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 ПОРТФОЛИО***

***по дисциплине***

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Репин Максим Евгеньевич

(фамилия, имя, отчество студента; группа)

1. **Тема портфолио**

Программа для составления шагов тестирования и фиксирования результатов тестирования

2. **Срок сдачи студентом отчета:** «13 » \_\_апреля\_\_ 2023\_г.

3. **Содержание отчета:**

Техническое задание, хранилище данных, интерфейс, диаграмма классов, реализация, тестирование

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»

**ПОРТФОЛИО**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Название темы портфолио»**

Выполнил:

студент гр.\_\_\_ИСТ-19-2б\_\_\_\_

\_\_\_\_Репин Максим Ревгеньевич\_\_\_\_\_

(Ф.И.О.)

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

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

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

*(должность, ФИО руководителя)*

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

*(оценка) (подпись)*

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

*(дата)*

**Пермь 2021**

Содержание

[***ЗАДАНИЕ*** 1](#_Toc132726009)

[(подпись, дата) (расшифровка) 1](#_Toc132726010)

[Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1](#_Toc132726011)

[Техническое задание 5](#_Toc132726012)

[Общие сведения 5](#_Toc132726013)

[Авторизованный доступ 5](#_Toc132726014)

[Редактирование данных 5](#_Toc132726015)

[Предоставление отчётности 5](#_Toc132726016)

[Реакция на исключительные случаи 6](#_Toc132726017)

[Этапы и сроки выполнения 6](#_Toc132726018)

[Разработка базы данных 7](#_Toc132726019)

[Предметная область 7](#_Toc132726020)

[Логическая модель 7](#_Toc132726021)

[Физическая модель 8](#_Toc132726022)

[Заполнение БД 9](#_Toc132726023)

[Разработка интерфейса 11](#_Toc132726024)

[Интерфейс 11](#_Toc132726025)

[Окно входа в систему 11](#_Toc132726026)

[Окно выбора тест-кейса 11](#_Toc132726027)

[Редактирование тест-кейсов под админом 12](#_Toc132726028)

[Окно прохождения шагов тестирования 13](#_Toc132726029)

[Отчёты 14](#_Toc132726030)

[Реакция на ошибочный ввод данных 16](#_Toc132726031)

[Разработка структуры классов 17](#_Toc132726032)

[Диаграмма классов 17](#_Toc132726033)

[Реализация программы 20](#_Toc132726034)

[Реализация 20](#_Toc132726035)

[Авторизированный доступ 20](#_Toc132726036)

[Редактирование шагов в тест-кейсах 22](#_Toc132726037)

[Предоставление отчётности 23](#_Toc132726038)

[Виды отчётностей 24](#_Toc132726039)

[Обработка исключительных ситуаций 27](#_Toc132726040)

# Техническое задание

## Общие сведения

В данном документе описано техническое задание (далее – ТЗ) к разработке автоматизированной системы (далее – АС) для прохождения ручного тестирования.

Разрабатываемая АС называется TestTracker. Такое название выбрано, так как оно отражает цель АС – отслеживание прогресса прохождения тестирования.

Цель АС TestTracker заключается в хранении и предоставлении информации менеджерам по тестированию о порядке тестирования продукта. Далее, тестировщики проходят необходимые шаги тестирования, прогресс и результат тестирования фиксируется в TestTracker.

Разрабатываемое ПО автоматизирует процесс фиксации и передачи информации между руководящими должностями и тестировщиками. Тестировщики получают набор необходимых шагов тестирования, а руководящие должности получают отчёт о проверенных тест-комплектах.

## Авторизованный доступ

ПО TestTracker разрабатывается для работы двум типам пользователей: руководителей отдела тестирования и тестировщикам.

* Тестировщики проходят описанные тест-кейсы и за ними фиксируется результат прохождения тестирования.
* Руководители могут составлять и редактировать тест-кейсы, а также просматривать отчётность о пройденных тест-кейсах.

## Редактирование данных

Обе роли – руководители и тестировщики, имеют возможность редактировать тестовые случаи и шаги, необходимые для достижения ожидаемого результата.

## Предоставление отчётности

ПО TestTracker предоставляет отчётность пяти видов:

1. Таблица тест-кейсов и их тестовых шагов;
2. Таблица комплектов тестовых случаев;
3. Таблица результатов пройденных тест-кейсов за период времени;
4. Таблица результатов пройденных тестировщиком тест-кейсов;
5. Таблица статистики о качестве тестирования (соотношение Passed / Blocked тест-кейсов) за промежуток времени.

Отчёт в виде таблицы результатов пройденных тестировщиком тест-кейсов предоставляется в видео Excel таблицы.

## Реакция на исключительные случаи

Исключительные случаи возникают, когда пользователь допускает ошибку во вводе данных. Планируется предусмотреть реакцию на исключительные ситуации:

* При достижении лимита на длину текста в тестовом шаге или любом другом текстовом поле появляется сообщение, информирующее об ограничении длины поля.

## 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 |

# Разработка базы данных

## Предметная область

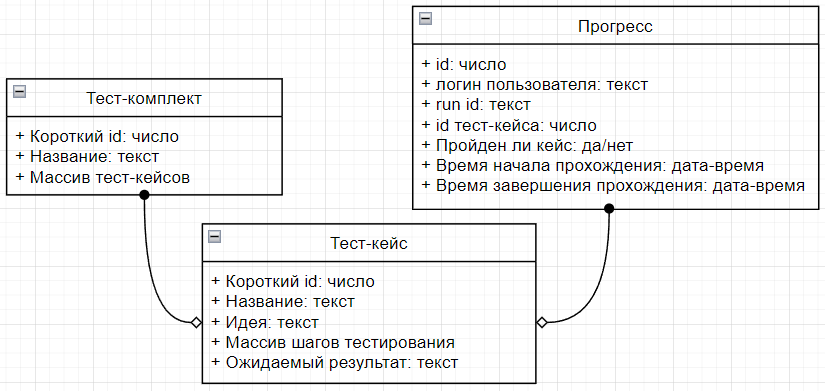
Основным элементом предметной области является тест-кейс. У тест-кейса есть id, название, описание, ожидаемый результат и массив тестовых шагов.

Тест-кейсы объединяются в комплекты. Один тестовый случай может быть включен в несколько комплектов. 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.

## Логическая модель

Изобразим логическую модель на основе описанной предметной области на рисунке 1.

  
Рис. 1 – Логическая модель

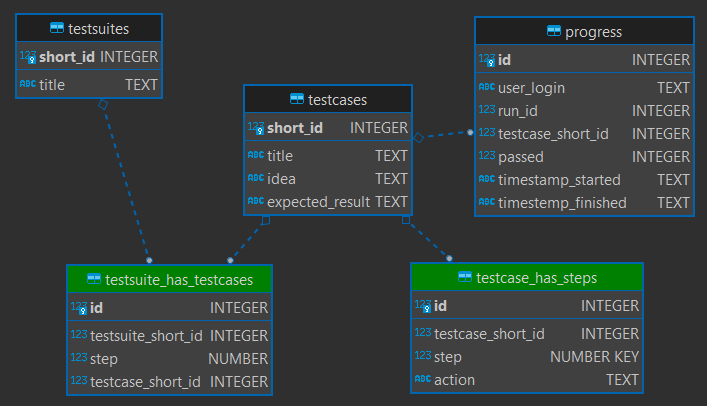
Каждая из сущностей имеет id. Сущность Тест-кейс имеет текстовые поля: название, идею, ожидаемый результат. Кроме этого, у сущности Тест-кейс есть массив шагов, шаги описаны в виде текста и отсортированы по порядку.

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

Сущность Прогресс имеет текстовые поля логин пользователя и id запуска. 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.

  
Рис. 2 – Физическая модель

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.

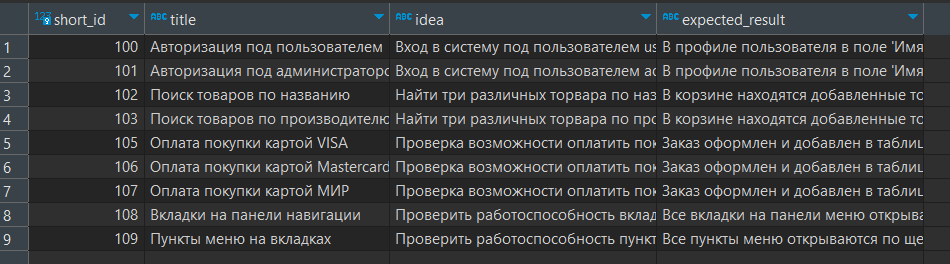
SQLite имеет ограниченное количество типов полей: NULL, INTEGER, REAL, TEXT, BLOB. Для текстовых и числовых полей длина не указывается. Учитывая эти ограничения, были выбраны подходящие типы данных для полей сущностей.

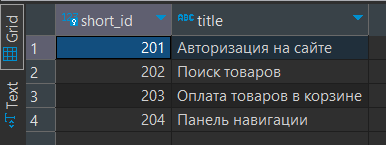
Для связи между Testsuite и Testcase создана таблица testsuite\_has\_testcase. У каждой записи в таблице есть id, далее хранится ключ testsuite и testcase, а также есть поле типа Number (преобразуется в Integer) для указания порядка тест-кейсов в комплекте.

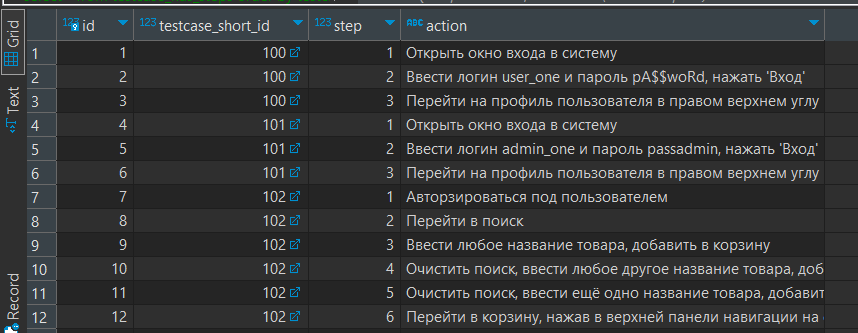
Для описания шагов тест-кейса была создана отдельная таблица testcase\_has\_steps. Таблица ссылается на id testcase’а, имеет числовое поле step и текстовое поле action.

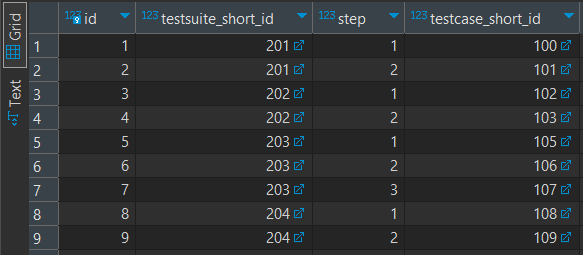
В таблице progress для сохранения результата прохождения тестирования используется поле passed с типом integer, так как в SQLite нет типа Boolean. Запись со значением 1 означает, что тест-кейс пройден, а 0 – не пройден. Для сохранения времени начала и завершения прохождения тест-кейса используется тип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

## Заполнение БД

Заполнение таблицы testcases:  


Заполнение таблицы testsuites:  


Заполнение таблицы testcase\_has\_steps:   


Заполнение таблицы testsuite\_has\_testcase:  


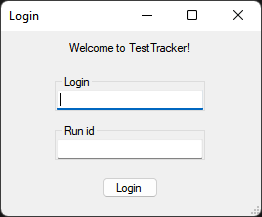
# Разработка интерфейса

## Интерфейс

В соответствии с ТЗ, готовая программа должна запускаться как классическое приложение для Windows. Поэтому, проектирование и разработка интерфейса производится с помощью Windows Forms в Visual Studio.

## Окно входа в систему

При запуске программы пользователя встречает окно входа в систему. Для того чтобы перейти к главному окну со списком тест-кейсов, необходимо указать свой логин – имя пользователя и идентификатор запуска (run id).

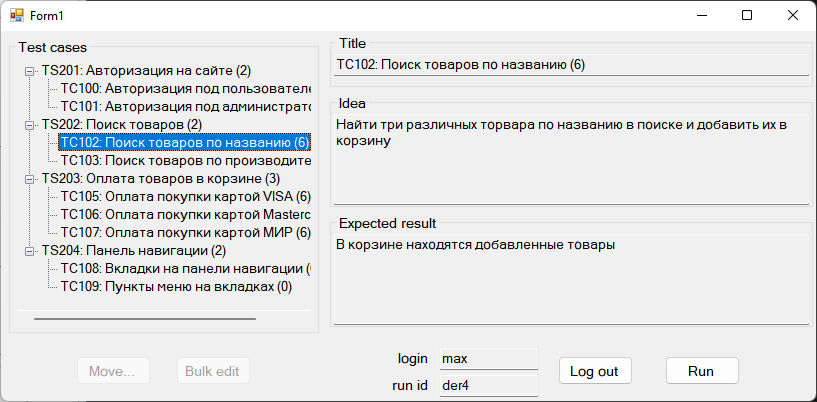
  
Рис. 1 – Окно входа в систему

Login необходим для идентификации пользователя, который проходит шаги тестирования. Run id необходим для того, чтобы группировать вместе различные запуски прохождения тест-кейсов. Таким образом, в отчёте будет отображаться, какие тестовые случаи были проверены в заданный промежуток времени или в рамках указанной задачи.

## Окно выбора тест-кейса

В программе есть две основные роли – администраторы и тестировщики. Авторизовавшись в программе как тестировщик, пользователь имеет возможность запускать тест-кейсы.

На рисунке 2 изображено окно выбора тест-кейса для прохождения тестирования. В левой части отображается дерево тестовых случаев, объединённых в комплекты. В начале названия комплекта отображается полный id объекта, а в конце – количество тестовых случаев в комплекте. Аналогично для тест-кейсов, однако в скобочках отображается количество шагов.

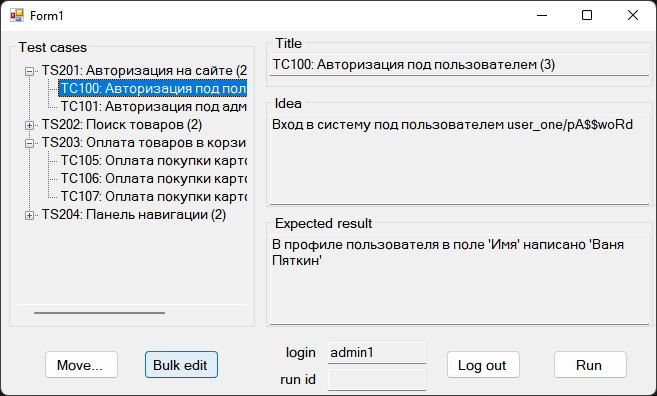
 Рис. 2 – Окно выбора тест-кейса

В правой части окна отображается полная информация о выбранном тестовом случае: название, описание и ожидаемый результат.

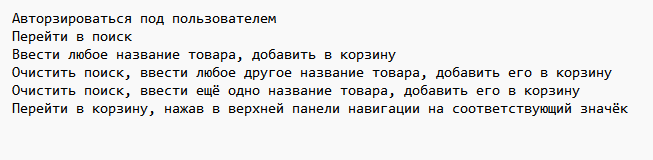
В нижней части окна отображается логин текущего пользователя и указанный run id. В этой же области находятся кнопка “Log out” – для выхода обратно на окно авторизации в системе. Рядом располагается кнопка “Run” – кнопка перехода к окну прохождения шагов тестирования.

## Редактирование тест-кейсов под админом

Администраторы могут редактировать состав тест-кейсов и выбирать, к каким комплектам они относятся. Поэтому для авторизовавшихся администраторов становятся активными кнопки “Move…” и “Bulk edit” (см. рис. 3).

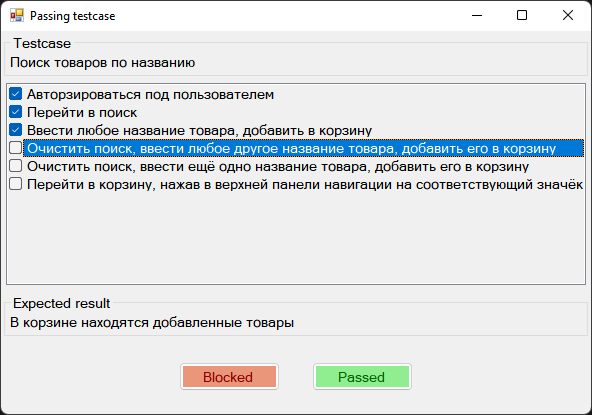
  
Рис. 3 – Bulk edit под админом

В режиме Bulk edit появляется окно (см. рис. 4), в котором построчно перечислены шаги тестирования тест-кейса. Содержимое этого окна можно редактировать: изменять текст или добавлять новые строчки. Сохранение изменений приведёт к тому, что изменится содержимое тест-кейса в базе данных и для последующих прохождений тест-кейсов.

Рис. 4 – Bulk edit тест-кейса

## Окно прохождения шагов тестирования

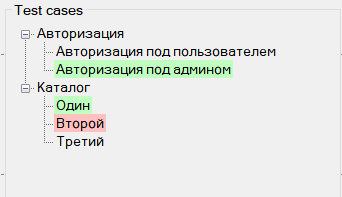
На рисунке 5 изображено окно прохождения шагов тестирования. На этом окне отображается информация о выбранном тестовом случае: название, ожидаемый результат и шаги, как прийти к этому результату. Шаги выполнены в виде списка чек-боксов для того, чтобы по ходу тестирования отмечать пройденные шаги.

  
Рис. 5 – Окно прохождения шагов тестирования

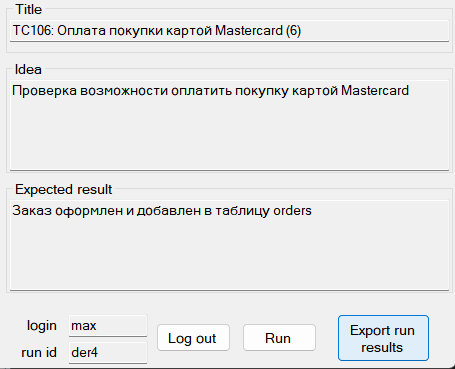
В случае если по завершению прохождения шагов тестирования пользователь приходит к ожидаемому результату, необходимо нажать кнопку “Passed”. Тогда в БД сохранится информация, что во время прохождения указанного Run id данный тест-кейс был пройден и ошибок не найдено. В случае если во время прохождения тестирования ожидаемый результат не получилось достичь, пользователь нажимает на кнопку Blocked. В БД сохраняется информация, что тест-кейс не удалось выполнить. Также, тестировщик должен сообщить разработчику об ошибке, сославшись на run id.

## Отчёты

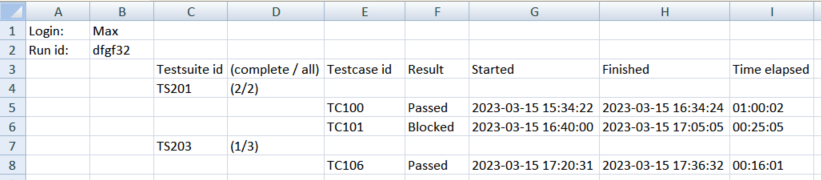
Во время прохождения тестирования, пройденные в текущем run id тест-кейсы будут помечаться зелёным или красным цветом, в зависимости от результата тестирования (Passed – зелёный, Blocked – красный). См. рис. 6.

  
Рис. 6 – Результат тестирования в интерфейсе

Закончив тестирование, пользователь может экспортировать результат тестирования в форме отчёта. Для этого нужно нажать на кнопку Export run results (см. рис. 7).

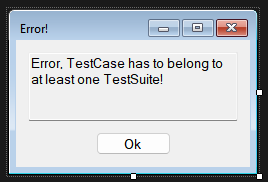
  
Рис. 7 – Export run results

Экспортированный результат будет иметь формат Excel документа и иметь разметку как на примере на рисунке 8.

Рис. 8 – Пример экспортированного отчёта

## Реакция на ошибочный ввод данных

Если во время перемещения тест-кейса не было выбрано родительского комплекта, пользователь видит сообщение об ошибке. На рисунке 9 изображено сообщение об ошибке.

  
Рис. 9 – Реакция на ошибочный ввод данных

# Разработка структуры классов

## Диаграмма классов

Прежде чем приступить к разработке, необходимо разработать диаграмму классов. На рисунке 1 изображена диаграмма классов, спроектированная средствами сайта diagrams.net.

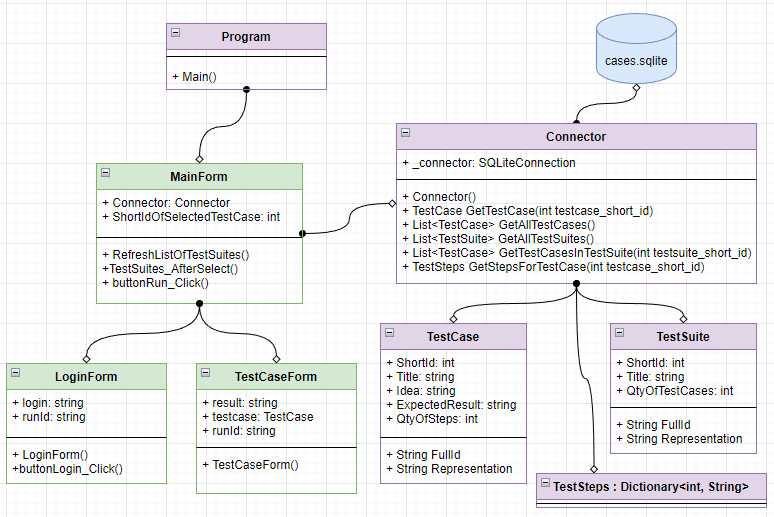
Класс Program – основной класс программы. При запуске у него вызывается метод Main(), который отрисовывает класс MainForm.

Класс LoginForm описывает отображение формы для авторизации для пользователя. Форма отрисовывается при запуске программы. По нажатию кнопки Log in вызывается метод buttonLogin\_Clicked(), который пропускает пользователя на главную форму.

Класс Connector описан для подключения к SQLite базе данных. Методы класса возвращают информацию из базы данных или загружают в неё информацию. Поле \_connector является объектом встроенного класса для подключения к базе данных. В таблице 1 описаны методы, их аргументы и возвращаемый результат, если он есть.

Таблица 1. Методы класса Connector

|  |  |  |
| --- | --- | --- |
| Название метода | Аргументы | Результат |
| Connector() | Нет | Конструктор класса |
| GetTestCase() | int testcase\_short\_id | Возвращает объект класса TestCase с указанным id, если он существует |
| GetAllTestCases() | Нет | Возвращает массив объектов всех описанных тест-кейсов |
| GetAllTestSuites() | Нет | Возвращает массив объектов всех описанных комплектов |
| GetTestCasesInTestSuite() | int testsuite\_short\_id | Возвращает массив объектов тест-кейсов из определённого комплекта |
| GetStepsForTestCase() | int testcase\_short\_id | возвращает экземпляр объекта |

 Рис. 1 – Диаграмма классов

В классе TestCase есть числовые поля ShortId, QtyOfSteps и текстовые поля Title, Idea, ExpetedResult. Также, у класса TestCase есть свойства FullId, который возвращает текст “TC” и добавляет ShortId на конце, и свойство Representation, которые возвращает FullId тест-кейса, его название и количество шагов в тест-кейсе в формате “TC999999: Название (16)”.

В классе TestSuite есть числовые поля ShortId и QtyOfTestCases и текстовое поле Title. У класса TestSuite есть такие же свойства, как и у класса TestCase, выполняющие ту же функцию, однако количество шагов заменено на количество тест-кейсов в комплекте.

Класс TestSteps наследуется от словаря с числовым ключом и текстовым значением.

В классе MainForm описан интерфейс главной формы. У этого класса есть два поля – Connector (объект класса Connector) и числовое поле ShortIdOfSelectedTestCase (id выбранного в интерфейсе TestCase’а). Метод RefreshListOfTestSuites() обновляет в интерфейсе список комплектов и тест-кейсов. Метод TestSuites\_AfterSelect() обновляет значение поля, в котором хранится выбранный TestCase. Метод buttonRun\_Click() запускает для пользователя окно формы TestCaseForm для прохождения тестирования.

Класс TestCaseForm описывает отображение окна для прохождения тестирования. Для запуска окно принимает в качестве аргументов объекты классов TestCase и TestSteps. Поле result хранит необходимый для достижения результат, поле testcase хранит объект класса TestCase, поле RunId хранит текущий установленный run id.

# Реализация программы

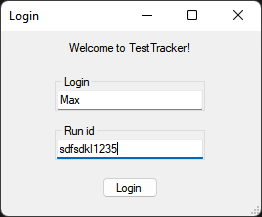
## Реализация

Программа реализована. Код программы написан в редакторе кода Visual Studio от компании Microsoft.

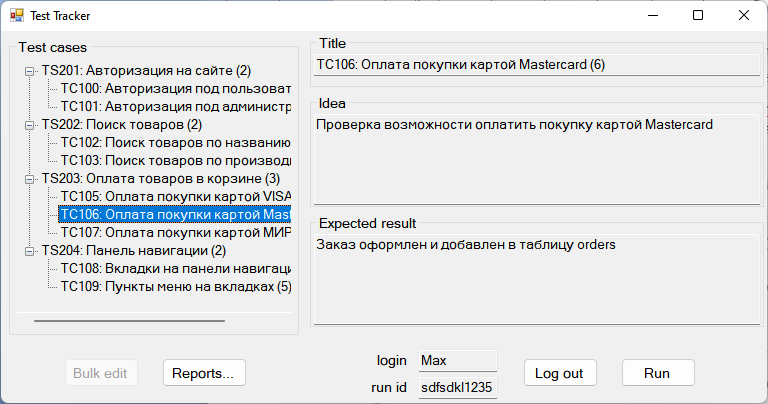
Код написан на языке C#, интерфейс программы разработан с помощью Windows Forms.

## Авторизированный доступ

При запуске программы появляется окно авторизации (см. рис 1). Пользователь вводит свой логин, под которым он хочет представиться, и свой run id, в рамках которого он будет проводить тестирование.

  
Рис. 1 – 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 2).

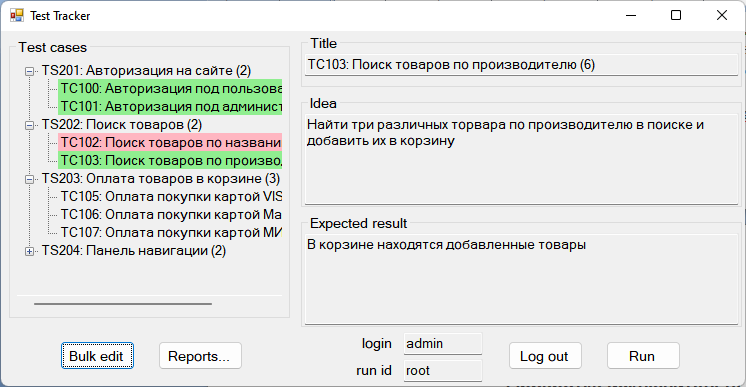
  
Рис 2 – Главное меню для пользователя

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.

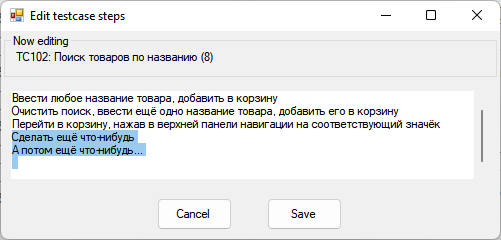
По мере того, как пользователь проходит тест-кейсы, элементы в дереве на главном меню окрашиваются в цвет, соответствующий результату тестирования: зелёный цвет означает, что в рамках указанного run id, тест-кейс пройден со значением “Passed”, красный – со значением “Blocked”.

На рисунке 3 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. 3 – Главное меню для администратора

## Редактирование шагов в тест-кейсах

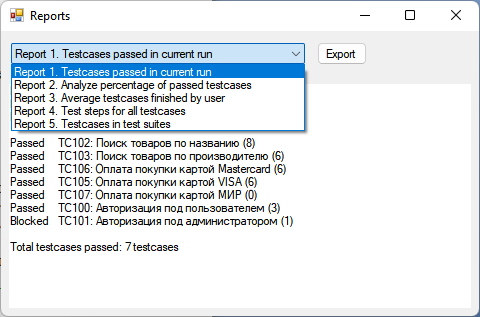
На рисунке 4 изображено окно редактирования шагов тестирования. Только администратор может перейти к этому окну, нажав на кнопку “ Bulk edit” в главном меню.

  
Рис. 4 – 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.

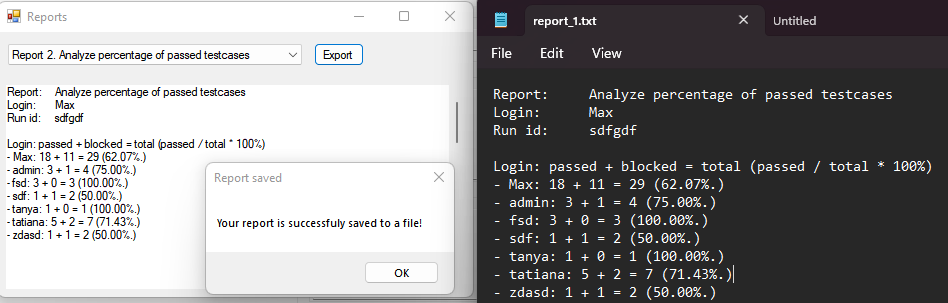
## Предоставление отчётности

На рисунке 5 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. 5 – Окно формирования отчётов

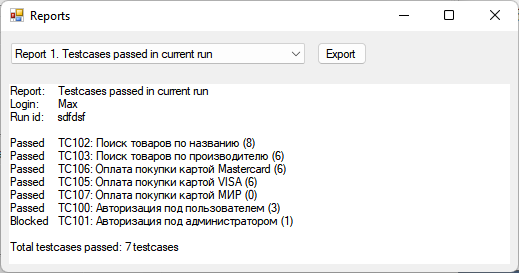
На окне Reports расположен селектор, с помощью которого можно выбрать нужный отчёт, кнопка экспорта отчёта во внешний файл, и текстовое поле, которое заполняется отчётом. При выборе отчёта в селекторе, текстовое поле очищается и заполняется только что выбранным отчётом.

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. 6).

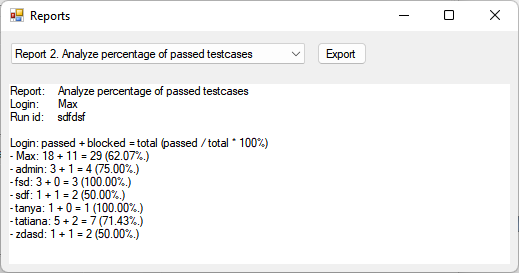
Рис. 6 – Сообщение об успешном экспорте

## Виды отчётностей

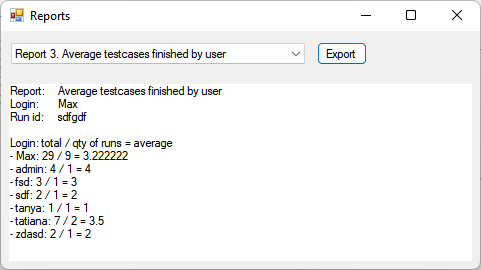
Первый отчёт выводит результат запуска тест-кейсов в текущем run\_id (см. рис. 7).

  
Рис. 7 – Первый отчёт

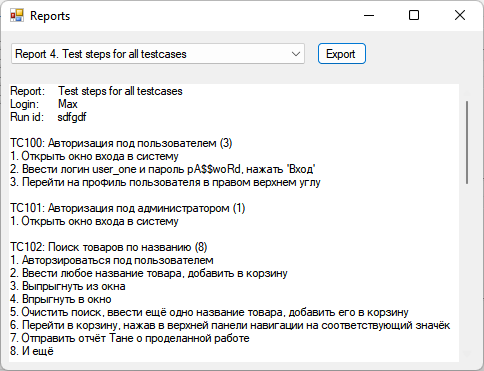
Второй отчёт выводит статистику всех запусков от всех пользователей (см. рис. 8). С помощью этого отчёта можно проанализировать, кто проходит больше тест-кейсов, и кто находит больше багов во время прохождения шагов.

  
Рис. 8 – Второй отчёт

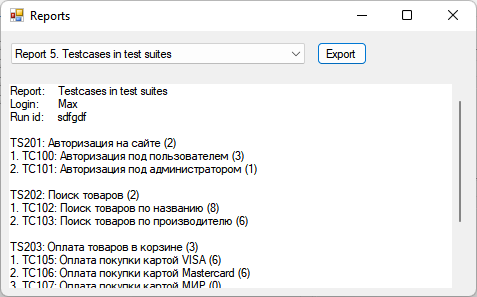
Третий отчёт выводит статистику по общему количеству пройденных тест-кейсов пользователями и количество запусков (см. рис. 9). Таким образом, можно проанализировать, какое количество тест-кейсов проходит каждый пользователь в среднем за один run.

  
Рис. 9 – Третий отчёт

Четвёртый отчёт выводит информацию по каждому тест-кейсу в системе и его шаги тестирования (см. рис. 10).

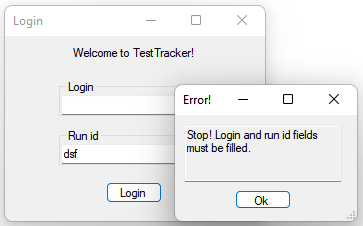
  
Рис. 10 – Четвёртый отчёт

Пятый отчёт выводит информацию по все комплектам и краткую информацию по тест-кейсам в них (см. рис. 11).

  
Рис. 11 – Пятый отчёт

## Обработка исключительных ситуаций

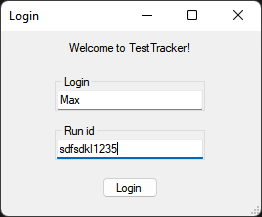
При попытке авторизоваться обработчик проверяет, оба ли поля заполнены. На рисунке 12 изображено окно с просьбой заполнить оба поля на форме авторизации, если хотя бы одно поле оказалось пустым.

  
Рис. 12 – Обработка полей ввода во время авторизации

# Тестирование программы

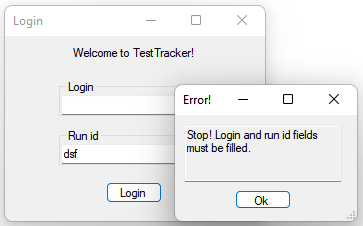
## Авторизированный доступ

При запуске программы появляется окно авторизации (см. рис 1). Пользователь вводит свой логин, под которым он хочет представиться, и свой run id, в рамках которого он будет проводить тестирование.

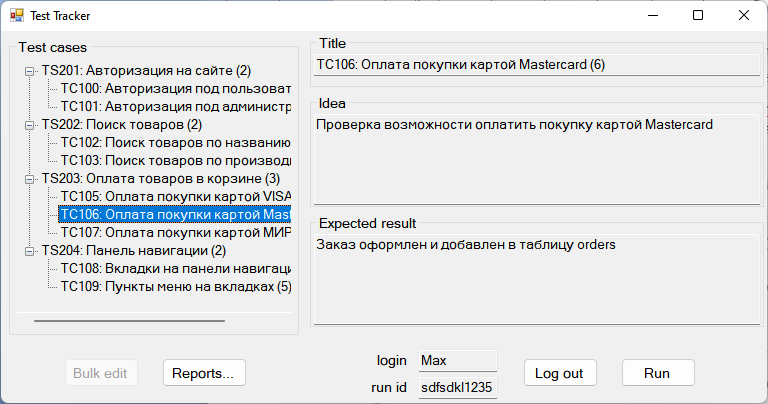
  
Рис. 1 – Окно авторизации

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 2 изображено окно с просьбой заполнить оба поля на форме авторизации, если хотя бы одно поле оказалось пустым.

  
Рис. 2 – Обработка полей ввода во время авторизации

Если авторизоваться как обычный пользователь (тестировщик), то в главном окне программы будут доступны все возможности, кроме редактирования шагов в тест-кейсах (см. рис 3).

  
Рис 3 – Главное меню для пользователя

Обратим внимание, что на рис. 3 внизу слева есть кнопка редактирования шагов тестирования “Bulk edit”. Она должна быть активной только у администратора. В таблице 1 приведены результаты тестирования этого компонента.

Таблица 1. Тестирование авторизации

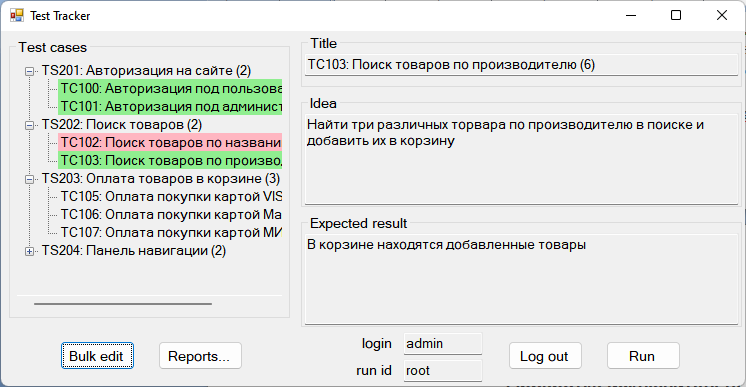
|  |  |  |
| --- | --- | --- |
| Login | Run id | Результат авторизации |
| «пустое поле» | «пустое поле» | Сообщение об ошибке с просьбой указать логин и пароль |
| Nikita | «пустое поле» | Сообщение об ошибке с просьбой указать логин и пароль |
| «пустое поле» | Fdgfdg3222 | Сообщение об ошибке с просьбой указать логин и пароль |
| Nikita | Fdwerklbdf | Открывается главное меню, Bulk edit не доступен |
| admin | root | Открывается главное меню, Bulk edit доступен |

Таким образом, окно авторизации протестировано, работает в соответствии с требованиями к программе.

## Работа с тест-кейсами

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 4 изображено окно, когда пользователь прошёл 4 тест-кейса.

  
Рис. 4 – Главное меню для администратора

В таблице 2 приведены результаты тестирования окрашивания тест-кейсов в интерфейсе в разных условиях.

Таблица 2. Результаты прохождения тест-кейсов

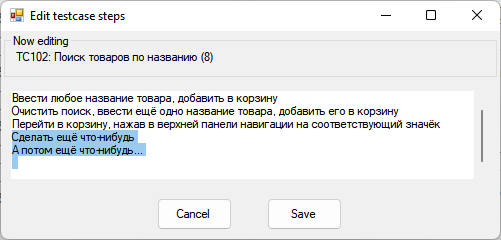
|  |  |  |  |
| --- | --- | --- | --- |
| Состояние тест-кейса до запуска | Результат запуска | Состояние тест-кейса после запуска | Результат тестирования |
| Не пройден, серый | Passed | Пройден, зелёный | Правильно |
| Не пройден, серый | Blocked | Пройден, красный | Правильно |
| Пройден, зелёный | Passed | Пройден, зелёный | Правильно |
| Пройден, зелёный | Blocked | Пройден, красный | Правильно |
| Пройден, красный | Passed | Пройден, зелёный | Правильно |
| Пройден, красный | Blocked | Пройден, красный | Правильно |

Результат тестирования показал, что тест-кейсы окрашиваются правильно при их прохождении в любом случае.

## Редактирование шагов в тест-кейсах

На рисунке 5 изображено окно редактирования шагов тестирования. Только администратор может перейти к этому окну, нажав на кнопку “ Bulk edit” в главном меню.

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.

  
Рис. 5 – Окно редактирования шагов в тест-кейсах

В таблице 3 приведены результаты тестирования реакции программы на нажатие разных кнопок во время редактирования шагов тест-кейсов.

Таблица 3. Реакция на нажатия кнопок в Bulk edit

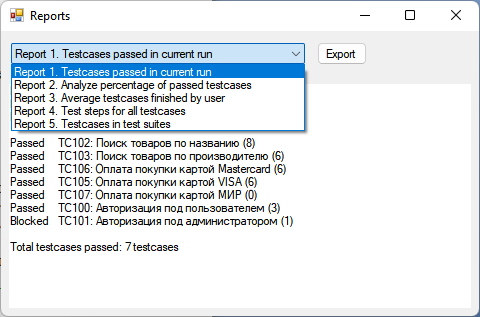
|  |  |  |
| --- | --- | --- |
| Нажатие на кнопку | Состояние БД после нажатия | Оценка результата |
| Save | Изменённый текст сохранился | Правильно |
| Cancel | Изменённый текст не сохранился | Правильно |
| «Закрыть окно» | Изменённый текст не сохранился | Правильно |

## Предоставление отчётности

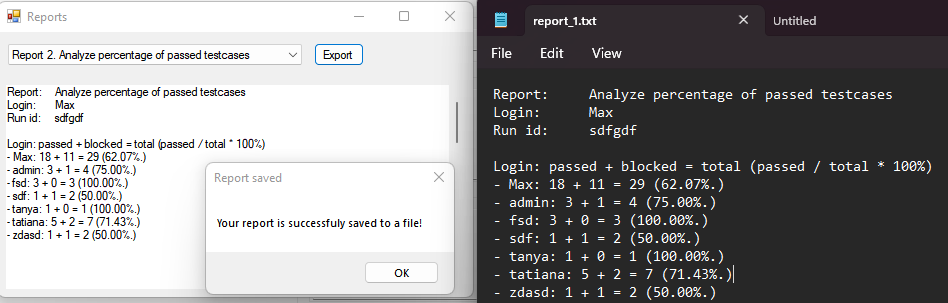
На рисунке 6 изображено окно формирования отчётов. Для перехода к этому окну необходимо нажать на кнопку “Reports…” на главном меню. Просматривать отчёты могут как пользователи, так и администраторы.

На окне Reports расположен селектор, с помощью которого можно выбрать нужный отчёт, кнопка экспорта отчёта во внешний файл, и текстовое поле, которое заполняется отчётом.

Тестирование показало, что при выборе любого доступного элемента в селекторе – текстовое поле в окне заполняется нужным отчётом.

  
Рис. 6 – Окно формирования отчётов

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. 7).

Рис. 7 – Сообщение об успешном экспорте

Тестирование показало, что каждый из 5 отчётов экспортируется без ошибок и в полном объёме.

### Приложение 1. Test case, Test suite, Test steps

namespace TestTracker.Models

{

public struct TestCase

{

public int ShortId;

public string Title;

public string Idea;

public string ExpectedResult;

public int QtyOfSteps;

public string FullId

{ get { return $"TC{ShortId}"; } }

public String Representation

{ get { return $"{FullId}: {Title} ({QtyOfSteps})"; } }

}

public struct TestSuite

{

public int ShortId;

public string Title;

public int QtyOfTestCases;

public string FullId

{ get { return $"TS{ShortId}"; } }

public string Representation

{ get { return $"{FullId}: {Title} ({QtyOfTestCases})"; } }

}

public class TestSteps : Dictionary<int, String>

{

public string RepresentAsText

{

get

{

string result = "";

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

{

result += this[i].ToString() + "\n";

}

return result;

}

}

}

}

### Приложение 2. Connector

namespace TestTracker

{

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 TestCase GetTestCase(int testcase\_short\_id)

{

string statement = $"select \* from view\_testcases where short\_id={testcase\_short\_id}";

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

SQLiteDataReader reader = cmd.ExecuteReader();

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);

return testcase;

}

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;

### Приложение 3. Connector

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;

}

public void InsertRunResult(string user\_login, string run\_id, int testcase\_short\_id, string timestamp\_started, string timestamp\_finished, string passed)

{

string statement = ExistsSavedResult(run\_id, testcase\_short\_id) ?

$"update progress set passed='{passed}' where user\_login='{user\_login}' and run\_id='{run\_id}' and testcase\_short\_id={testcase\_short\_id}" :

$"insert into progress (user\_login, run\_id, testcase\_short\_id, timestamp\_started, timestamp\_finished, passed) values ('{user\_login}', '{run\_id}', {testcase\_short\_id}, '{timestamp\_started}', '{timestamp\_finished}', '{passed}')";

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

cmd.ExecuteNonQuery();

}

public Dictionary<TestCase, string> GetResultsOfTestCases(string user\_login, string run\_id)

{

string statement = $"select testcase\_short\_id, passed from progress where user\_login='{user\_login}' and run\_id='{run\_id}'";

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

SQLiteDataReader reader = cmd.ExecuteReader();

Dictionary<TestCase, string> result = new Dictionary<TestCase, string>();

while (reader.Read())

{

int testcase\_short\_id = reader.GetInt32(0);

string passed = reader.GetString(1);

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

result.Add(tcase, passed);

}

return result;

}

}

### Приложение 4. Main Form

namespace TestTracker

{

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();

}

}

RefreshListOfTestSuites();

}

private void RefreshBasicInfo()

{

this.textBoxLogin.Text = this.user\_login;

this.textBoxRunId.Text = this.run\_id;

this.treeViewOfTestSuites.BeginUpdate();

foreach (TreeNode node\_suite in treeViewOfTestSuites.Nodes)

{

foreach (TreeNode node\_case in node\_suite.Nodes)

{

string passed = Connector.GetResultIfExists(this.user\_login, this.run\_id, Convert.ToInt32(node\_case.Tag));

switch (passed)

{

case "Passed":

node\_case.BackColor = Color.LightGreen;

break;

case "Blocked":

node\_case.BackColor = Color.LightPink;

break;

}

if (node\_case.Tag != null)

{

TestCase tcase = Connector.GetTestCase(Convert.ToInt32(node\_case.Tag));

node\_case.Text = tcase.Representation;

}

}

}

### Приложение 5. Main Form

private void treeViewOfTestSuites\_AfterSelect(object sender, TreeViewEventArgs e)

{

string FullIdOfTC = e.Node.Name;

if (FullIdOfTC.StartsWith("TC") == false) {

ShortIdOfSelectedTestCase = -1;

textBoxTitleOfTC.Clear();

textBoxIdeaOfTC.Clear();

textBoxExpectedResultOfTC.Clear();

buttonRun.Enabled = false;

return;

}

string substring = FullIdOfTC.Substring(2);

int testcase\_short\_id = Int32.Parse(substring);

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

ShortIdOfSelectedTestCase = testcase.ShortId;

textBoxTitleOfTC.Text = testcase.Representation;

textBoxIdeaOfTC.Text = testcase.Idea;

textBoxExpectedResultOfTC.Text = testcase.ExpectedResult;

buttonRun.Enabled = true;

}

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 btnBulkEdit\_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 EditStepsForm(tcase, tsteps))

{

var result = form.ShowDialog();

if (result == DialogResult.OK)

{

TestSteps newSteps = form.newSteps;

Connector.ReplaceTestSteps(tcase, newSteps);

}

RefreshBasicInfo(); }}}}}

### Приложение 6. Report Form

namespace TestTracker.Forms

{

public partial class ReportForm : Form

{

private Connector connector;

private string user\_login;

private string run\_id;

public ReportForm(Connector connector, string user\_login, string run\_id)

{

this.connector = connector;

this.user\_login = user\_login;

this.run\_id = run\_id;

InitializeComponent();

}

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))

blocked[pair.Key] = 0;

}

### Приложение 7. Report Form

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;

}

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)total[item.Key] / qty\_of\_runs[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)

{

### Приложение 8. Report Form

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;

}

}

}