовые поля логин пользователя и run id. Сущность имеет id тест-кейса, который был проверен. Сущность имеет поле «Пройден ли тест-кейс» с возможными значениями да/нет. Также, есть два поля типа дата-время – время начала прохождения тестирования и время окончания.

## Физическая модель

На рисунке 2 изображена физическая модель (ER диаграмма) таблиц в базе данных и их связей.

Физическая диаграмма основана на логической диаграмме. Поля, имеющие текстовый тип в SQLite обозначаются типом TEXT с неограниченной длиной. Все Id имеют тип Integer, так как хранят только числа.

SQLite имеет ограниченное количество типов полей: NULL, INTEGER, REAL, TEXT, BLOB. Для текстовых и числовых полей длина не указывается. Учитывая эти ограничения, были выбраны подходящие типы данных для полей сущностей.

  
Рис. 2 – Физическая модель

Для связи между Testsuite и Testcase создана таблица testsuite\_has\_testcase. У каждой записи в таблице есть id, далее хранится ключ testsuite и testcase, а также есть поле типа Number (преобразуется в Integer) для указания порядка тест-кейсов в комплекте.

Для описания шагов тест-кейса была создана отдельная таблица testcase\_has\_steps. Таблица ссылается на id testcase’а, имеет числовое поле step и текстовое поле action.

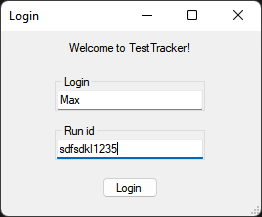
В таблице progress для сохранения результата прохождения тестирования используется поле passed с типом integer, так как в SQLite нет типа Boolean. Запись со значением 1 означает, что тест-кейс пройден, а 0 – не пройден. Для сохранения времени начала и завершения прохождения тест-кейса используется тип TEXT, так как в SQLite нет типа данных date.

# Программный продукт

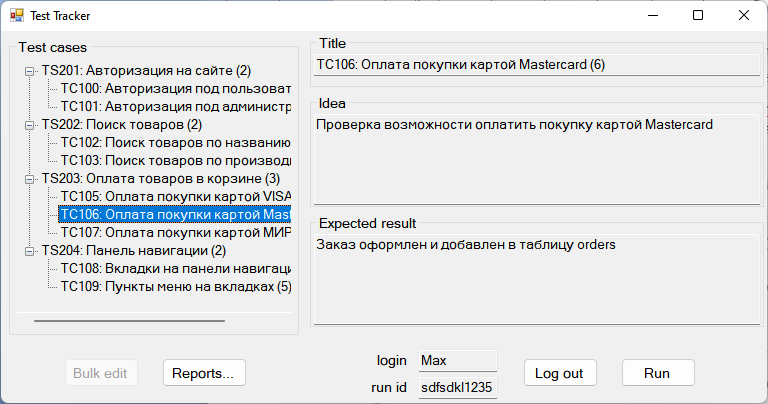
В рамках курсовой работы была написана и протестирована программа, база данных заполнена тестовыми данными.

## Авторизированный доступ

При запуске программы появляется окно авторизации (см. рис 3). Пользователь вводит свой логин, под которым он хочет представиться, и свой run id, в рамках которого он будет проводить тестирование.

  
Рис. 3 – Окно авторизации

Если авторизоваться как обычный пользователь (тестировщик), то в главном окне программы будут доступны все возможности, кроме редактирования шагов в тест-кейсах (см. рис 4).

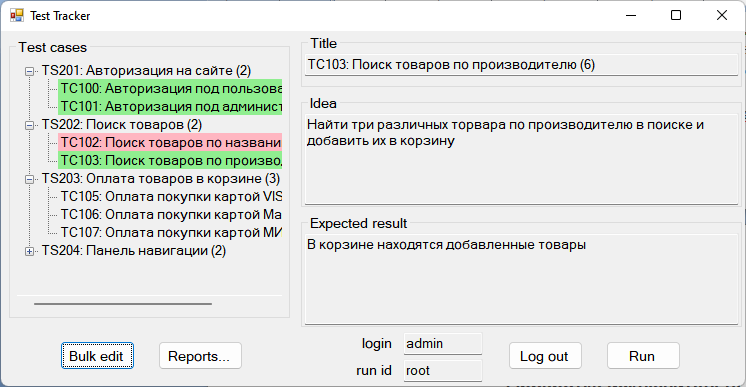
  
Рис 4 – Главное меню для пользователя

Слева на главном меню находится дерево тест-кейсов, объединённых в комплекты. Справа – информация по выделенному в дереве тест-кейсу. Внизу находятся кнопки управления и текущий login и run id, указанные пользователем. Находясь на главном меню программы можно:

* Раскрыть комплекты и выбрать любой из доступных тест-кейсов,
* Запустить любой тест-кейс для прохождения шагов тестирования,
* Вернуться к окну авторизации и авторизоваться под другим пользователем,
* Перейти к окну формирования отчётов,
* Изменить шаги для выбранного тест-кейса.

По мере того, как пользователь проходит тест-кейсы, элементы в дереве на главном меню окрашиваются в цвет, соответствующий результату тестирования: зелёный цвет означает, что в рамках указанного run id, тест-кейс пройден со значением “Passed”, красный – со значением “Blocked”.

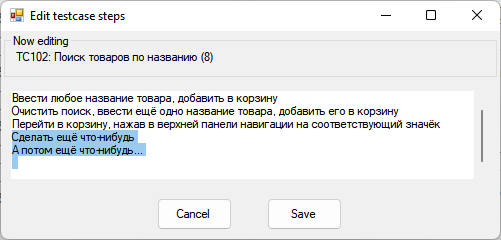
На рисунке 5 изображено окно, когда пользователь прошёл 4 тест-кейса. Так как пользователь имеет роль администратора, кнопка “Bulk edit” стала активной. Только администратор может изменять шаги в тест-кейсах.

  
Рис. 5 – Главное меню для администратора

В Приложении 1 частично описана форма главного окна MainForm, в конструкторе класса которого вызывается объект класса LoginForm, из которого получаем логин и run id пользователя. Метод RefreshBasicInfo() проверяет, авторизовался администратор или обычный пользователь.

## Редактирование шагов в тест-кейсах

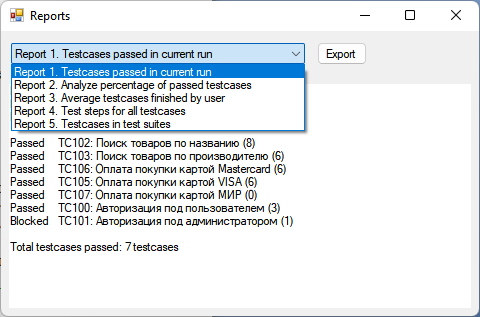
На рисунке 6 изображено окно редактирования шагов тестирования. Только администратор может перейти к этому окну, нажав на кнопку “ Bulk edit” в главном меню.

  
Рис. 6 – Окно редактирования шагов в тест-кейсах

Окно представляет собой текстовое поле, которое можно редактировать. Для сохранения изменений необходимо нажать на кнопку “Save”. В ином случае, изменения текстового поля не будут применены.

## Предоставление отчётности

На рисунке 7 изображено окно формирования отчётов. Для перехода к этому окну необходимо нажать на кнопку “Reports…” на главном меню. Просматривать отчёты могут как пользователи, так и администраторы.

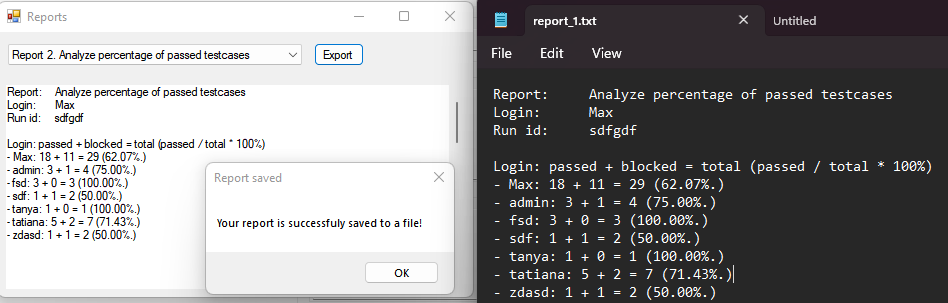
  
Рис. 7 – Окно формирования отчётов

На окне Reports расположен селектор, с помощью которого можно выбрать нужный отчёт, кнопка экспорта отчёта во внешний файл, и текстовое поле, которое заполняется отчётом. При выборе отчёта в селекторе, текстовое поле очищается и заполняется только что выбранным отчётом.

В Приложении 2 приведена часть описания класса Connector, который используется для получения и предоставления данных для БД. Например, метод GetAllTestCases() возвращает массив всех объектов TestCase, а метод GetStepsForTestCase() возвращает массив шагов тестирования.

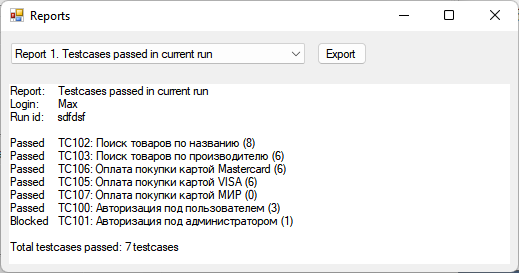
В приложении 3 описана часть класса ReportForm, которая отвечает за окно с выводом отчётов. Например, метод comboBox1\_SelectedValueChanged() вызывается, когда в селекторе меняется значение. В зависимости от выбранного отчёта, вызывается требуемый метод. В Приложении 3 приведены методы, отрисовывающие первый и второй отчёты, а в Приложении 4 – третий, четвёртый и пятый.

Любой из отчётов можно экспортировать во внешний файл, нажав на кнопку “Export”. Перед пользователем появится диалоговое окно выбора места сохранения файла и его названия. После подтверждения выбора, появится файл и пользователь будет оповещён об успешном экспорте (см. рис. 8).

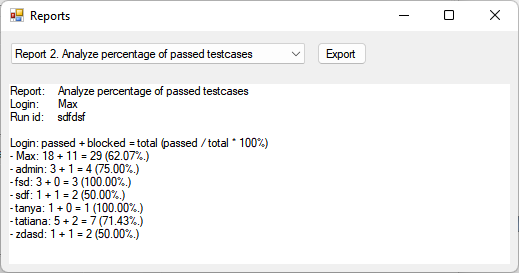
Рис. 8 –Сообщение об успешном экспорте

## Виды отчётностей

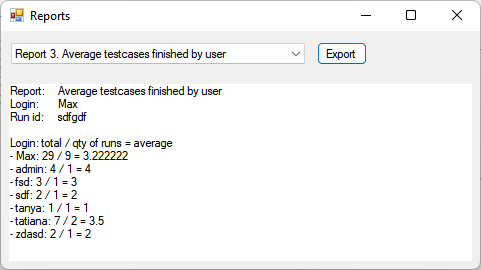
Первый отчёт выводит результат запуска тест-кейсов в текущем run\_id (см. рис. 9).

  
Рис. 9 – Первый отчёт

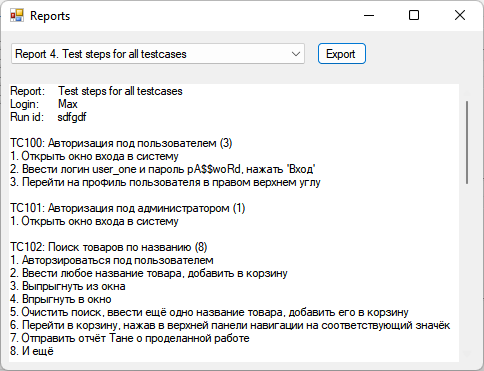
Второй отчёт выводит статистику всех запусков от всех пользователей (см. рис. 10). С помощью этого отчёта можно проанализировать, кто проходит больше тест-кейсов, и кто находит больше багов во время прохождения шагов.

  
Рис. 10 – Второй отчёт

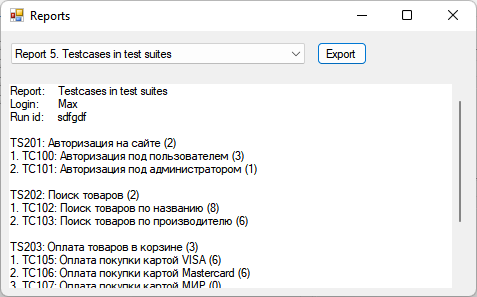
Третий отчёт выводит статистику по общему количеству пройденных тест-кейсов пользователями и количество запусков (см. рис. 11). Таким образом, можно проанализировать, какое количество тест-кейсов проходит каждый пользователь в среднем за один run.

  
Рис. 11 – Третий отчёт

Четвёртый отчёт выводит информацию по каждому тест-кейсу в системе и его шаги тестирования (см. рис. 12).

  
Рис. 12 – Четвёртый отчёт

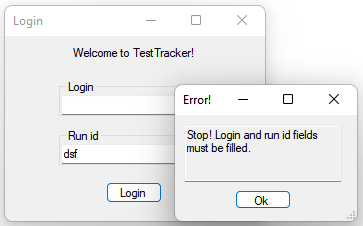
Пятый отчёт выводит информацию по все комплектам и краткую информацию по тест-кейсам в них (см. рис. 13).

  
Рис. 13 – Пятый отчёт

## Обработка исключительных ситуаций

Во время тестирования программы была обнаружена ошибка: программа позволяла авторизоваться, не заполнив поля Login и Run id. Это приведёт к тому, что не будет заполнена информация, кто проходил тестирование, что не информативно.

Ошибка была исправлена. При попытке авторизоваться обработчик проверяет, оба ли поля заполнены. На рисунке 14 изображено окно с просьбой заполнить оба поля на форме авторизации, если хотя бы одно поле оказалось пустым.

  
Рис. 14 – Обработка полей ввода во время авторизации

В Приложении 5 приведено описание класса LoginForm, отвечающего за отрисовку окна авторизации. Метод AllFieldsFilled() проверяет, все ли поля заполнены.

# Выводы

В рамках данной курсовой работы была разработана программа для фиксирования шагов прохождения тестирования и результатов тестирования.

Задача была разбита на этапы, ограниченные временными рамками. Каждый из этапов был выполнен своевременно и в полном объёме.

Все требования к программе соблюдены и реализованы. Создана база данных, наполнена тестовыми данными. Реализован интерфейс и логика взаимодействия элементов системы.

В ходе тестирования программы был найден баг, позволяющий авторизоваться в системе, не указав login и run id пользователя. Данный баг бы устранён. Тестирование всех остальных компонентов программы не выявило ошибок.

В результате получена автономная программа, которая может запускаться на любом компьютере под управлением операционной системы Windows. Работа программы гарантируется на Windows 10 и Windows 11, корректная работа на более ранних версиях не гарантируется.

Язык C# является кроссплатформенным, но так как интерфейс программы разрабатывался с помощью Windows Form, Windows – единственная операционная система, на которой может запуститься программа баз эмуляторов.

Плюсом полученной программы является корректность её работы. В результате получена рабочая программа. Однако её функционал значительно отстаёт от существующих аналогов и не составляет им конкуренции, что является минусом.

# Список литературы

1. Котляров, В. П. Основы тестирования программного обеспечения / В. П. Котляров. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 334 с. – ISBN 5-94774-406-4. – EDN ZGJMOD.
2. Царев, Ю. В. Особенности ручного и автоматизированного тестирования программного обеспечения / Ю. В. Царев, Е. Ф. Сильянова, С. А. Кисельников // Вестник науки. – 2021. – Т. 4, № 6(39). – С. 117-121. – EDN NQXYEF.
3. Денисов, Е. Ю. Автоматизированное тестирование интерактивных программных комплексов / Е. Ю. Денисов, И. А. Калугина // . – 2013. – № 16. – С. 207-210. – EDN RPDEKZ.
4. Барвин, С. К. Автоматизация ручного регрессионного тестирования web-приложений / С. К. Барвин, Д. В. Попов // . – 2019. – № 4(96). – С. 19. – EDN BOSYXJ.
5. Дмитриева, Д. Д. О необходимости освоения технологии ручного тестирования / Д. Д. Дмитриева // Вопросы информационной безопасности в современных условиях : Материалы студенческой научно-практической конференции, Самара, 11 ноября 2022 года. – Самара: Поволжский государственный университет телекоммуникаций и информатики, 2022. – С. 13. – EDN HWUWWG.

### Приложение 1. Класс Main Form

public partial class MainForm : Form

{

private Connector Connector;

int ShortIdOfSelectedTestCase;

string user\_login;

string run\_id;

public MainForm()

{

InitializeComponent();

Connector = new Connector();

using (var loginForm = new LoginForm())

{

var result = loginForm.ShowDialog();

if (result == DialogResult.OK)

{

this.user\_login = loginForm.login;

this.run\_id = loginForm.runId; RefreshBasicInfo();

}

}

}

private void RefreshBasicInfo()

{

if (user\_login == "admin" && run\_id == "root")

{ btnBulkEdit.Enabled = true; }

else { btnBulkEdit.Enabled = false; }

}

private void buttonRun\_Click(object sender, EventArgs e)

{

int testcase\_short\_id = this.ShortIdOfSelectedTestCase;

try

{

TestCase tcase = Connector.GetTestCase(testcase\_short\_id);

TestSteps tsteps = Connector.GetStepsForTestCase(testcase\_short\_id);

using (var form = new TestCaseForm(tcase, tsteps))

{

var result = form.ShowDialog();

if (result == DialogResult.OK)

{

string passed = form.result;

Connector.InsertRunResult(this.user\_login, this.run\_id, tcase.ShortId, "2023-13-13", "2023-13-13", passed);

}

}

}

private void buttonLogOut\_Click(object sender, EventArgs e)

{

using (var loginForm = new LoginForm())

{

var result = loginForm.ShowDialog();

if (result == DialogResult.OK)

{

this.user\_login = loginForm.login;

this.run\_id = loginForm.runId;

}

}

}

}

### Приложение 2. Класс Connector

public class Connector

{

private SQLiteConnection \_connection;

public Connector()

{

var project\_folder = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName;

var pathToSQLiteDatabase = $@"URI=file:{project\_folder}\Database\cases.sqlite";

\_connection = new SQLiteConnection(pathToSQLiteDatabase);

\_connection.Open();

}

public List<TestCase> GetAllTestCases()

{

List<TestCase> cases = new List<TestCase>();

string statement = $"select \* from view\_testcases";

var cmd = new SQLiteCommand(statement, \_connection);

SQLiteDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

TestCase testcase;

testcase.ShortId = reader.GetInt32(0);

testcase.Title = reader.GetString(1);

testcase.Idea = reader.GetString(2);

testcase.ExpectedResult = reader.GetString(3);

testcase.QtyOfSteps = reader.GetInt32(4);

cases.Add(testcase);

}

return cases;

}

public List<TestSuite> GetAllTestSuites()

{

List<TestSuite> suites = new List<TestSuite>();

string statement = $"select \* from view\_testsuites";

var cmd = new SQLiteCommand(statement, \_connection);

SQLiteDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

TestSuite suite;

suite.ShortId = reader.GetInt32(0);

suite.Title = reader.GetString(1);

suite.QtyOfTestCases = reader.GetInt32(2);

suites.Add(suite);

}

return suites;

}

public TestSteps GetStepsForTestCase(int testcase\_short\_id)

{

string statement = $"select step, action from testcase\_has\_steps where testcase\_short\_id={testcase\_short\_id} order by step asc";

var cmd = new SQLiteCommand(statement, \_connection);

SQLiteDataReader reader = cmd.ExecuteReader();

TestSteps steps = new TestSteps();

while (reader.Read())

{

int step = reader.GetInt32(0);

string action = reader.GetString(1);

steps.Add(step, action);

}

return steps;

}

}

### Приложение 3. Формирование первого и второго отчётов

public partial class ReportForm : Form

{

private Connector connector;

private string user\_login; private string run\_id;

private void comboBox1\_SelectedValueChanged(object sender, EventArgs e)

{ switch (comboBox1.SelectedIndex) {

case 0: FormReportOne(); break;

case 1: FormReportTwo(); break;

case 2: FormReportThree(); break;

case 3: FormReportFour(); break;

case 4: FormReportFive(); break; } }

private void FormReportOne()

{

string report = "Report:\tTestcases passed in current run\n";

report += $"Login:\t{this.user\_login}\n";

report += $"Run id:\t{this.run\_id}\n";

report += $"\n";

Dictionary<TestCase, string> passed\_cases = connector.GetResultsOfTestCases(user\_login, run\_id);

foreach (TestCase tcase in passed\_cases.Keys)

{ report += $"{passed\_cases[tcase]}\t{tcase.Representation}\n"; }

report += $"\nTotal testcases passed: {passed\_cases.Count} testcases";

richTextBoxReport.Text = report; }

private void FormReportTwo()

{

Dictionary<string, int> passed = connector.GetPairOfLoginAndExpectedResult("Passed");

Dictionary<string, int> blocked = connector.GetPairOfLoginAndExpectedResult("Blocked");

Dictionary<string, int> total = new Dictionary<string, int>();

foreach (var pair in passed)

{ total[pair.Key] = pair.Value;

if (!blocked.ContainsKey(pair.Key)) { [pair.Key] = 0; }

}

foreach (var pair in blocked)

{

if (!passed.ContainsKey(pair.Key)) { passed[pair.Key] = 0; }

if (total.ContainsKey(pair.Key))

{ total[pair.Key] += pair.Value;

}

else { total[pair.Key] = 0; } }

string report = "Report:\tAnalyze percentage of passed testcases\n";

report += $"Login:\t{this.user\_login}\n";

report += $"Run id:\t{this.run\_id}\n";

report += $"\n";

report += $"Login: passed + blocked = total (passed / total \* 100%)\n";

foreach (var pair in total)

{

string login = pair.Key;

int one = passed[login];

int two = blocked[login];

double percentage = (double)one / (one + two);

report += $"- {login}: {one} + {two} = {one+two} ({String.Format("{0:P2}.", percentage)})\n";

}

richTextBoxReport.Text = report;

}

}

### Приложение 4. Формирование третьего, четвёртого и пятого отчётов

private void FormReportThree()

{

List<(string, int)> finished = connector.GetQtyOfCasesFinished();

Dictionary<string, int> total = new Dictionary<string, int>();

Dictionary<string, int> qty\_of\_runs = new Dictionary<string, int>();

foreach (var pair in finished)

{

if (total.ContainsKey(pair.Item1)) {total[pair.Item1] += pair.Item2;}

else { total[pair.Item1] = pair.Item2; }

if (qty\_of\_runs.ContainsKey(pair.Item1)){qty\_of\_runs[pair.Item1]+=1;}

else { qty\_of\_runs[pair.Item1] = 1; }

}

Dictionary<string, float> average = new Dictionary<string, float>();

foreach (var item in total)

{ average[item.Key] = (float)qty\_of\_runs[item.Key] / total[item.Key]; }

string report = "Report:\tAverage testcases finished by user\n";

report += $"Login:\t{this.user\_login}\n";

report += $"Run id:\t{this.run\_id}\n";

report += $"\n";

report += $"Login: total / qty of runs = average\n";

foreach (var pair in total)

{ string login = pair.Key;

report+=$"- {login}: {total[login]}/{qty\_of\_runs[login]} = {average[login]}\n";

}

richTextBoxReport.Text = report;

}

private void FormReportFour()

{

List<TestCase> testcases = connector.GetAllTestCases();

string report = "Report:\tTest steps for all testcases\n";

report += $"Login:\t{this.user\_login}\n";

report += $"Run id:\t{this.run\_id}\n";

report += $"\n";

foreach (TestCase tcase in testcases)

{

report += $"{tcase.Representation}\n";

TestSteps steps = connector.GetStepsForTestCase(tcase.ShortId);

for (int i = 1; i<=steps.Count; i++)

{ report += $"{i}. {steps[i]}\n"; }

report += "\n";

}

richTextBoxReport.Text = report;

}

private void FormReportFive()

{

List<TestSuite> suites = connector.GetAllTestSuites();

string report = "Report:\tTestcases in test suites\n";

report += $"Login:\t{this.user\_login}\n";

report += $"Run id:\t{this.run\_id}\n";

report += $"\n";

foreach (TestSuite suite in suites)

{

report += $"{suite.Representation}\n";

List<TestCase> cases = connector.GetTestCasesInTestSuite(suite.ShortId);

int count = 1;

foreach (TestCase tcase in cases)

{ report += $"{count}. {tcase.Representation}\n";

count++; }

report += "\n"; }

richTextBoxReport.Text = report;

}

### Приложение 5. Класс Login Form

public partial class LoginForm : Form

{

public string login;

public string runId;

public LoginForm()

{

InitializeComponent();

this.login = "";

this.runId= "";

}

private void buttonLogin\_Click(object sender, EventArgs e)

{

this.login = this.textBoxLogin.Text;

this.runId = this.textBoxRunId.Text;

if (AllFieldsFilled() == true)

{

this.DialogResult = DialogResult.OK;

this.Close();

return;

}

// in case fields are not filled

using (ErrorLoginForm errorForm = new ErrorLoginForm())

{ errorForm.ShowDialog(); }

}

private void textBoxRunId\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{ buttonLogin\_Click(sender, e); }

}

private void textBoxLogin\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{ textBoxRunId.Focus(); }

}

private bool AllFieldsFilled()

{

if (this.login.Length > 0 && this.runId.Length > 0)

{ return true; }

return false;

}

}