Частное учреждение образования

«Колледж бизнеса и права»

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_ И. В. Малафей

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023

СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ПО АВТОМАТИЗАЦИИ СБОРА И ОБРАБОТКИ ДАННЫХ ДЛЯ ПОДГОТОВКИ И ПРОВЕДЕНИЯ ОЛИМПИАДЫ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ДП Т.992011.401

Председатель цикловой комиссии (Т. Г. Багласова )

Руководитель проекта (И. Б. Сватко )

Консультант по экономической части (Е. А. Андреева )

Консультант по охране труда (В. С. Кудласевич)

Учащийся (М. М. Быковский)

Рецензент ( )

2023

СОДЕРЖАНИЕ

Изм.

Лист

№докум.

Подпись

Дата

Лист

3

ДП Т.992011.401 ПЗ

Разраб.

Быковский М.М.

Провер.

Сватко И.Б.

Т. контр.

*Багласова Т.Г.*

Н. контр.

Басалыга Л.В.

Утверд.

Багласова Т.Г.

*СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ПО АВТОМАТИЗАЦИИ СБОРА И ОБРАБОТКИ ДАННЫХ ДЛЯ ПОДГОТОВКИ И ПРОВЕДЕНИЯ ОЛИМПИАДЫ*

Лит.

Листов

КБП

101

у

у

[Введение 4](#_Toc136961002)

[1 Описание задачи 6](#_Toc136961003)

[1.1 Анализ предметной области 6](#_Toc136961004)

[1.2 Постановка задачи 10](#_Toc136961005)

[2 Проектирование системы 12](#_Toc136961006)

[2.1 Требования к приложению 12](#_Toc136961007)

[2.2 Проектирование модели 13](#_Toc136961008)

[2.3 Проектирование структуры базы данных 14](#_Toc136961009)

[2.4 Концептуальный прототип 15](#_Toc136961010)

[3 Описание реализации программного средства 19](#_Toc136961011)

[3.1 Инструменты разработки и применяемые технологии 19](#_Toc136961012)

[3.2 Порядок авторизации пользователей 20](#_Toc136961013)

[3.3 Организация данных 21](#_Toc136961014)

[3.4 Функции: логическая и физическая организация 26](#_Toc136961015)

[3.5 Входные и выходные данные 30](#_Toc136961016)

[3.6 Модульное тестирование 31](#_Toc136961017)

[3.7 Описание справочной системы 42](#_Toc136961018)

[4 Применение 44](#_Toc136961022)

[4.1 Назначение программного средства 44](#_Toc136961023)

[4.2 Условия применения 44](#_Toc136961024)

[5 Охрана труда и окружающей среды 46](#_Toc136961025)

[5.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда 46](#_Toc136961026)

[5.2 Разработка мер по нормализации и защиты работающих от статического электричества при эксплуатации ПЭВМ 48](#_Toc136961027)

[5.3 Обеспечение пожарной безопасности 55](#_Toc136961028)

[5.4 Охрана окружающей среды 56](#_Toc136961029)

[6 Экономический раздел 58](#_Toc136961030)

[6.1 Обоснование необходимости выведения на рынок программного средства 58](#_Toc136961031)

[6.2 Составление плана по разработке программного средства 58](#_Toc136961032)

[6.3 Расчет затрат на разработку программного средства 59](#_Toc136961033)

[6.4 Расчет экономического эффекта у разработчика и пользователя программы 65](#_Toc136961034)

[6.4.1 Экономический эффект у разработчика программы 65](#_Toc136961035)

[6.4.2 Экономический эффект у пользователя программы 66](#_Toc136961036)

[Заключение 68](#_Toc136961037)

[Список информационных источников 69](#_Toc136961038)

[Приложение А Текст программы 71](#_Toc136961039)

[Приложение Б Результаты работы программы 91](#_Toc136961040)

# 

# Введение

В последние годы академические олимпиады приобрели популярность как средство оценки и признания академического мастерства. Однако подготовка и проведение таких соревнований требуют значительных ресурсов и опыта. В связи с этим, создание программного средства по автоматизации сбора и обработки данных для подготовки и проведения олимпиады является актуальным и затрагивает практически любое учреждение образования.

Целью дипломного проекта является создание программного средства по автоматизации сбора и обработки данных для подготовки и проведения олимпиады со следующими возможностями: регистрация участников олимпиады; выгрузка данных об итогах проведения олимпиады; публикация заданий для участников олимпиады:

* активации задания;
* получение решений от участников в заданные сроки;
* закрепление полученных решений за участником.

Для достижения цели дипломного проекта нужно решить следующие задачи:

* выполнить исследование предметной области и проектирование системы, результатом которой будет модель системы;
* по модели выполнить проектирование задачи;
* разработать программное средство;
* описать созданное программное средство;
* выбрать методику испытаний;
* провести тестирование разработанного программного средства на основании модульных тестов;
* описать условия применения и привести примеры области применения.

Решение поставленных задач отражено в пояснительной записке, которая состоит из шести разделов и содержит необходимую информацию по организации и использованию программного средства.

В первом разделе «Описание задачи» проводится исследование предметной области, описываются бизнес-процессы, происходящие в предметной области. Кроме того, описываются задачи из исследованной предметной области, которые могут быть решены при использовании программного средства, приводятся способы решения задач, анализируются имеющиеся способы решения задач в настоящее время, сделан акцент на описании существующих аналогов.

Второй раздел «Проектирование системы» описывает требования к программному средству и концептуальный прототип. Приводится описание спроектированных диаграмм в рамках исследованной предметной области.

В третьем разделе «Описание реализации программного средства» приводится описание разработанного программного средства: применяемые инструменты разработки и технологии, порядок авторизации пользователей, организация данных в рамках выбранной системы управления базами данных, логическая и физическая организация функций, входные и выходные данные. Кроме того, приводится тестирование на основании подготовленных модульных тестов, описывается справочная система.

Четвертый раздел «Применение» содержит описание назначения программного средства и условий его применения.

Пятый раздел «Охрана труда и окружающей среды» содержит информацию о правовых, нормативных, социально-экономических и организационных вопросах охраны труда, о разработке мер по нормализации и защите работающих от статического электричества при эксплуатации персональной электронно-вычислительной машины (ПЭВМ), о требованиях пожарной безопасности и охране окружающей среды.

В шестом экономическом разделе приводится обоснование необходимости выведения программного средства на рынок, составляется план по разработке программного средства и рассчитываются затраты на его разработку, а также рассчитывается экономический эффект у разработчика и пользователя программного средства.

В заключении подводится общий итог о проделанной работе, анализируется степень реализации функций, извлекаются соответствующие выводы о качестве разработанного программного средства.

В приложении А представлен текст программы.

В приложении Б представлены результаты работы программы.

В графической части представлены диаграмма вариантов использования, диаграмма деятельности и диаграмма «Сущность-связь».

1. Описание задачи

## Анализ предметной области

Соревнования давно стали неотъемлемой частью человеческого общества, предоставляя отдельным людям и командам возможность продемонстрировать свои навыки и посоревноваться с другими в самых разных областях. Эти соревнования разнообразны: от академических конкурсов до спортивных турниров.

Академические конкурсы­ это соревнования, которые демонстрируют знания и понимание учеником определенного предмета и могут варьироваться от местных соревнований до национальных и даже международных конкурсов: научные ярмарки, конкурсы орфографии и так далее.

Спортивные турниры ­ форма соревнований, в которых спортсмены соревнуются в самых разных видах спорта, от футбола до баскетбола, от плавания до легкой атлетики. Эти соревнования часто организуются школами, клубами или спортивными ассоциациями, и на них могут съезжаться спортсмены со всего мира.

Одним из видов соревнований является олимпиада ­ соревновательное мероприятие, на котором учащиеся из различных учебных заведений могут продемонстрировать свои знания и навыки в различных областях образования, соревнуясь со своими сверстниками.

Одна из областей образования, которая приобрела значительную популярность в последние годы, является программирование. Соревнования по программированию требуют от участников умения использовать свои навыки программирования для решения сложных задач, требующих творческого и аналитического мышления, часто связанных с алгоритмами и структурами данных. Олимпиады по программированию часто открыты для учащихся старших классов и колледжей, а также для профессионалов, и могут проводиться как на местном, так и на международном уровне.

Сбор и анализ данных ­ важнейшая часть организации успешной олимпиады. Анализируя результаты участников, организаторы могут:

* определить эффективность заданий, формат соревнования и уровень сложности. Эта информация может быть использована для улучшения будущих соревнований, корректировки уровня сложности или изменения формата, чтобы сделать соревнования интересными для участников;
* дать ценную информацию о сильных и слабых сторонах отдельных участников, а также об общей результативности в разрезе различных показателей. Эта информация может быть использована для адаптации заданий и соревнований к индивидуальным потребностям и способностям участников, что в конечном итоге повысит общее качество олимпиады.

Одним из примеров олимпиад в области программирования является Международная олимпиада в сфере информационных технологий «IT-Планета», которая дает шанс для учащихся, студентов и молодых дипломированных специалистов проявить себя в сфере информационных технологий в реальном контексте. Следует отметить и ежегодную Международную олимпиаду по информатике (IOI), Международное соревнование по программированию Google Code Jam и Международный студенческий конкурс по программированию ACM (ICPC). Организаторы этих соревнований предоставляют участникам платформу для демонстрации своих навыков программирования и возможность соревнования с другими участниками на глобальной арене.

Для участия в Международной олимпиаде в сфере информационных технологий «IT-Планета» необходимо зарегистрироваться как участникам, которые планируют участвовать в олимпиаде, так и образовательным учреждениям, которые представляют участники олимпиады. Окно регистрации участника представлено на рисунке 1.1.

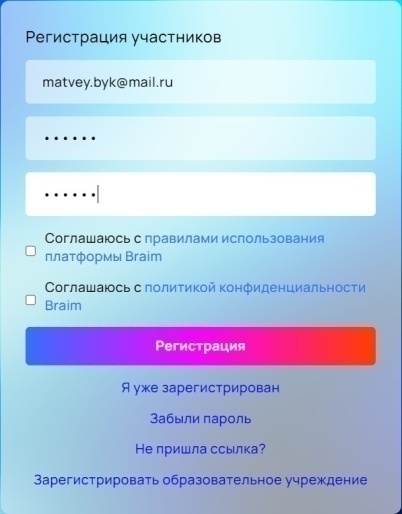


Рисунок 1.1

Окно регистрации образовательного учреждения представлено на рисунке 1.2.

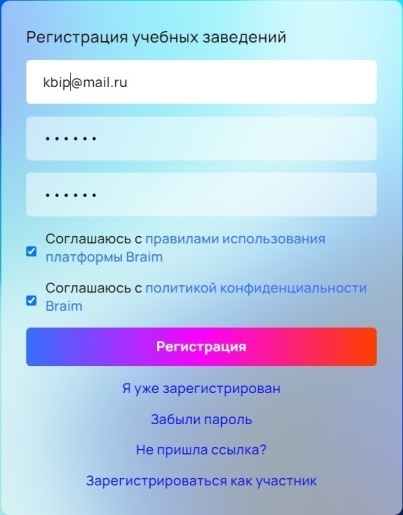


Рисунок 1.2

Для каждого зарегистрированного участника доступен личный кабинет со следующими возможностями: просмотр данных профиля (вкладка «Профиль»), просмотр выбранных категорий соревнований (вкладка «Соревнования»), просмотр полученных сертификатов (вкладка «Достижения»). Вид личного кабинета с активной вкладкой «Профиль» представлен на рисунке 1.3.

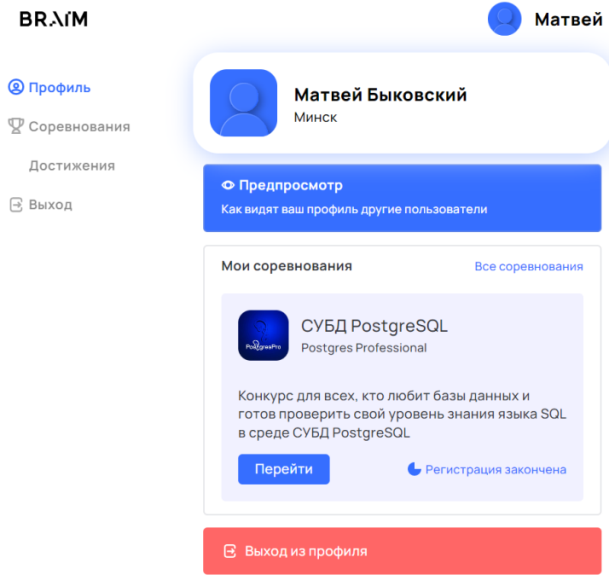


Рисунок 1.3

Окно редактирования данных участника представлено на рисунке 1.4

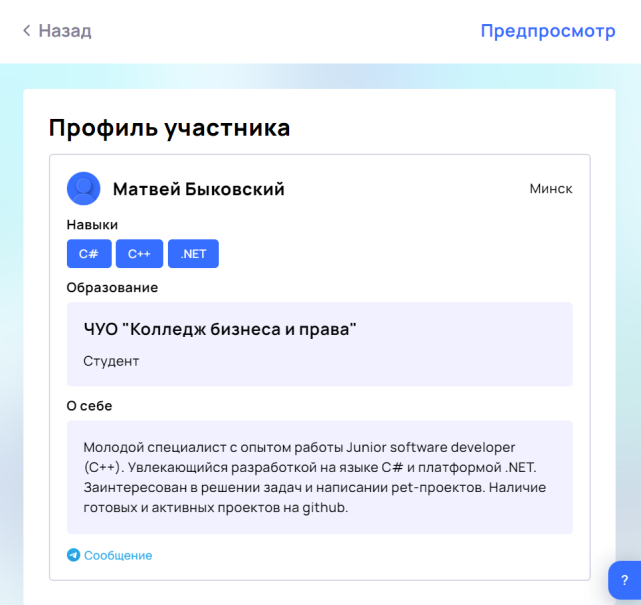


Рисунок 1.4

Окно личного кабинета участника с активной вкладкой «Соревнования» представлено на рисунке 1.5.

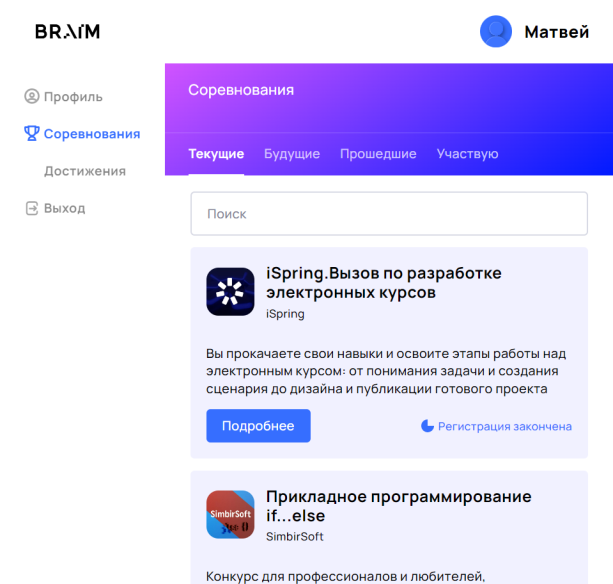


Рисунок 1.5

Окно личного кабинета с активной вкладкой «Достижения» представлено на рисунке 1.6.

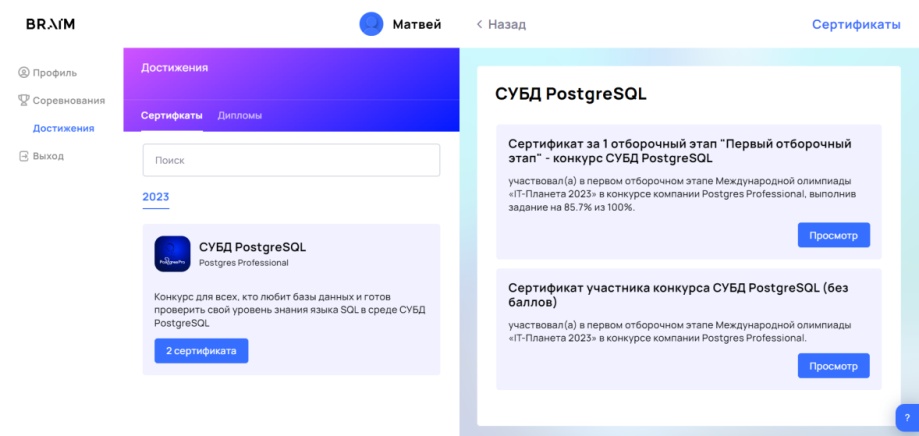


Рисунок 1.6

Для каждого зарегистрированного учебного заведения доступен личный кабинет ответственного от учреждения образования. Вкладки окна личного кабинета ответственного от учреждения образования представлены на рисунке 1.7.

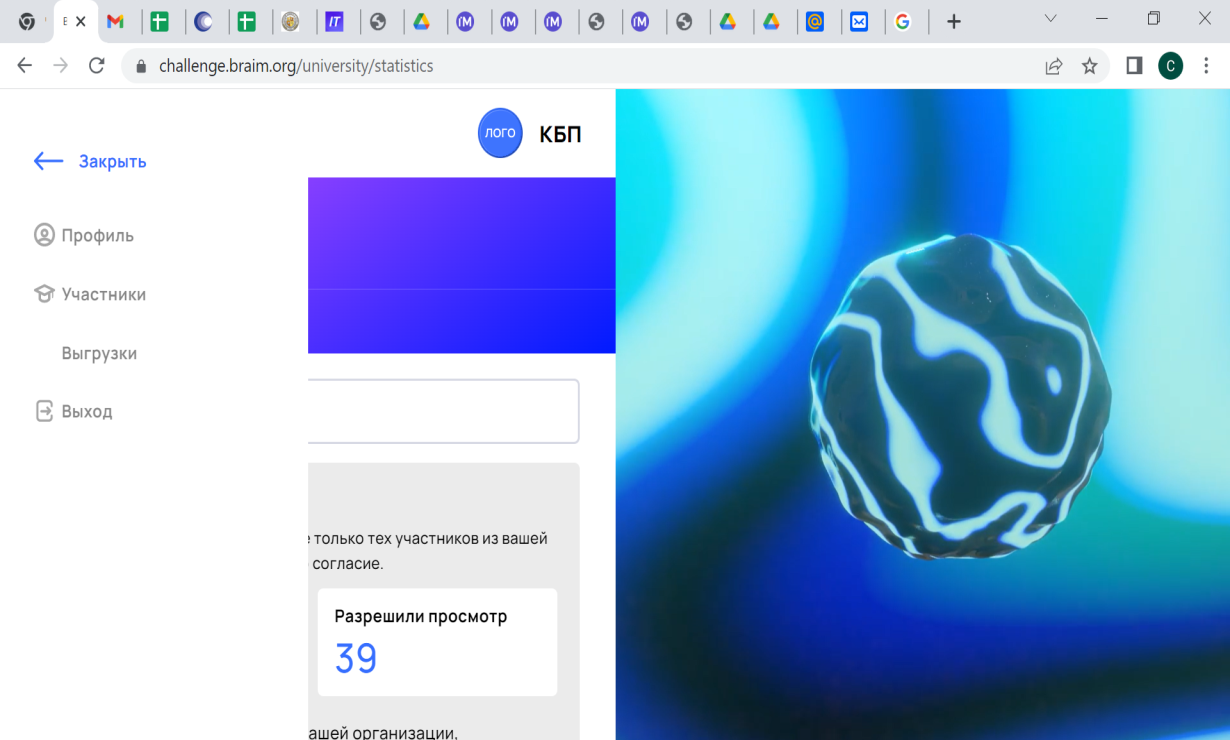


Рисунок 1.7

## Постановка задачи

Частное учреждение образования «Колледж бизнеса и права» проводит олимпиады по программирования между своими учащимися и учащимися филиалов. Для организации регистрации участников на олимпиаду используют онлайн-инструмент Google Forms, позволяющий создавать формы для сбора данных, а для проверки знаний участников используется бесплатный интернет-сервис «MasterTest».

При проведении олимпиад описанным способом возникает ряд неудобств для организаторов. Данные, собранные во время регистрации, выгружаются в программу для работы с электронными таблицами Microsoft Excel и далее требуется дополнительная обработка организаторами стандартными инструментами для получения необходимой аналитической отчетности. Кроме того, отсутствует интеграция между службой регистрации и службой тестирования знаний участников олимпиады, что приводит к значительным временным и трудовым затратам для получения общей результативности в разрезе различных показателей. Соответственно адаптировать задания и соревнования к индивидуальным потребностям и способностям участников в конечном итоге не представляется возможным.

Исходя из анализа предметной области, можно выделить следующие задачи, подлежащие автоматизации:

* предусмотреть работу программы в режиме различных пользователей: «Организатор», «Участник»;
* реализовать возможность регистрации участников олимпиады;
* в режиме организатора олимпиады предусмотреть возможности просмотра информации об участниках олимпиады и выгрузки данных об итогах проведения олимпиады;
* в режиме организатора олимпиады предусмотреть возможности публикации заданий для участников олимпиады (публикация заданий до начала олимпиады; активация заданий), получения решений от участников в заданные сроки, закрепления полученных решений за участником.

Разрабатываемое в рамках дипломного проектирования программное средство по автоматизации сбора и обработки данных для подготовки и проведения олимпиады объединит функционал регистрации и тестирования, обеспечив беспрепятственную передачу данных между ними. Программное средство исключит необходимость дополнительной обработки данных с использованием электронных таблиц Microsoft Excel и обеспечит организаторам возможность анализа результатов тестирования именно зарегистрированных участников. Кроме того, программное средство повысит общую эффективность организации и проведения олимпиад, сделав процесс более удобным и доступным.

На данный момент существуют аналоги разрабатываемого программного средства:

* онлайн платформа «Coderbyte», предлагающая конкурсы и соревнования для программистов всех уровней подготовки;
* платформа для проведения соревнований по программированию «Kattis»;
* системы управления конкурсами (CMS), разработанные специально для управления конкурсами и олимпиадами по кодированию.

Каждая из этих платформ имеет свои преимущества и недостатки. Разрабатываемое программное средства в рамках дипломного проекта обеспечит баланс между всеми факторами, предоставляя собой удобный способ автоматизации сбора и обработки данных для подготовки и проведения олимпиады в частном учреждении образования «Колледж бизнеса и права».

1. Проектирование системы

## Требования к приложению

Разрабатываемое программное средство должно иметь понятный и удобный в использовании пользовательский интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено по следующим направлениям:

* взаимодействие пользователя с пользовательским интерфейсом и формам;
* доступность пользователя к внутренней функциональности программы при помощи пользовательского интерфейса.

Внешний интерфейс системы должен быть интуитивно понятным для пользователей с минимальной подготовкой и соответствовать требованиям современных настольных приложений. В связи с этим будет использована дизайн-система «Material Design», которая соответствует рекомендациям по созданию интерфейсов от Google: поддерживает инструменты для эффективной навигации и обратной связи; ориентирована на пользователя; обладает единым визуальным языком и простотой.

Основными цветами форм будут фиолетовый и розовый, а синий, красный, голубой и желтый будут используются для отображения элементов навигационного меню и некоторых элементов управления. Для визуального представления различных функций и возможностей в интерфейс будут включены иконки, соответствующие рекомендациям Google. Они обеспечат интуитивно понятные визуальные подсказки, помогающие пользователям эффективно ориентироваться и взаимодействовать с программой. Каждая иконка будет тщательно продумана, чтобы четко и последовательно передавала свое назначение.

Для обеспечения легкого доступа к различным разделам или окнам программного средства будет предусмотрено навигационное вертикальное меню слева.

В интерфейсе будет использован универсальный шрифт «Roboto» в различных вариациях, который специально оптимизирован для читаемости на различных цифровых платформах.

Чтобы обеспечить удобство и эффективность работы как ответственных за проведение олимпиады, так и участников олимпиад, программное средство должно включать в себя систему логинов и паролей с двумя ролями: «Ответственный» и «Участник». Помимо аспекта авторизации, будет продуман дизайн и конфигурация интерфейса программного средства для каждой роли. Это обеспечит удобство и индивидуальный подход к пользователю. Оптимизировав внешний вид для каждой роли, программное средство сможет предоставить ответственным за проведение олимпиады интуитивно понятный интерфейс для управления информацией об участниках, загрузки результатов и публикации заданий. Аналогично, участники олимпиады должны иметь удобный интерфейс, позволяющий им легко регистрироваться на олимпиаду, проходить тесты и просматривать свои личные результаты.

Функциональные требования представлены на диаграмме структурного анализа и проектирования (SADT), которая представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области и отображает функциональную структуру объекта, т.е. производимых им действий и связей между ними. Методология SADT предназначена для моделирования широкого круга информационных систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям. Диаграмма SADT нулевого уровня представлена на рисунке 2.1.

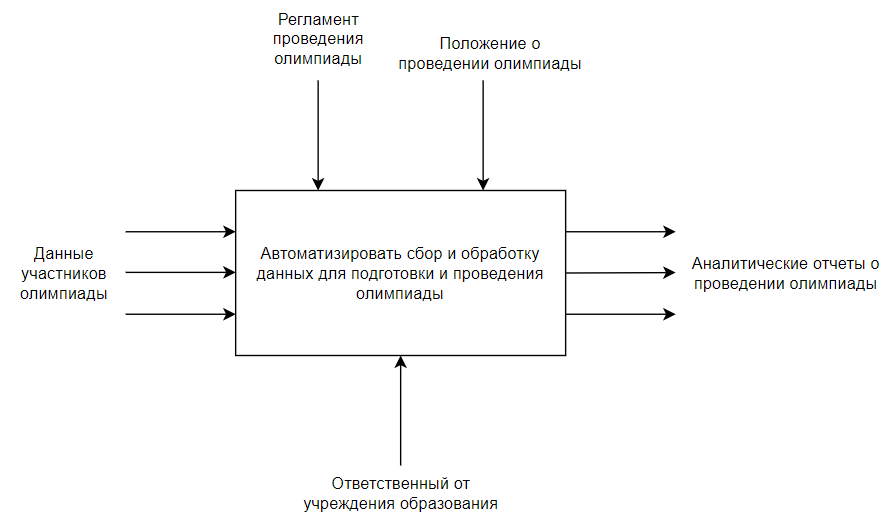


Рисунок 2.1.

## Проектирование модели

Диаграмма вариантов использования служит для отражения отношений между актерами и прецедентами. Основная цель использования таких диаграмм – визуализация возможностей пользователя в программном средстве.

Основными функциями программы являются:

* регистрация участников олимпиады;
* просмотр информации об участниках олимпиады;
* возможность публикации викторин для участников олимпиады;
* возможность публикации викторин до начала олимпиады;
* возможность активации викторины;
* возможность получения решений от участников в заданные сроки;
* генерирование отчетов.

Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма деятельности­ это графическое представление потока деятельности, действий или задач в системе. Она используется для моделирования потока управления в программе и дает четкое представление о последовательности действий, выполняемых в конкретном сценарии или процессе. На диаграмме деятельности действия изображаются в виде закругленных прямоугольников, а переходы между действиями ­ стрелками. Решения, циклы и параллельные потоки также могут быть представлены на диаграммах деятельности. Основная цель диаграммы деятельности ­ обеспечить четкое и ясное понимание поведения системы, выявить потенциальные узкие места или области для улучшения, а также облегчить общение между разработчиками и заинтересованными.

Диаграмма деятельности для создания и активации события представлена в графической части на листе 2.

## Проектирование структуры базы данных

Диаграмма «Сущность-связь»­ это графическое представление сущностей и их связей друг с другом. Она используется при проектировании баз данных для иллюстрации отношений между таблицами, сущностями, атрибутами и ключами.

Диаграмма «Сущность-связь» состоит из трех основных компонентов: сущности, атрибуты и отношения. Сущности ­ это объекты или понятия в системе, такие как пользователи или викторины. Атрибуты описывают характеристики сущности, например, имя или количество вопросов. Отношения описывают связи между сущностями, например, ответственный создает викторину.

Определим сущности для данного программного средства и построим диаграмму «Сущность-связь». Исследовав предметную область, можно выделить следующие сущности, относящиеся к данному проекту: «Пользователь», «Роль», «Категория», «Вопрос», «Ответ», «Викторина»,«Событие», «ЗапланированноеСобытие», «Группа».

Диаграмма «Сущность-связь» представлена в графической части на листе 3.

Для сущности «Пользователь» атрибутами будут являться:

* имя;
* фамилия;
* почта;
* датаСоздания.

Для сущности «Роль» атрибутом будет являться «Название».

Для сущности «Категория» атрибутами будут являться:

* название;
* датаСоздания.

Для сущности «Вопрос» атрибутами будут являться:

* описание;
* номер;
* типПроверки;
* датаСоздания.

Для сущности «Ответ» атрибутами будут являться:

* описание;
* верныйОтвет;
* датаСоздания.

Для сущности «Викторина» атрибутами будут являться:

* название;
* описание;
* времяНаПрохождение;
* пароль;
* датаСоздания.

Для сущности «Событие» атрибутами будут являться:

* название;
* статус;
* датаИВремяАктивации;
* продолжительность;
* датаСоздания.

Для сущности «ЗапланированноеСобытие» атрибутами будут являться:

* активировано;
* датаСоздания.

Для сущности «Группа» атрибутами будут являться:

* название;
* датаСоздания группы.

## Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – системы меню и окон.

Основными наиболее часто используемыми компонентами на формах будут являться:

* кнопка «Button» для запуска действий или событий, таких как отправка формы, сохранение данных, удаление информации или выполнение определенных задач;
* поле для ввода текстовой информации «Textbox» для ввода и редактирования текстовых или числовых данных;
* список предопределенных опций «ComboBox» для выбора критерия поиска данных из элемента «Grid»;
* контейнер компоновки «Grid» для отображения данных в табличном формате;
* флажок «Checkbox» для предоставления пользователям множественного выбора (включение или выключение опций, выбор нескольких элементов из списка, подтверждение действия).

При запуске программы будет отображаться окно авторизации для пользователя, макет которого представлен на рисунке 2.1.

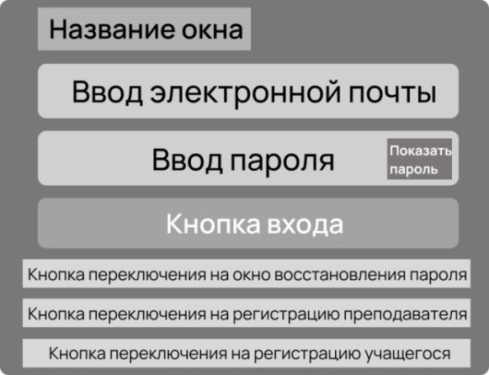


Рисунок 2.1

В окне авторизации пользователя будут представлены возможности авторизации существующего пользователя, возможность перехода кокну регистрации нового пользователя либо кокну восстановления пароля. При нажатии на кнопку «Восстановление пароля» будет отображаться окно для восстановления пароля пользователя, макет которого представлен на рисунке 2.2.

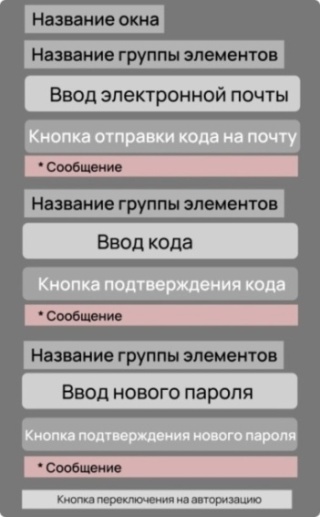


Рисунок 2.2

В окне восстановления пароля пользователя будет представлена возможность восстановления пароля с помощью определения нового.

При авторизации пользователя в роли организатора, будет отображаться главное окно приложения, макет которого представлен на рисунке 2.3.

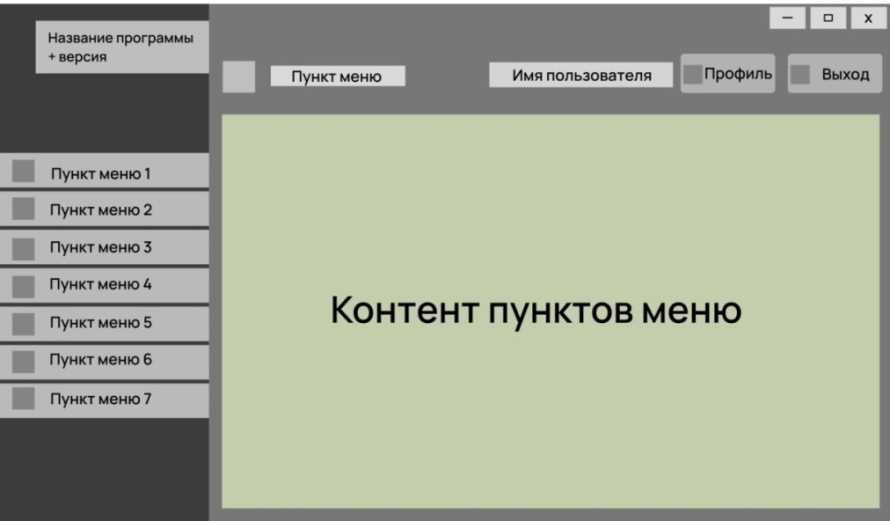


Рисунок 2.3

В главном окне для организаторов будут представлены возможности публикации и активации заданий, получения и обработки решений от участников в заданные сроки

В главном окне приложения, макет которого представлен на рисунке 2.3, при выборе вкладки «События» организатору будет отображено окно с информацией о событиях, макет которого представлен на рисунке 2.4.

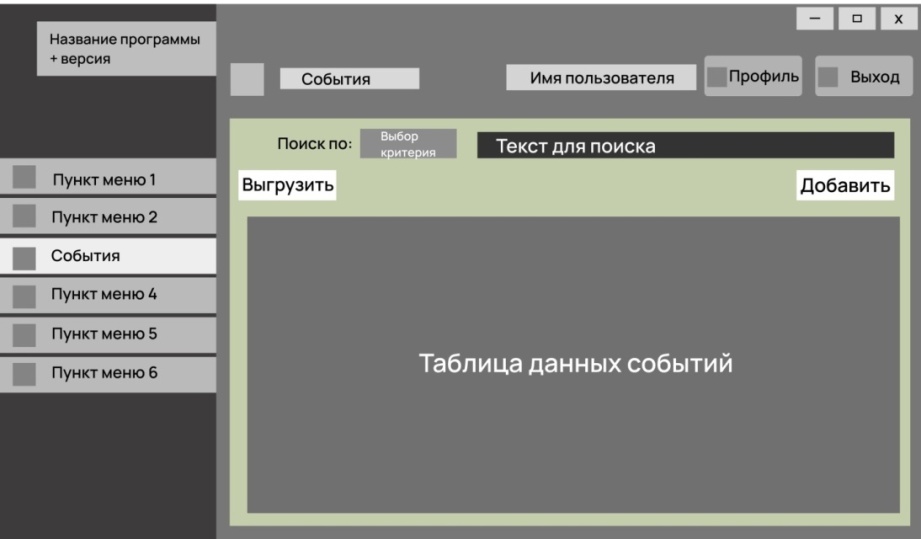


Рисунок 2.4

В окне с информацией о событиях будет представлена возможность создания события и управления событиями. Для добавления события будет предусмотрена кнопка «Новое событие». После ее нажатия будет отображено окно добавления данных о новом событии, макет которого представлен на рисунке 2.5.

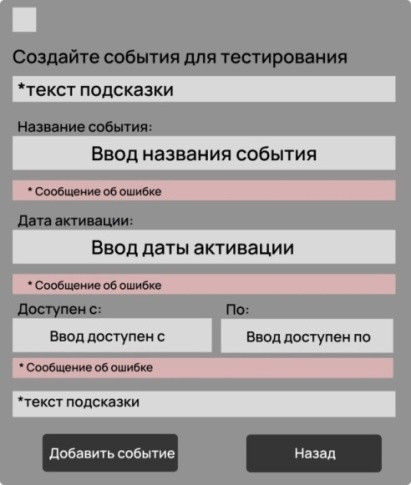


Рисунок 2.5

Для изменения параметров события необходимо будет нажать кнопку «Настройки», после чего будет отображено окно настроек события, макет которого представлен на рисунке 2.6.

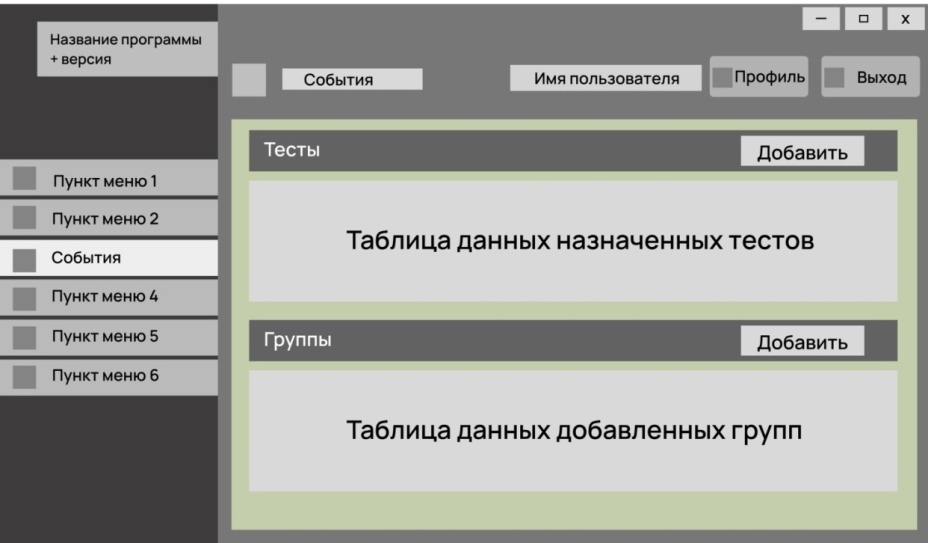


Рисунок 2.6

1. Описание реализации программного средства

## Инструменты разработки и применяемые технологии

При разработке программного средства по автоматизации сбора и обработки данных для подготовки и проведения олимпиады использовались следующие инструменты разработки и технологии:

* операционная система Windows 10 x64;
* среда программирования Visual Studio 2022;
* язык программирования C#;
* технология Windows Presentation Foundation(WPF);
* объектно-ориентированная технология для работы с базой данных Entity Framework;
* система управления базами данных MySQL;
* инструмент для создания диаграмм и схем Draw.io.

Выбор инструментов разработки и технологий для создания программы по автоматизации сбора и обработки данных для подготовки и проведения олимпиады был сделан на основе тщательного рассмотрения и оценки различных факторов. Каждый инструмент и технология были выбраны с учетом их конкретных преимуществ и соответствия требованиям проекта.

Программное приложение разрабатывалось и тестировалось под управлением операционной системы Windows 10 [6].

Операционная система Windows 10 была выбрана благодаря ее широкому распространению и надежному набору инструментов для разработчиков. Она обеспечивает стабильную и привычную среду для разработки программного обеспечения, гарантируя совместимость с широким спектром аппаратных и программных компонентов. Популярность Windows 10 также обеспечивает хорошую доступность ресурсов для устранения неполадок и разработки.

Visual Studio ­­это комплексная и широко распространенная интегрированная среда разработки (IDE), которая предлагает множество функций для улучшения процесса разработки программного обеспечения [7]. Она предоставляет обширный набор инструментов для кодирования, отладки, тестирования и развертывания приложений. Visual Studio 2022 была выбрана специально для использования новейших функций и улучшений, обеспечивая современный опыт разработки.

C# ­ это мощный и универсальный язык программирования, обеспечивающий высокоуровневое управление и высокую производительность [8]. Широкое применение и обширные библиотеки делают его надежным выбором для разработки эффективных и надежных программных приложений. Кроме того, выбор языка C# соответствует существующей системе программного обеспечения и обеспечивает совместимость с другими технологиями, используемыми в проекте.

WPF был выбран в качестве основы пользовательского интерфейса благодаря своим богатым возможностям и гибкости при создании визуально привлекательных настольных приложений [9]. С WPF разработчики могут использовать возможности XAML для проектирования и настройки пользовательских интерфейсов, а декларативная модель программирования упрощает отделение логики пользовательского интерфейса от бизнес-логики, что приводит к созданию поддерживаемого и масштабируемого кода.

Entity Framework представляет собой каркас объектно-реляционного отображения (ORM), который упрощает взаимодействие с базой данных, позволяя разработчикам работать с сущностями базы данных как с объектами [10]. Использование Entity Framework способствует абстрагированию, повышает производительность и снижает потребность в ручных SQL-запросах. Выбор объектно-ориентированной технологии соответствует цели проекта по созданию программных компонентов, которые инкапсулируют функциональность и обеспечивают четкую модель программирования.

Выбор MySQL в качестве системы управления базами данных для программного средства обоснован широким распространением, экономичностью и совместимостью с различными операционными системами [14]. Проверенная надежность, масштабируемость и производительность MySQL делают ее надежным выбором для управления большими объемами данных. Кроме того, совместимость MySQL с различными операционными системами и экономическая эффективность способствуют его пригодности для данного проекта.

Инструмент Draw.io был выбран в качестве инструмента для создания диаграмм UML и других визуальных представлений благодаря своей широкой функциональности и простоте использования [11]. Он предоставляет широкий выбор шаблонов и поддерживает различные типы диаграмм, что делает его подходящим для представления сложных отношений и структур в ясной и интуитивно понятной манере.

## Порядок авторизации пользователей

Аутентификация и авторизация ­ важнейшие аспекты для безопасного функционирования любого программного средства, работающего с конфиденциальной информацией. Создавая роли, предоставляя им необходимые привилегии и назначая роли пользователям, можно гарантировать, что пользователи смогут получить доступ только к той информации, которая относится к их ролям.

В программном средстве по автоматизации сбора и обработки данных для подготовки и проведения олимпиады предусмотрены две роли – «Организатор» и «Участник». Важно точно определить характеристики ролей и наделить их необходимыми привилегиями.

Пользователь в роли «Организатор» может создавать и настраивать викторины с вопросами, создавать категории и распределять викторины по категориям, создавать группы и добавлять в них участников, назначать викторины участникам (группам участников) и просматривать их результаты, выгружать необходимую информацию. С другой стороны, пользователь в роли «Участник» может только выполнять назначенные тесты и просматривать свои результаты.

Пользователям с любой ролью необходимо зарегистрироваться посредством ввода электронной почты, имени, фамилии и пароля.

Для входа в программу и тестирования функционала в роли участника необходимо в поле «Почта» ввести «student@mail.ru», в поле «Пароль» ввести «123456», в роли организатора – «organizer@mail.ru» и «123456».

## Организация данных

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных.

Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

База данных соответствует реляционной модели данных, где каждый выделенной в ходе проектировании сущности соответствует таблица.

Структура базы данных разрабатываемого программного средства включает 13 таблиц.

В таблице «aspnetroles» отображена информация о ролях пользователей, структура которой представлена в таблице 3.1.

Таблица 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор роли |
| Name | varchar | 256 | Название роли |
| NormalizedName | varchar | 256 | Нормализованное название роли |
| ConcurrencyStamp | longtext | 4 | Механизм отслеживания версий роли |

В таблице «aspnetusers» отображена информация о пользователях, структура которой представлена в таблице 3.2. Таблица «aspnetusers» включает поля по умолчанию, которые автоматически генерируются: «UserName», «NormalizedUserName», «NormalizedEmail», «EmailConfirmd», «SecurityStamp», «ConcurrencyStamp», «PhoneNumber», «PhoneNumberConfirmed», «TwoFactorEnabled», «LockoutEnd», «LockoutEnabled» и «AccessFailedCount». Однако важно отметить, что эти поля являются необязательными и не используются в программе, поэтому присутствуют на схеме базы данных, представленной на рисунке 3.1, но не описаны на диаграмме «Сущность-связь» и в таблице 3.2.

Таблица 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор пользователя |
| TeacherId | varchar | 255 | Уникальный идентификатор организатора из таблицы «aspnetusers» |
| FirstName | longtext | 4 | Имя пользователя |
| LastName | longtext | 4 | Фамилия пользователя |
| Email | varchar | 256 | Адрес электронной почты пользователя |
| CreatedOn | datetime | 8 | Дата и время создания пользователя |
| PasswordHash | longtext | 4 | Хеш пароля пользователя |

В таблице «categories» описана информация о категориях викторин, структура которой представлена в таблице 3.3.

Таблица 3.3

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор категории |
| CreatorId | varchar | 255 | Уникальный идентификатор организатора из таблицы «aspnetusers» |
| Name | longtext | 4 | Название категории |
| CreatedOn | datetime | 8 | Дата и время создания категории |

В таблице «questions» описана информация о вопросах для викторин, структура которой представлена в таблице 3.4.

Таблица 3.4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор вопроса |

Продолжение таблицы 3.4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| QuizId | varchar | 255 | Уникальный идентификатор викторины из таблицы «quizzes» |
| Text | longtext | 4 | Описание вопроса |
| Number | int | 4 | Порядковый номер вопроса |
| IsFullEvaluation | tinyint | 1 | Индикатор проверки вопроса |
| CreatedOn | datetime | 8 | Дата и время создания вопроса |

В таблице «answers» хранятся ответы для вопросов, структура которой представлена в таблице 3.5.

Таблица 3.5

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор ответа |
| QuestionId | varchar | 255 | Уникальный идентификатор вопроса из таблицы «questions» |
| Text | longtext | 4 | Описание ответа |
| IsRightAnswer | tinyint | 1 | Индикатор правильного ответа |
| CreatedOn | datetime | 8 | Дата и время создания ответа |

В таблице «quizzes» описана информация о викторинах, структура которой представлена в таблице 3.6.

Таблица 3.6

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор викторины |
| CreatorId | varchar | 255 | Уникальный идентификатор организатора из таблицы «aspnetusers» |
| CaregoryId | varchar | 255 | Уникальный идентификатор категории из таблицы «categories» |
| EventId | varchar | 255 | Уникальный идентификатор события из таблицы «events» |
| Name | longtext | 4 | Название викторины |
| Description | longtext | 4 | Описание викторины |
| Timer | int | 4 | Время на выполнение викторины (в минутах) |
| Password | longtext | 4 | Пароль викторины |
| CreatedOn | datetime | 8 | Дата и время создания викторины |

В таблице «events» описана информация о событиях, структура которой представлена в таблице 3.7.

Таблица 3.7

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор события |
| CreatorId | varchar | 255 | Уникальный идентификатор организатора из таблицы «aspnetusers» |
| Name | longtext | 4 | Название события |
| Status | int | 4 | Статус события |
| ActivationDateAndTime | datetime | 8 | Дата и время активации события |
| DurationOfActivity | time | 3 | Продолжительность события |
| CreatedOn | datetime | 8 | Дата и время создания события |

В таблице «scheduledjobs» хранится информация для смены статуса события, структура которой представлена в таблице 3.8.

Таблица 3.8

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор активации события |
| EventId | varchar | 255 | Уникальный идентификатор события из таблицы «events» |
| JobId | longtext | 4 | Уникальный идентификатор работы, запущенной для смены статуса события |
| IsActivationJob | tinyint | 1 | Индикатор активированного события |
| CreatedOn | datetime | 8 | Дата и время активации события |

В таблице «results» описана информация о результатах участников по пройденным викторинам, структура которой представлена в таблице 3.9.

Таблица 3.9

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор результата |
| QuizId | varchar | 255 | Уникальный идентификатор викторины из таблицы «quizzes» |
| StudentId | varchar | 255 | Уникальный идентификатор участника из таблицы «aspnetusers» |
| Points | decimal | 4 | Полученное количество баллов |
| MaxPoints | int | 4 | Максимальное количество баллов |
| TimeSpent | time | 3 | Время прохождения викторины |
| CreatedOn | datetime | 8 | Дата и время создания результата |

В таблице «groups» описана информация о группах для участников, структура которой представлена в таблице 3.10.

Таблица 3.10

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор группы |
| CreatorId | varchar | 255 | Уникальный идентификатор организатора из таблицы «aspnetusers» |
| Name | longtext | 4 | Название группы |
| CreatedOn | datetime | 8 | Дата и время создания группы |

В таблице «studentsgroups» описана информация об участниках в группах, структура которой представлена в таблице 3.11.

Таблица 3.11

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор участника в группе |
| StudentId | varchar | 255 | Уникальный идентификатор пользователя из таблицы «aspnetusers» |
| GroupId | varchar | 255 | Уникальный идентификатор группы из таблицы «groups» |
| CreatedOn | datetime | 8 | Дата и время создания участника в группе |

В таблице «eventsgroups» описана информация о событиях для групп, структура которой представлена в таблице 3.12.

Таблица 3.12

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | varchar | 255 | Уникальный идентификатор события для группы |
| EventId | varchar | 255 | Уникальный идентификатор события из таблицы «events» |
| GroupId | varchar | 255 | Уникальный идентификатор группы из таблицы «groups» |
| CreatedOn | datetime | 8 | Дата и время создания группы |

В таблице «aspnetuserroles» описана информация о ролях пользователей, структура которой представлена в таблице 3.13.

Таблица 3.13

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| UserId | varchar | 255 | Уникальный идентификатор пользователя из таблицы «aspnetusers» |
| RolesId | varchar | 255 | Уникальный идентификатор роли из таблицы «aspnetroles» |

Физическая схема базы данных представлена на рисунке 3.1.

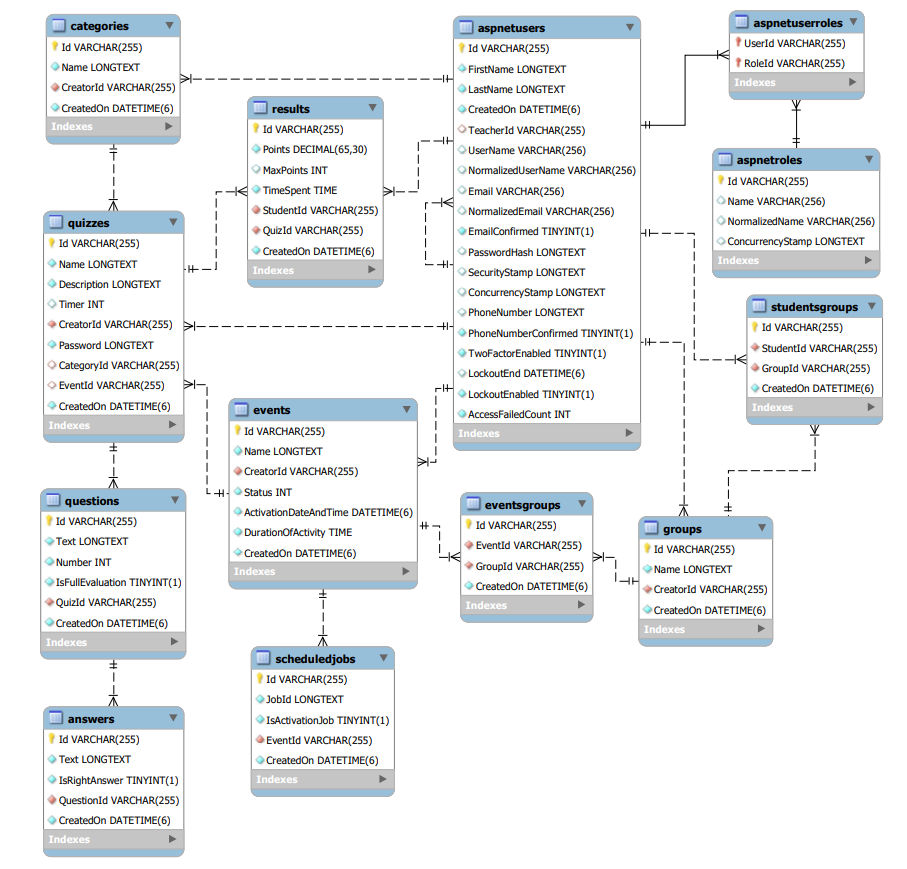


Рисунок 3.1

## Функции: логическая и физическая организация

В процессе разработки программного средства по автоматизации сбора и обработки данных для подготовки и проведения олимпиады было реализовано множество функций. Каждая из функций отвечает за определённые действия и выполняется в определённый момент времени.

Для регистрации нового пользователя в программе необходимо в окне авторизации нажать на компонент button под названием «Регистрация учащегося» или «Регистрация Организатора». После чего будет отображено окно регистрации, в котором нужно ввести необходимые данные в соответствующие поля и нажать компонент button под названием «Регистрация», после чего будет вызвана функция «RegisterAsync», код которой представлен ниже

public async Task<bool> RegisterAsync(ApplicationUser newUser, string password)

{

newUser.UserName = newUser.Email;

var result = await userManager.CreateAsync(newUser, password);

return result.Succeeded;

}

Для авторизации пользователя в программе необходимо в окне авторизации ввести необходимые данные в соответствующие поля и нажать компонент button под названием «Войти», после чего будет вызвана функция «LoginAsync», код которой представлен ниже

public async Task<bool> LoginAsync(string email, string password)

{

var user = await userManager.FindByEmailAsync(email);

if (user == null)

{

return false;

}

var isRightPassword = await userManager.CheckPasswordAsync(user, password);

if (isRightPassword)

{

var roles = await userManager.GetRolesAsync(user);

if (roles.Contains("Organizer"))

{

accountStore.CurrentUserRole = UserRole.Teacher;

}

else

{

accountStore.CurrentUserRole = UserRole.Student;

}

accountStore.CurrentUser = user;

return true;

}

return false;

}

Для создания викторины пользователю необходимо выбрать вкладку «Викторины» в главном окне программы. После отображения данных о викторинах необходимо нажать компонент button под названием «Новая викторина» для отображения окна создания викторины. В результате будет отображено окно добавления новых данных, в котором необходимо ввести данные в соответствующие поля и нажать компонент button под названием «Добавить викторину» для вызова функции «CreateQuizAsync», код которой представлен ниже

public async Task<string> CreateQuizAsync(string name, string description, int? timer, string creatorId, string password)

{

var quiz = new Quiz

{

Name = name,

Description = description,

Timer = timer,

CreatorId = creatorId,

Password = password

};

await quizRepository.AddAsync(quiz);

await quizRepository.SaveChangesAsync();

return quiz.Id;

}

Для создания события пользователю необходимо выбрать вкладку «События» в главном окне программы. После отображения данных о событиях необходимо нажать компонент button под названием «Новое событие» для отображения окна создания события. В результате будет окно, в котором нужно ввести необходимые данные в соответствующие поля и нажать компонент button под названием «Добавить событие» для вызова функции «CreateEventAsync», код которой представлен ниже

public async Task<string> CreateEventAsync(

string name,

string activationDate,

string activeFrom,

string activeTo,

string creatorId)

{

var activationDateAndTime = GetActivationDateAndTimeUtc(activationDate, activeFrom);

var durationOfActivity = GetDurationOfActivity(activationDate, activeFrom, activeTo);

var @event = new Event

{

Name = name,

Status = Status.Pending,

ActivationDateAndTime = activationDateAndTime,

DurationOfActivity = durationOfActivity,

CreatorId = creatorId,

};

await repository.AddAsync(@event);

await repository.SaveChangesAsync();

return @event.Id;

}

Для назначения нескольких тестов на одно событие и изменения его статуса пользователю необходимо выбрать вкладку «События» в главном окне программы. После отображения окна событий, необходимо нажать компонент button с иконкой шестеренки, чтобы открылось окно настроек события. В результате будет отображено окно, в котором нужно нажать компонент button под названием «Добавить» в шапке таблицы «Викторины». В открывшемся окне необходимо будет отметить викторины для назначения с помощью компонента checkbox и нажать на компонент button под названием «Назначить викторину» для вызова функции «AssignQuizzesToEventAsync», код которой представлен ниже

public decimal CalculateResult(IList<QuestionViewModel> originalQuizQuestions, IList<AttemptedQuizQuestionViewModel> attemptedQuizQuestions)

{

decimal totalPoints = 0;

foreach (var attemptedQuestion in attemptedQuizQuestions)

{

var originalQuestion = originalQuizQuestions.FirstOrDefault(q => q.Id == attemptedQuestion.Id);

if (originalQuestion != null)

{

decimal questionPoints = 0;

if (attemptedQuestion.IsFullEvaluation)

{

bool isFullEvaluationMatch = attemptedQuestion.Answers.All(a =>

originalQuestion.Answers.Any(oa => oa.Id == a.Id && oa.IsRightAnswer == a.IsRightAnswerAssumption));

if (isFullEvaluationMatch)

{

questionPoints = 1;

}

}

else

{

int correctAnswerCount = originalQuestion.Answers.Count(a => a.IsRightAnswer);

int selectedAnswerCount = attemptedQuestion.Answers.Count(a => a.IsRightAnswerAssumption);

int selectedCorrectAnswerCount = attemptedQuestion.Answers.Count(a =>

originalQuestion.Answers.Any(oa => oa.Id == a.Id && oa.IsRightAnswer == a.IsRightAnswerAssumption && oa.IsRightAnswer));

if (selectedCorrectAnswerCount > 0)

{

if (correctAnswerCount == selectedCorrectAnswerCount && selectedAnswerCount == selectedCorrectAnswerCount)

{

questionPoints = 1;

}

else

{

decimal partialPoints = 1m / originalQuestion.Answers.Count();

questionPoints = selectedCorrectAnswerCount \* partialPoints;

}

}

}

totalPoints += questionPoints;

}

}

return Math.Round(totalPoints, 2);

}

Полный код программы представлен в приложении А.

## Входные и выходные данные

В данном программном средстве входными являются следующие данные: информация о группах и участниках, о категориях викторин, о викторинах, о вопросах и ответах, о событиях.

Входными данными при добавлении участника организатором является электронная почта.

Входными данными при добавлении группы является название группы.

Входными данными при добавлении категории для викторин является название категории.

Входными данными при регистрации организатора или участника являются:

* электронная почта;
* имя
* фамилия;
* пароль.

Входными данными при добавлении викторины являются:

* название викторины;
* описание викторины;
* время на прохождение викторины;
* пароль для викторины.

Входными данными при добавлении вопроса являются:

* описание вопроса;
* тип проверки вопроса.

Входными данными при добавлении ответа являются:

* описание ответа;
* индикатор верного ответа.

Входными данными при добавлении события являются:

* название события;
* дата активации;
* время активации «С»;
* время активации «До».

Выходными данными являются следующие отчеты:

* отчет о текущих участниках, добавленных организатором, который представлен в приложении Б на рисунке Б.1;
* отчет о викторинах, содержащий соответствующие вопросы, который представлен в приложении Б на рисунке Б.2;
* отчет о категориях для викторин, который представлен в приложении Б на рисунке Б.3;
* отчет о группах участников, добавленных организатором, который представлен в приложении Б на рисунке Б.4;
* статистический отчет по количеству участников в каждом событии, который представлен в приложении Б на рисунке Б.5;
* статистический отчет, оценивающий успеваемость различных групп, который представлен в приложении Б на рисунке Б.6;
* статистический отчет, представляющий успеваемость по разным категориям, который представлен в приложении Б на рисунке Б.7;
* статистический отчет, определяющий наиболее успешных участников по каждой викторине, который представлен в приложении Б на рисунке Б.8;
* статистический отчет, отображающий процент выполнения тестов различными группами, который представлен в приложении Б на рисунке Б.9;
* статистический отчет, показывающий распределение успеваемости учащихся по различным категориям, который представлен в приложении Б на рисунке Б.10;
* статистический отчет, содержащий информацию о времени, затраченном каждым студентом на прохождение викторин, который представлен в приложении Б на рисунке Б.11;
* отчет по итогам прохождения викторин по разным событиям, который представлен в приложении Б на рисунке Б.12.

## Модульное тестирование

Модульное тестирование (Unit-testing) — уровень тестирования, на котором тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Модульное тестирование является важной составной частью отладочного тестирования, выполняемого разработчиками для отладки написанного ими кода.

Тестирование было проведено при помощи тестовой среды xUnit, которая по умолчанию включена в интегрированную среду разработки Microsoft Visual Studio и популярна среди разработчиков, которые ее используют.

Главное окно программы представлено в приложении Б на рисунке Б.15.

В таблице 3.14 представлено описание теста метода «RegisterAsync» класса «UserAccountService.cs» для регистрации нового пользователя, когда экземпляр пользователя валидный (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.13 и рисунке Б.14).

Таблица 3.14

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка регистрации валидного пользователя с валидным паролем |
| Название теста | RegisterAsync\_ValidUserAndPassword\_ReturnTrue |
| Входные данные | Новый экземпляр пользователя с валидными данными |
| Действие | Вызов метода «RegisterAsync» с входными данными |
| Ожидаемый результат | true |
| Фактический результат | true |
| Статус | Passed (Пройден) |

Код теста «RegisterAsync\_ValidUserAndPassword\_ReturnTrue» представлен ниже

[Fact]

public async Task RegisterAsync\_ValidUserAndPassword\_ReturnsTrue()

{

// Arrange

var newUser = new ApplicationUser();

var password = "password";

mockUserManager.Setup(x => x.CreateAsync(newUser, password))

.ReturnsAsync(IdentityResult.Success);

// Act

var result = await userAccountService.RegisterAsync(newUser, password);

// Assert

Assert.True(result);

}

В таблице 3.15 представлено описание теста метода «RegisterAsync» класса «UserAccountService.cs» для регистрации нового пользователя, когда экземпляр пользователя невалидный (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.13 и рисунке Б.14).

Таблица 3.15

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка регистрации невалидного пользователя с валидным паролем |
| Название теста | RegisterAsync\_InvalidUserAndPassword\_ReturnTrue |
| Входные данные | Новый экземпляр пользователя с валидными данными |
| Действие | Вызов метода «RegisterAsync» с входными данными |
| Ожидаемый результат | false |
| Фактический результат | false |
| Статус | Passed (Пройден) |

Код теста «RegisterAsync\_InvalidUserAndPassword\_ReturnTrue» представлен ниже

[Fact]

public async Task RegisterAsync\_InvalidUserAndPassword\_ReturnsFalse()

{

// Arrange

var newUser = new ApplicationUser();

var password = "password";

mockUserManager.Setup(x => x.CreateAsync(newUser, password))

.ReturnsAsync(IdentityResult.Failed());

// Act

var result = await userAccountService.RegisterAsync(newUser, password);

// Assert

Assert.False(result);

}

В таблице 3.16 представлено описание теста метода «CreateGroupAsync» класса «GroupsService.cs» для создания новой группы с валидными данными (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.16, рисунке Б.17 и рисунке Б.18).

Таблица 3.16

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка создания группы с валидными данными |
| Название теста | CreateGroupAsync\_ShouldCreateCorrectly |
| Входные данные | * название группы: «Test Group»; * идентификатор создателя: случайно сгенерированный уникальный идентификатор. |
| Действие | Вызов метода CreateGroupAsync с названием группы и идентификатором создателя |

Продолжение таблицы 3.16

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Ожидаемый результат | * группа не принимает пустое значение; * имя группы совпадает с «Test Group»; * идентификатор создателя совпадает со сгенерированным идентификатором создателя; * количество групп в базе данных равно одному. |
| Фактический результат | * группа не принимает пустое значение; * имя группы совпадает с «Test Group»; * идентификатор создателя совпадает со сгенерированным идентификатором создателя; * количество групп в базе данных равно одному. |
| Статус | Passed (Пройден) |

Код теста «CreateGroupAsync\_ShouldCreateCorrectly» представлен ниже

[Fact]

public async Task CreateGroupAsync\_ShouldCreateCorrectly()

{

//Arrange

var creatorId = Guid.NewGuid().ToString();

//Act

await Service.CreateGroupAsync("Test Group", creatorId);

//Assert

var group = await DbContext.Groups.FirstOrDefaultAsync();

Assert.NotNull(group);

Assert.Equal("Test Group", group.Name);

Assert.Equal(creatorId, group.CreatorId);

Assert.Equal(1, DbContext.Groups.Count());

}

В таблице 3.17 представлено описание теста метода «CreateEventAsync» класса «EventsService.cs» для создания нового события с валидными данными (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.19, рисунке Б.20 и рисунке Б.21).

Таблица 3.17

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка создания события с валидными данными |
| Название теста | CreateEventAsync\_ShouldCreateEventCorrectly |
| Входные данные | * название события: «Test Event» * дата активации события «01/04/2020» * доступен с «08:00» * доступен до «10:00» * идентификатор создателя: случайно сгенерированный уникальный идентификатор |

Продолжение таблицы 3.17

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Действие | Вызов метода «CreateEventAsync» с ожидаемым названием события, датой активации, диапазоном активного времени и идентификатором создателя |
| Ожидаемый результат | * событие не является пустым; * имя группы совпадает с «TestEvent»; * дата активации события совпадает с ожидаемой датой активации события; * продолжительность события соответствует ожидаемой продолжительности события; * статус события установлен в «Pending»; * идентификатор создателя совпадает со сгенерированным идентификатором создателя. |
| Фактический результат | * событие не является пустым; * имя группы совпадает с «TestEvent»; * дата активации события совпадает с ожидаемой датой активации события; * продолжительность события соответствует ожидаемой продолжительности события; * статус события установлен в «Pending»; * идентификатор создателя совпадает со сгенерированным идентификатором создателя. |
| Статус | Passed (Пройден) |

Код теста «CreateEventAsync\_ShouldCreateEventCorrectly» представлен ниже

[Fact]

public async Task CreateEventAsync\_ShouldCreateEventCorrectly()

{

//Arrange

var creatorId = Guid.NewGuid().ToString();

var expectedEventName = "Test Event";

var eventActivationDate = "01/04/2020";

var activeFrom = "08:00";

var activeTo = "10:00";

var expectedEventActivationDate = new DateTime(2020, 4, 1, 5, 00, 00);

var expectedEventDuration = new TimeSpan(2, 0, 0);

//Act

var eventId = await Service.CreateEventAsync(

expectedEventName,

eventActivationDate,

activeFrom,

activeTo,

creatorId);

//Assert

var @event = await DbContext.Events.FirstOrDefaultAsync(x => x.Id == eventId);

Assert.NotNull(@event);

Assert.Equal(expectedEventName, @event.Name);

Assert.Equal(expectedEventActivationDate, @event.ActivationDateAndTime);

Assert.Equal(expectedEventDuration, @event.DurationOfActivity);

Assert.Equal(Status.Pending, @event.Status);

Assert.Equal(creatorId, @event.CreatorId);

}

В таблице 3.18 представлено описание теста метода «AssignQuizzesToEventAsync» класса «EventsService.cs» для назначения викторины на событие (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.22, рисунке Б.23, рисунке Б.24 и рисунке Б.25).

Таблица 3.18

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка правильности назначения викторины на событие |
| Название теста | AssigQuizToEventAsync\_ShouldSetQuizToEventCorrectly |
| Входные данные | * название викторины: сгенерированное название викторины; * название события: «Event»; * статус события: «Pending»; * дата и время активации события: текущее время UTC; * продолжительность: 30 минут; * идентификатор создателя: сгенерированный уникальный идентификатор. |
| Действие | Вызов метода «AssignQuizzesToEventAsync» с идентификатором викторины и идентификатором события |
| Ожидаемый результат | Викторина назначена на событие |
| Фактический результат | Викторина назначена на событие |
| Статус | Passed (Пройден) |

Код теста «AssigQuizToEventAsync\_ShouldSetQuizToEventCorrectly» представлен ниже

[Fact]

public async Task AssigQuizToEventAsync\_ShouldSetQuizToEventCorrectly()

{

//Arrange

var creatorId = Guid.NewGuid().ToString();

var quiz = await CreateQuizAsync();

var @event = new Event

{

Name = "Event",

Status = Status.Pending,

ActivationDateAndTime = DateTime.UtcNow,

DurationOfActivity = TimeSpan.FromMinutes(30),

CreatorId = creatorId,

};

await DbContext.Events.AddAsync(@event);

await DbContext.SaveChangesAsync();

DbContext.Entry(@event).State = EntityState.Detached;

//Act

await Service.AssignQuizzesToEventAsync(new List<string>() { quiz.Id }, @event.Id);

//Assert

var eventWithAssignedQuiz = await DbContext.Events.FirstOrDefaultAsync(x => x.Id == @event.Id);

Assert.NotEmpty(eventWithAssignedQuiz.Quizzes);

}

В таблице 3.19 представлено описание теста метода «CalculateResult» класса «ResultHelper.cs» для вычисления результата тестирования в случае, когда даны все верные ответы при его прохождении и включена полная оценка (если допущена одна ошибка, весь вопрос не засчитывается)(визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.26).

Таблица 3.19

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка вычисления результата тестирования, когда все ответы верны и включена полная оценка |
| Название теста | CalculateResult\_FullEvaluation\_AllAnswersCorrect\_Returns1 |
| Входные данные | * вопросы викторины с правильными ответами; * вопросы викторины с правильными предположениями. |
| Действие | Вызов метода «CalculateResult» с входными данными |
| Ожидаемый результат | Результат равен единице |
| Фактический результат | Результат равен единице |
| Статус | Passed (Пройден) |

Код теста «CalculateResult\_FullEvaluation\_AllAnswersCorrect\_Returns1» представлен ниже

[Fact]

public void CalculateResult\_FullEvaluation\_AllAnswersCorrect\_Returns1()

{

// Arrange

var originalQuizQuestions = new List<QuestionViewModel>

{

new QuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

new AnswerViewModel { Id = "1", IsRightAnswer = true },

new AnswerViewModel { Id = "2", IsRightAnswer = true }

}

}

};

var attemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

new AttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = true,

Answers = new List<AttemptedQuizAnswerViewModel>

{

new AttemptedQuizAnswerViewModel { Id = "1", IsRightAnswerAssumption = true },

new AttemptedQuizAnswerViewModel { Id = "2", IsRightAnswerAssumption = true }

}

}

};

// Act

decimal result = \_resultHelper.CalculateResult(originalQuizQuestions, attemptedQuizQuestions);

// Assert

Assert.Equal(1, result);

}

В таблице 3.20 представлено описание теста метода «CalculateResult» класса «ResultHelper.cs» для вычисления результата тестирования в случае, когда даны не все верные ответы при его прохождении и включена полная оценка(визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.26).

Таблица 3.20

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка вычисления результата тестирования, когда не все ответы верны и включена полная оценка |
| Название теста | CalculateResult\_FullEvaluation\_AllAnswersCorrect\_Returns0 |
| Входные данные | * оригинальные вопросы викторины с правильными ответами; * вопросы викторины с неверными предположениями. |
| Действие | Вызов метода «CalculateResult» с входными объектами и данными |
| Ожидаемый результат | Результат равен нолю |
| Фактический результат | Результат равен нолю |
| Статус | Passed (Пройден) |

Код теста «CalculateResult\_FullEvaluation\_NotAllAnswersCorrect\_ Returns0» представлен ниже

public void CalculateResult\_FullEvaluation\_NotAllAnswersCorrect\_Returns0()

{

// Arrange

var originalQuizQuestions = new List<QuestionViewModel>

{

new QuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

new AnswerViewModel { Id = "1", IsRightAnswer = true },

new AnswerViewModel { Id = "2", IsRightAnswer = true }

}

}

};

var attemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

new AttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = true,

Answers = new List<AttemptedQuizAnswerViewModel>

{

new AttemptedQuizAnswerViewModel { Id = "1", IsRightAnswerAssumption = true },

new AttemptedQuizAnswerViewModel { Id = "2", IsRightAnswerAssumption = false }

}

}

};

// Act

decimal result = \_resultHelper.CalculateResult(originalQuizQuestions, attemptedQuizQuestions);

// Assert

Assert.Equal(0, result);

}

В таблице 3.21 представлено описание теста метода «CalculateResult» класса «ResultHelper.cs» для вычисления результата, когда все ответы верны и включена частичная оценка (если допущена одна ошибка, то за каждый правильный ответ будет начислено количество баллов) (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.26).

Таблица 3.21

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка вычисления результата тестирования, когда не все ответы верны и включена частичная оценка |
| Название теста | CalculateResult\_PartialEvaluation\_AllAnswersCorrect\_  Returns1 |
| Входные данные | * оригинальные вопросы викторины с правильными ответами; * вопросы викторины с правильными предположениями. |
| Действие | Вызов метода «CalculateResult» с входными объектами и данными. |
| Ожидаемый результат | Результат равен единице |
| Фактический результат | Результат равен единице |

Продолжение таблицы 3.21

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Статус | Passed (Пройден) |

Код теста «CalculateResult\_PartialEvaluation\_AllAnswersCorrect\_Returns1»представлен ниже

[Fact]

public void CalculateResult\_PartialEvaluation\_AllAnswersCorrect\_Returns1()

{

// Arrange

var originalQuizQuestions = new List<QuestionViewModel>

{

new QuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

new AnswerViewModel { Id = "1", IsRightAnswer = true },

new AnswerViewModel { Id = "2", IsRightAnswer = true }

}

}

};

var attemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

new AttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = false,

Answers = new List<AttemptedQuizAnswerViewModel>

{

new AttemptedQuizAnswerViewModel { Id = "1", IsRightAnswerAssumption = true },

new AttemptedQuizAnswerViewModel { Id = "2", IsRightAnswerAssumption = true }

}

}

};

// Act

decimal result = \_resultHelper.CalculateResult(originalQuizQuestions, attemptedQuizQuestions);

// Assert

Assert.Equal(1, result);

}

В таблице 3.22 представлено описание теста метода «CalculateResult» класса «ResultHelper.cs» для вычисления результата, когда некоторые ответы являются правильными и включена частичная оценка(визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.26).

Таблица 3.22

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка вычисления результата, когда некоторые ответы являются правильными и включена частичная оценка |

Продолжение таблицы 3.22

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Название теста | CalculateResult\_PartialEvaluation\_SomeAnswersCorrect\_ReturnsPartialPoints |
| Входные данные | * оригинальные вопросы викторины с правильными ответами; * вопросы викторины с неверными предположениями |
| Действие | Вызов метода «CalculateResult» с входными объектами и данными |
| Ожидаемый результат | Результат равен 0.5 |
| Фактический результат | Результат равен 0.5 |
| Статус | Passed (Пройден) |

Код теста «CalculateResult\_PartialEvaluation\_SomeAnswersCorrect\_ ReturnsPartialPoints» представлен ниже

[Fact]

public void CalculateResult\_PartialEvaluation\_SomeAnswersCorrect\_ReturnsPartialPoints()

{

// Arrange

var originalQuizQuestions = new List<QuestionViewModel>

{

new QuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

new AnswerViewModel { Id = "1", IsRightAnswer = true },

new AnswerViewModel { Id = "2", IsRightAnswer = true }

}

}

};

var attemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

new AttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = false,

Answers = new List<AttemptedQuizAnswerViewModel>

{

new AttemptedQuizAnswerViewModel { Id = "1", IsRightAnswerAssumption = true },

new AttemptedQuizAnswerViewModel { Id = "2", IsRightAnswerAssumption = false }

}

}

};

// Act

decimal result = \_resultHelper.CalculateResult(originalQuizQuestions, attemptedQuizQuestions);

// Assert

Assert.Equal(0.5m, result);

}

На рисунке 3.2 представлены результаты выполнения всех тестов для упомянутых выше классов. Всего было разработано 76 модульных тестов.

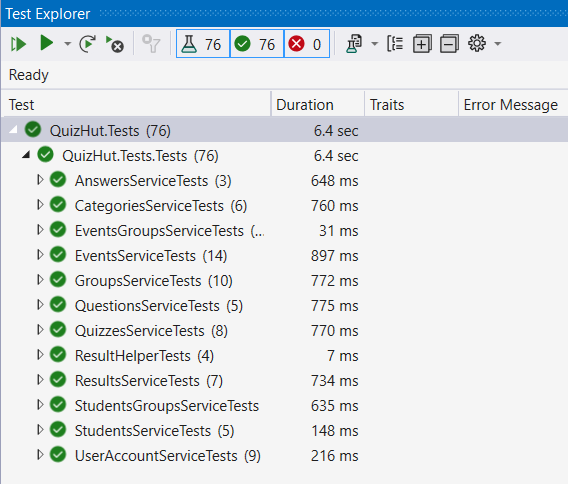


Рисунок 3.2

## Описание справочной системы

Для полноценной работы пользователя с программой необходимо предоставить ему надежную справочную систему, которая содержит информацию о методах и процедурах работы с программой, а также объясняет, что происходит при нажатии определенных кнопок или выборе пунктов меню.

Пользователь должен иметь доступ к справочной информации на протяжении всего процесса использования программы. Для этого пользователь может нажать кнопку «Справка» в левом нижнем углу главного окна программы. Справочная система данного проекта представляет собой отдельное приложение с расширением «\*.chm». Разработка этой справочной системы осуществлялась с использованием программы Dr.Explain [12].

Справочная система представляет собой описание возможных действий для работы с программой и состоит и следующих разделов:

* «Начало работы».
* «Работа с программой в роли «Организатор».
* «Работа с программой в роли «Участник».

Структура справочной системы представлена на рисунке 3.3.

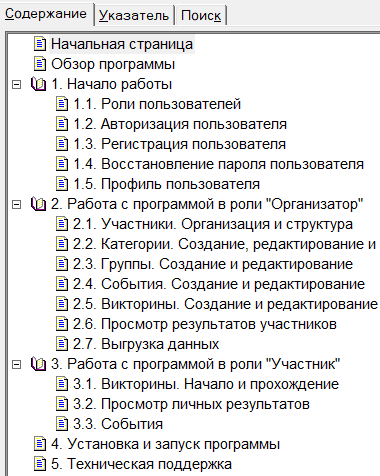


Рисунок 3.3

В целом, справочная система является важным компонентом любого программного средства, цель которого – предоставить пользователям понятный и доступный источник информации о функциях и возможностях программы.

1. Применение

## Назначение программного средства

Программное средство «QuizHut.exe» разработано специально для учебных заведений с целью оптимизации сбора и обработки данных для подготовки и проведения олимпиад по программированию. «QuizHut.exe» автоматизирует такие ключевые задачи как:

* регистрация участников олимпиады;
* публикация заданий;
* назначение тестирования и заданий участникам;
* анализ информации по результатам проведения олимпиады;
* выгрузка необходимой информации.

Благодаря интеграции функционала регистрации и тестирования программное средство обеспечит бесперебойную передачу данных между различными этапами проведения олимпиад, устраняя необходимость ручной обработки данных и значительно упрощая работу организаторов.

В программном средстве «QuizHut.exe» предусмотрены различные режимы работы для организаторов и участников. В режиме организатора предусмотрены возможность получения полной информацию об участниках, загрузка/выгрузка данных, связанных с результатами проведения олимпиады, публикация заданий для участников, назначение тестирования/задания соответствующим участникам и получение результата работы в указанные сроки. Такая возможность комплексного управления и обработки данных позволяет организаторам оптимизировать рабочий процесс и обеспечить бесперебойное проведение олимпиады повышая этим общую организацию и эффективность олимпиад по программированию.

## Условия применения

Программное средство «QuizHut.exe» предназначено для работы на компьютере под управлением операционной системы Windows 10 с установленной системой управления базами данных MySQL.

Инструкции, описанные ниже, гарантируют успешный запуск программы и обеспечат правильное ее функционирование:

* убедиться, что на компьютере установлена операционная система Windows 10;
* установить систему управления базами данных (СУБД) MySQL с официального сайта на компьютер;
* открыть командную строку (cmd), нажав клавиши Windows плюс R;
* ввести значение «cmd» и нажать клавишу«Enter»;
* выполнить команду «setx QH\_DATABASE\_CONNECTION\_STRING «строка\_подключения», заменив значение параметра «строка\_подключения» на фактическую строку подключения базы данных MySQL;
* выполнить команду «setx QH\_SENDGRID\_API\_KEY «апи\_ключ», заменив значение параметра «апи\_ключ» на ключ API, предоставленный SendGrid для доставки электронной почты;
* выполнить команду «setx QH\_SENDGRID\_FROM\_EMAIL [quizhutkbip@gmail.com»](mailto:quizhutkbip@gmail.com), чтобы программа могла отправлять электронные письма;
* запустить «setup.exe» файл и следовать указаниям установщика.

1. Охрана труда и окружающей среды

## Правовые, нормативные, социально-экономические и организационные вопросы охраны труда

Функционирование системы управления охраной труда (СУОТ) обеспечено непосредственно директором организации. Обязанности по охране труда (ОТ) грамотно распределены между работниками. Проводится периодичное усовершенствований работы системы охраны труда [].

Управление охраной труда в ООО «МОДСЕН» осуществляет директор, в структурных подразделениях - руководители структурных подразделений.

Полномочия и ответственность руководителей структурных подразделений организации по осуществлению контроля за соблюдением законодательства об охране труда определяются положениями об отделе.

В организации «МОДСЕН» ответственный за охрану труда занимает должность заведующего по хозяйству. Раз в три года он должен проходить проверку знаний в комиссии по вопросам охраны труда Минского городского исполнительного комитета.

Функции ответственного за охрану труда в организации:

* выявление опасных и вредных производственных факторов на рабочих местах;
* проведение анализа состояния условий и охраны труда, причин нарушения законодательства о труде и охране труда, производственного травматизма, профессиональной и профессионально обусловленной заболеваемости;
* оказание помощи подразделениям предприятия в организации и проведении замеров параметров опасных и вредных производственных факторов, аттестации и сертификации рабочих мест и производственного оборудования на соответствие требованиям охраны труда;
* информирование работающих от лица работодателя о состоянии условий труда на рабочих местах, о причинах возникновения профессиональных заболеваний и мероприятиях по их предупреждению, о принятых мерах по защите от опасных и вредных производственных факторов;
* участие в подготовке документов на выплату возмещения вреда, причиненного здоровью работающих в результате несчастного случая на производстве или профессионального заболевания;
* проведение проверок, обследований (или участие в проверках, обследованиях) технического состояния зданий, сооружений, оборудования, машин и механизмов на соответствие их нормативным правовым актам по охране труда, эффективности работы вентиляционных систем, состояния санитарно-технических устройств, санитарно-бытовых помещений, средств коллективной и индивидуальной защиты;
* разработка совместно с руководителями подразделений и другими службами предприятия мероприятий по предупреждению несчастных случаев на производстве и профессиональных заболеваний, по улучшению условий труда и доведению их до требований нормативных правовых актов по охране труда, а также оказание организационной помощи по выполнению запланированных мероприятий;
* участие в составлении раздела «Охрана труда» коллективного договора, соглашения по охране труда предприятия;
* участие в работе комиссии по приемке в эксплуатацию законченных строительством или реконструированных объектов производственного назначения, по приемке из ремонта установок, агрегатов, станков и другого оборудования;
* составление (при участии руководителей подразделений и соответствующих служб предприятия) перечней профессий и видов работ, на которые должны быть разработаны инструкции по охране труда;
* оказание методической помощи руководителям подразделений предприятия при разработке и пересмотре инструкций по охране труда для работников, стандартов предприятия по безопасности труда;
* разработка программы и проведение вводного инструктажа по охране труда со всеми вновь принимаемыми на работу, командированными, учащимися и студентами, прибывшими на производственную практику или обучение;
* участие в работе комиссий по проверке знаний по охране труда у работников предприятия;
* составление отчетности по охране труда по установленным формам и в соответствующие сроки и др.

Порядок подготовки, переподготовки, стажировки, инструктажа, повышения квалификации и проверки знаний, работающих по вопросам охраны труда установлен инструкцией о порядке подготовки, переподготовки, стажировки, инструктажа, повышения квалификации и проверки знаний, работающих по вопросам охраны труда, утвержденной постановлением Министерства труда и социальной защиты Республики [5]. Обучение охране труда проводят руководители структурных подразделений в течении первого дня после приема на практику.

По характеру и времени проведения инструктажи по охране труда подразделяются на:

* вводный;
* первичный на рабочем месте;
* повторный;
* внеплановый;
* целевой.

Порядок расследования несчастных случаев на производстве определен Правилами расследования несчастных случаев на производстве и профессиональных заболеваний, утвержденными постановлением Совета Министров Республики Беларусь от 15 января 2004 года № 30.

Целями расследования несчастных случаев являются:

* установление причин, вызвавших несчастный случай;
* установление виновного;
* разработка плана мероприятий, направленных на предотвращение подобных случаев.

Основными видами контроля являются:

* контроль за соблюдением законодательства об охране труда, осуществляемый руководителями и специалистами организации в соответствии с их должностными обязанностями;
* общественный контроль за соблюдением законодательства об охране труда, осуществляемый профсоюзами;
* периодический контроль за соблюдением законодательства об охране труда, осуществляемый представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда (уполномоченных лиц по охране труда работников);

Периодический контроль функционирования СУОТ проводится нескольких видов, а именно:

* ежедневный – осуществляется в структурных подразделениях их руководителями с участием общественного инспектора профсоюза по охране труда;
* ежемесячный – осуществляется в отделах главой отдела с участием общественного инспектора профсоюза по охране труда, руководителей отдела и ответственных за охрану труда;
* ежеквартальный – осуществляется в организации в целом руководителем организации с участием руководителей служб общественного инспектора профсоюза по охране труда;

Периодический контроль за соблюдением законодательства об охране труда проводится ежедневно, ежемесячно, ежеквартально с участием общественного инспектора и предусматривает участие работников в деятельности по улучшению условий и охраны труда, профилактике несчастных случаев и заболеваний на производстве [5].

Периодический контроль осуществляется представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда. Правовые, нормативные, социально-экономические и организационные вопросы охраны труда.

## Разработка мер по нормализации и защиты работающих от статического электричества при эксплуатации ПЭВМ

Широкое использование во всех областях хозяйственной деятельности диэлектрических материалов и органических соединений (полимеров, бумаги, твердых и жидких углеводородов, нефтепродуктов и тому подобное) неизбежно сопровождается образованием зарядов статического электричества, которые не только осложняют проведение технологических процессов, но и зачастую становятся причиной пожаров и взрывов, приносящих большой материальный ущерб. Нередко это приводит к гибели людей.

Статическое электричество — это совокупность явлений, связанных с возникновением, сохранением и релаксацией свободного электрического заряда на поверхности, или в объеме диэлектриков, или на изолированных проводниках [3]. Образование и накопление зарядов на перерабатываемом материале связано с двумя следующими условиями:

наличие контакта поверхностей, в результате чего создается двойной электрический слой, возникновение которого связано с переходом электронов в элементарных донорско-акцепторных актах на поверхности контакта. Знак заряда определяет неодинаковое сродство материала поверхностей к электрону;

хотя бы одна из контактирующих поверхностей должна быть из диэлектрического материала.

Заряды будут оставаться на поверхностях после прекращения контакта только в том случае, если время разрушения контакта меньше времени релаксации зарядов. Последнее в значительной степени определяет величину зарядов на разделенных поверхностях.

Основная величина, характеризующая способность к электризации, - это удельное электрическое сопротивление (Р) поверхностей контактирующих материалов. Если они имеют низкое сопротивление, то при разделении заряды с них стекают, и эти поверхности несут незначительный заряд. Если же сопротивление материалов высокое или скорость отрыва поверхностей велика, то заряды будут сохраняться. Способность веществ электризоваться также характеризуется удельной электропроводимостью , или удельным объемным сопротивлением Рv (при чем обратно пропорционально Рv).

Условно принято, что при удельном электрическом сопротивлении материалов менее 105Ом­ заряды не сохраняются и материалы не электризуются.

В отдельных случаях склонность к электризации плоских полимерных материалов целесообразно оценивать по величине удельного поверхностного электрического сопротивления Рs, Ом. Большинство полимерных пленок и материалов не электризуется, если Рs меньше 1011Ом.

В соответствии с Правилами защиты от статического электричества в производствах химической, нефтехимической и нефтеперерабатывающей промышленности все вещества и материалы в зависимости от удельного объемного сопротивления подразделяются на диэлектрические (Рv больше 108Ом), антистатические (Рvот105 до 108Oмм) и электропроводящие (Рvменьше105Oмм).

Основными факторами, влияющими на электризацию веществ, являются их электрофизические свойства и скорость разделения поверхностей. Экспериментально установлено, что чем интенсивнее осуществляется процесс, т.е. чем выше скорость отрыва, тем больший заряд остается на поверхности.

Известны следующие пути заряжения объектов: непосредственное контактирование с наэлектризованными материалами, индуктивное и смешанное заряжение.

К чисто контактному заряжению поверхностей относится, например, электризация при перекачивании углеводородного топлива, растворителей по трубопроводам. Известно, что трубопроводы из прозрачного диэлектрического материала при перекачивании жидкостей даже светятся.

Наряду с контактным часто происходит индуктивное заряжение проводящих объектов и обслуживающего персонала в электрическом поле движущегося плоского наэлектризованного материала.

Смешанное заряжение наблюдается тогда, когда наэлектризованный материал поступает в какие-либо емкости, изолированные от земли. Этот вид заряжения наиболее часто встречается при заливке горючих жидкостей в емкости, при подаче резиновых клеев, тканей, пленок в передвижные емкости, тележки и т.д. Образование зарядов статического электричества при контакте жидкого тела с твердым или одного твердого тела с другим во многом зависит от плотности соприкосновения трущихся поверхностей, их физического состояния, скорости и коэффициента трения, давления в зоне контакта, микроклимата окружающей среды, наличия внешних электрических полей и так далее.

Принципиальная схема электризации твердых материалов при их разделении представлена на рисунке 5.1



Рисунок 5.1

Заряды статического электричества могут накапливаться и на теле человека (при работе или контакте с наэлектризованными материалами и изделиями). Высокое поверхностное сопротивление тканей человека затрудняет стекание зарядов, и человек может длительное время находиться под большим потенциалом.

Основной опасностью при электризации различных материалов является возможность возникновения искрового разряда как с диэлектрической наэлектризованной поверхности, так и с изолированного проводящего объекта.

Разряд статического электричества возникает, если напряженность электростатического поля над поверхностью диэлектрика или проводника, обусловленная накоплением на них зарядов, достигает критической (пробойной) величины. Для воздуха эта величина составляет примерно 30 кВ/м.

Воспламенение горючих смесей искровыми разрядами статического электричества может произойти в том случае, если выделяющаяся в разряде энергия будет выше минимальной энергии зажигания горючей смеси.

Наряду с пожарной опасностью статическое электричество представляет опасность и для работающих.

Легкие «уколы» при работе с сильно наэлектризованными материалами вредно влияют на психику работающих и в определенных ситуациях могут способствовать травмам на технологическом оборудовании. Сильные искровые разряды, возникающие, например, при затаривании гранулированных материалов, могут приводить к болевым ощущениям. Неприятные ощущения, вызываемые статическим электричеством, могут явиться причинами развития неврастении, головной боли, плохого сна, раздражительности, покалываний в области сердца и так далее. Кроме того, при постоянном прохождении через тело человека малых токов электризации возможны неблагоприятные физиологические изменения в организме, приводящие к профессиональным заболеваниям. Систематическое воздействие электростатического поля повышенной напряженности может вызывать функциональные изменения центральной нервной, сердечно-сосудистой и других систем организма [5].

Использование для одежды искусственных или синтетических тканей приводит также к накоплению зарядов статического электричества на человеке. В ГОСТ 29191 (МЭК 801-2-91) приводятся сведения о том, что синтетические ткани могут заряжаться до потенциала, равного 15 кВ. Поэтому ток, протекающий через тело человека, одетого в костюм или халат из синтетической ткани, может достигать 3 мкА. Прикосновение к заземленным участкам рабочего места или к незаряженному телу вызывает искровой разряд с силой тока до 30 А.

Статическое электричество сильно влияет также на ход технологических процессов получения и переработки материалов и качество продукции. При больших плотностях заряда может возникать электрический пробой тонких полимерных пленок электро- и радиотехнического назначения, что приводит к браку выпускаемой продукции. Особенно большой ущерб наносит вызванное электростатическим притяжением налипание пыли на полимерные пленки.

Электризация затрудняет такие процессы, как просеивание, сушку, пневмотранспорт, печатание, транспортировку полимеров, диэлектрических жидкостей, формование синтетических волокон, пленок и тому подобное, автоматическое дозирование мелкодисперсных материалов, поскольку они прилипают к стенкам технологического оборудования и слипаются между собой.

Допустимые уровни напряженности электростатических полей устанавливаются ГОСТ 12.1.045 и СанПиН 11-16-94.

Предельно допустимые уровни (ПДУ) напряженности электростатического поля устанавливаются в зависимости от времени пребывания персонала на рабочих местах и не должны превышать:

* при воздействии до 1 ч ­ 60 кВ/м;
* при воздействии от 1 до 9 ч ­ ПДУ определяется по формуле (5.1)

, (5.1)

где Т ­ время, час.

При напряженности электростатического поля менее 20 кВ/м время пребывания в электростатическом поле не регламентируется.

Обобщенная схема методов защиты от воздействия статического электричества приведена на рисунке 5.2.

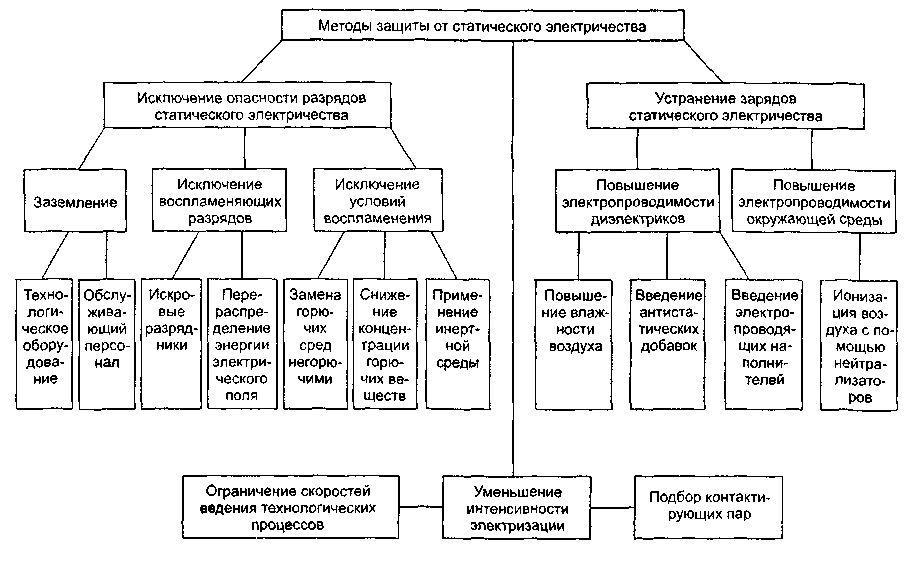


Рисунок 5.2.

Средства защиты от статического электричества должны применяться во всех взрывоопасных и пожароопасных помещениях и зонах открытых установок, отнесенных по классификации ПУЭ к классам B-I, B-Ia, B-I6, В-1г, В-П, В-Па, П-I, П-П [2].

При организации производства следует избегать процессов, сопровождающихся интенсивной генерацией зарядов статического электричества. Для этого необходимо правильно подбирать поверхности трения и скорости движения веществ, материалов, устройств, избегать процессов разбрызгивания, дробления, распыления, очищать горючие газы и жидкости от примесей и так далее.

Статическое электричество представляет собой значительный риск в различных отраслях промышленности, включая сектор информационных технологий. Для таких компаний, как ООО «МОДСЕН», очень важно принимать соответствующие меры по предотвращению электростатических разрядов, которые могут повредить чувствительные электронные компоненты и нарушить работу. Осознавая это, ООО «МОДСЕН» разработало комплексную программу безопасности от статического электричества, которая соответствует лучшим отраслевым практикам и нормативным требованиям.

Для предупреждения возможности накопления статического электричества на поверхностях оборудования, перерабатываемых материалов, а также на теле работающих выше минимальной энергии зажигания горючих смесей на предприятия обеспечено стекание возникающих зарядов с заряженных объектов.

В соответствии с ГОСТ 12.4.124 в ООО «МОДСЕН» это достигается использованием средств коллективной и индивидуальной защиты.

Средства коллективной защиты от статического электричества по принципу действия делятся на следующие виды: заземляющие устройства, нейтрализаторы, увлажняющие устройства, антиэлектростатические вещества, экранирующие устройства.

Одним из основополагающих понятий в области безопасности статического электричества является заземление. ООО «МОДСЕН» гарантирует, что все нетоковедущие металлические части, оборудование и рабочие места эффективно заземлены. Заземление служит важнейшим методом защиты, обеспечивая контролируемый путь для безвредного отвода статических зарядов в землю. Устанавливая преднамеренное электрическое соединение с землей или ее эквивалентом, ООО «МОДСЕН» предотвращает накопление статических зарядов и минимизирует возможность возникновения искровых разрядов.

В дополнение к заземлению ООО «МОДСЕН» использует передовые методы нейтрализации для дальнейшего снижения рисков, связанных со статическим электричеством. Они используют такие технологии, как коронные разряды, радиоизотопные нейтрализаторы и системы ионизированного воздушного потока для нейтрализации статических зарядов и поддержания сбалансированной электростатической среды. Эти методы эффективно противодействуют накоплению зарядов на поверхностях, снижая вероятность возникновения электростатических разрядов.

К индивидуальным средствам защиты от статического электричества относятся специальные электростатические обувь и одежда [5]. Для изготовления такой одежды должны применяться материалы с удельным поверхностным электрическим сопротивлением не более 107 Ом, а электрическое сопротивление между токопроводящим элементом антиэлектростатической одежды и землей должно быть от 106 до 108 Ом. Электрическое сопротивление между подпятником и ходовой стороной подошвы обуви должно быть от 106 до 108 Ом.

Обучение и информированность сотрудников играют решающую роль в обеспечении безопасности от статического электричества, и ООО «МОДСЕН» придает этим аспектам первостепенное значение. Они обеспечивают всестороннее обучение всех сотрудников, гарантируя, что они хорошо осведомлены о рисках и опасностях, связанных со статическим электричеством. Сотрудники изучают надлежащие процедуры обращения с электрифицированными материалами и оборудованием, распознавания потенциальных опасностей, связанных со статическим электричеством, и надлежащего реагирования в случае возникновения чрезвычайных ситуаций. Вооружая своих сотрудников необходимыми знаниями и навыками, ООО «МОДСЕН» способствует развитию культуры безопасности и обеспечивает активное участие каждого в предотвращении инцидентов, связанных со статическим электричеством.

Чтобы продемонстрировать свою приверженность безопасности статического электричества, ООО «МОДСЕН» активно отслеживает и соблюдает соответствующие нормативные акты и отраслевые стандарты. Они строго следуют национальным и международным рекомендациям, таким как Правила защиты от статического электричества на производственных объектах химической, нефтехимической и нефтеперерабатывающей промышленности (ГОСТ 12.4.124) в Республике Беларусь. Приводя свою практику в соответствие с этими правилами, ООО «МОДСЕН» обеспечивает соответствие или превышение необходимых требований по безопасности от статического электричества.

ООО «МОДСЕН» также ведет документированный учет мер по обеспечению безопасности статического электричества, оценки рисков и программ обучения. Эти записи служат доказательством их приверженности соблюдению требований и могут быть представлены для внутренних аудитов или инспекций регулирующих органов. Тщательно документируя свои системы заземления, методы нейтрализации и графики технического обслуживания, ООО «МОДСЕН» демонстрирует свою приверженность поддержанию безопасной рабочей среды и снижению рисков, связанных со статическим электричеством [4].

В заключение, ООО «МОДСЕН», IT-компания, подчеркивает важность безопасности статического электричества и строго придерживается правил и документов, регулирующих эту область. Благодаря таким мерам, как заземление, передовые методы нейтрализации, обучение сотрудников и соблюдение отраслевых стандартов, ООО «МОДСЕН» эффективно снижает риски, связанные со статическим электричеством, обеспечивая безопасность своих сотрудников и защиту оборудования. Уделяя приоритетное внимание безопасности статического электричества, ООО «МОДСЕН» поддерживает безопасную рабочую среду, способствующую успеху их деятельности.

## Обеспечение пожарной безопасности

В ООО «МОДСЕН» согласно ТКП 474-2013, помещение в котором будет создаваться, входит в категорию помещений В1-В4, пожароопасные, помещении есть горючие вещества и материалы, способные при взаимодействии с кислородом или воздуха гореть [3].

В качестве первичных средств пожаротушений, в организации используются порошковые, углекислотные огнетушители марки ОП-5 и ОУ-10 соответственно, так как они наибольшим образом подходят для тушения электронной техники и близлежащих объектов, также имеются пожарные краны. Огнетушители размещаются на видных и легкодоступных местах для сотрудников, для быстрой ликвидации в случаи возгорания, в каждом помещении с сотрудниками, не препятствуют безопасной эвакуации людей. Расстояние от возможного очага пожара, до места размещения первичного средства пожаротушения должно быть не более 20 метров. Пожарные краны, марки Ду50 размещены на коридоре в специально оборудованных полках с закрывающейся дверью, механизмом, снабжены пломбой.

В организации отсутствуют средства автоматического пожаротушения. В качестве средств и устройств об извещении о пожаре используются дымоуловители и пожарные извещатели. Дымоуловители равномерно распложены по помещениям и располагаются на потолке. Пожарные извещатели расположены в помещениях на стенках в легко доступных местах.

Пожарная дружина, как это обязательно установлено на каждом предприятии, входит в обязательный численный состав организации. Численный состав дружины составляет 10 человек, поскольку численность самой организации не превышает 100 человек. Командиры, старшие расчетов, назначены руководителем организации. В помещениях на видном месте вывешен табель действия при пожарах. Все члены дружины своевременно проходят контроль знаний.

Основными задачами добровольной пожарной дружины являются:

* контроль за соблюдением противопожарного режима;
* проведение разъяснительной работы среди работников по соблюдению противопожарного режима на рабочих местах и правил осторожного обращения с огнем в быту;
* надзор за исправностью средств пожаротушения и их укомплектованностью;
* вызов пожарной службы в случае возникновения пожара, принятие мер по его тушению имеющимися средствами и т.п.

Члены добровольной пожарной дружины принимают участие в локализации и ликвидации загораний, эвакуации людей и материальных ценностей из горящих помещений.

При эвакуации здания, сотрудники покидают офисные помещения, направляются к лестнице, по которой осуществляют спуск, затем выходят из здания. Пути эвакуации отражены на плане эвакуации. Сам план содержит схему занимаемых помещений организацией, с обозначениями. Каждый сотрудник обязательно ознакомляется с планом эвакуации.

## Охрана окружающей среды

Проблема защиты окружающей среды ‒ одна из важнейших задач современности [4]. Выбросы промышленных предприятий, энергетических систем и транспорта в атмосферу, водоемы и недра на современном этапе развития достигли таких размеров, что в большинстве районов земного шара уровни загрязнений существенно превышают допустимые санитарные нормы. Множество разработанных технологических процессов и появление новых видов продукции, особенно в химической промышленности, привели не только к увеличению количества загрязнений, но и к существенному увеличению числа токсичных примесей, поступающих в окружающую среду. Вредные выбросы промышленных предприятий и других источников загрязнения оказывают отрицательное воздействие не только на окружающую среду, но и в ряде случаев значительно влияют на процесс эксплуатации технических средств.

Решение проблемы защиты окружающей среды от выбросов промышленных предприятий состоит в создании замкнутых технологических циклов (безотходные системы), однако их разработка и внедрение требуют новых технологических и конструктивных решений, а также больших капиталовложений. В современных условиях часто используют способы защиты окружающей среды от примесей, основанные на максимальном их улавливании или обезвреживании в специальных аппаратах.

Проблема охраны окружающей среды − комплексная проблема. Планируя дальнейшее развитие индустриального производства, необходимо оценивать эффективность его развития. Комплексный характер проблемы охраны окружающей среды определяется сложностью системы, состоящей из природы и производства. Оптимальное развитие этой системы невозможно без комплексного учета социальных, экологических, технических, экономических, правовых и международных аспектов проблемы.

Вопросы экологии на ООО «МОДСЕН» играют не последнюю роль в ее производственной деятельности.

Используемая вода в организации, сточная вода, направляется в канализацию, для дальнейшей приемки водоочистительными сооружениями. Стоит заметить, что вода, должна приниматься согласно условиям местных исполнительных органов, по согласованию с органами Министерства природных ресурсов и охраны окружающей среды Республики Беларусь. Данные требования также в самом общем виде определяются в Правилах пользования централизованными системами водоснабжения, водоотведения (канализации) в населенных пунктах. Способы очистки сточных вод:

* механический – размельчение осадков и отложений различными инструментами и механизмами;
* гидромеханический – подача струи воды под давлением для размыва и выноса отложений и грязи;
* физико-химический – циркуляция специально подобранных растворителей в аппаратуре без ее вскрытия;
* выжигание отложений в струе воздуха без вскрытия аппаратов и др.

ОДО «Экология города» - это стремительно развивающаяся компания по вывозу отходов, переработке и утилизации отходов в Минске и Минской области, согласно нормам, в существующем законодательстве. Сама утилизация производится термически. Термическая утилизация, подразумевает собой переработку отходов, путем сжигания. Способы утилизации бытового мусора:

* захоронение – производственный мусор, который не может быть использован вторично, подвергается хранению на специальных полигонах и свалках. Сбор, переработка и складирование радиоактивных отходов — большая проблема для любого государства. Их хранят в могильниках, которые должны быть расположены как можно дальше от грунтовых вод, в сейсмически безопасных зонах;
* компостирование – естественное разложение тяжёлых бытовых отходов. В результате процесса получают компост, который эффективно используется в сельском хозяйстве;
* термическая переработка – превращение мусора в топливо, путём его сжигания;
* плазменная переработка – процесс представляет собой разложение ТБО без доступа воздуха. Мусор нагревается и расщепляется на вещества с более низкой молекулярной массой. Пиролиз имеет больше преимуществ, чем сжигание: не загрязняется окружающая среда, тяжелые металлы превращаются в золу, в остатках отходов не содержится токсичных веществ.

В ООО «МОДСЕН» используются лампы, как люминесцентные, так и ртутные. По истечению срока использования ламп прибытию их в негодность, нужна их утилизация. Утилизация ламп осуществляется путем оформления заявки в организации «БелВторСырье», затем вывоз и утилизация [4].

Приемом списанных мониторов также занимается организация «БелВторСырье», с оформлением предварительной заявки.

1. Экономический раздел

## Обоснование необходимости выведения на рынок программного средства

Целью экономического раздела дипломного проекта является оценка экономической эффективности разработки и определение целесообразности вывода разработанного программного средства на рынок. Экономический раздел должен охватывать следующие аспекты проекта:

* план разработки: детальный план разработки программного средства по автоматизации сбора и обработки данных для подготовки и проведения олимпиады;
* расчет стоимости разработки: расчет стоимости разработки, начиная с даты получения первой версии технического задания до оформления документации и сдачи программного средства;
* оценка экономической эффективности: оценка экономической эффективности программного средства с учетом затрат на разработку и потенциальных выгод от его внедрения на рынок.

Для оценки экономической эффективности программного средства необходимо применить комплексный и систематический подход, включающий этапы расчета сроков разработки с учетом всех факторов, определение стоимости разработки, и оценку экономической эффективности, учитывая как затраты на разработку, так и предполагаемые выгоды от использования программного средства.

## Составление плана по разработке программного средства

План разработки программного средства представлен в таблице 6.1.

Таблица 6.1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов и видов работ | Исполнитель (должность, квалификация) | Количество исполнителей | Трудоемкость, (человеко-дни, месяцы) |
| Подготовительный | техник-программист | 1 | 3 |
| Выбор методов и средств | техник-программист | 1 | 3 |
| Разработка алгоритмов и программ | техник-программист | 1 | 25 |
| Отладка программ и анализ результатов | техник-программист | 1 | 7 |
| Оформление документации и подготовка к сдаче разработки | техник-программист | 1 | 2 |

Гистограмма, отображающая стадии разработки программного средства, приведена на рисунке 6.1.

Рисунок 6.1

## Расчет затрат на разработку программного средства

Расчет стоимости разработки программного средства является важнейшим аспектом определения целесообразности и прибыльности проекта. Поскольку научно-техническая продукция в рыночных отношениях считается товаром, определение цены основного результата дипломного проекта имеет важное значение.

При определении цены важно учитывать отраслевые рекомендации по установлению цены на научно-техническую продукцию, которые, как правило, предполагают механическую аналогию с материальным производством, расчет затрат или предполагаемых расходов, а также учет минимального уровня рентабельности.

Чтобы оценить стоимость программного средства и определить его экономический эффект, необходимо рассмотреть несколько элементов в определенной последовательности [2]. К этим элементам относятся:

* материальные затраты: сюда входит стоимость любых физических материалов, необходимых для разработки программного средства;
* затраты на оплату труда: сюда входит стоимость труда разработчиков, дизайнеров и другого персонала, задействованного в проекте;
* вычеты на социальные нужды: сюда входят любые взносы или налоги, необходимые для программ социального обеспечения;
* амортизация основных средств и нематериальных активов: сюда входит амортизация любого оборудования или нематериальных активов, использованных при разработке программного средства;
* прочие затраты: сюда входят любые дополнительные расходы, не охваченные вышеперечисленными элементами, например, расходы на маркетинг и рекламу.

Учащиеся-дипломники отвечают за расчет затрат, связанных с созданием программного средства. Принимая во внимание различные элементы и следуя определенной последовательности, учащиеся могут точно оценить стоимость своего программного средства и определить его экономическую целесообразность.

Определяются материальные затраты на выполнение работ по теме, включая стоимость покупных комплектующих изделий и полуфабрикатов на изготовление макетов и опытных образцов. Расчет осуществляется по формуле (6.1)

, (6.1)

где – коэффициент, учитывающий транспортно-заготовительные расходы, ≈ от 1,05 до 1,10;

– норма расхода i-го вида материалов;

– действующая отпускная цена за единицу i-го вида материала, руб.;

– количество применяемых видов материалов.

Расчет целесообразно представить в таблице 6.2.

Таблица 6.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование материалов покупных полуфабрикатов и комплектующих изделий | Единица измерения | Количество | Цена приобретения без НДС, руб. | НДС, руб. | Цена с НДС, руб. |
| Бумага | пачка | 1 | 29,99 | 29,99 | 29,99 |
| Ручка шариковая | штука | 1 | 1,89 | 1,89 | 1,89 |
| Папка-скоросшиватель | штука | 1 | 5,90 | 5,90 | 5,90 |
| Диск | штука | 1 | 2,00 | 2,00 | 2,00 |
| Всего расходов | | | | | 39,78 |
| Всего с транспортно-заготовительными расходами | | | | | 42,96 |

Затраты на электроэнергию находятся исходя из продолжительности периода разработки программного обеспечения, количества кВт/ч, затраченных на его проектирование и тарифа за 1 кВт/ч. по следующей формуле (6.2)

, (6.2)

где – стоимость одного кВт/ч,руб.;

– количество кВт/ч.

Тарифы на электроэнергию применяются согласно приложению к декларации *«*Об уровне тарифов на электроэнергию, отпускаемую республиканским унитарным предприятием(РУП) Электроэнергетики ГПО «Белэнерго» для юридических лиц и ИП» на соответствующий период времени, когда разрабатывается программное обеспечение.

Базовый тариф для прочих потребителей составляет 0,43912 руб. за 1 кВт/ч. Время реализации проекта 40 дней, среднее потребление энергии в месяц составило 75 кВт/ч, то есть было потреблено 143 кВт/ч. Исходя из вышеизложенного получаем, что на электроэнергию было затрачено

Рэ= 0,43912143 = 62,79 руб.

Определяется основная заработная плата научно-технического персонала, непосредственно занятого выполнением работ.

К этой статье относятся основная заработная плата работников, а также премии, входящие в фонд заработной платы. Среднее количество рабочих дней в месяце равно 21, а средняя продолжительность рабочего дня составляет восемь часов. Следовательно, часовая заработная плата определяется делением размера оклада на количество рабочих часов в месяце (т.е. на 168 часов).

Тарифная ставка первого разряда на предприятии составляет 227 руб. Трудоемкость определяется исходя из данных, представленных в таблице 6.1 (столбец «Трудоёмкость» умноженный на восемь).

Величина затрат исчисляется исходя из численности различных категорий исполнителей и трудоемкости выполнения отдельных видов работ, тарифных ставок за один день или месячных должностных окладов, премиальных систем оплаты труда исполнителей по формуле (6.3)

(6.3)

где – тарифная ставка за день (месячный оклад) i-й категорий работников;

– время фактической работы работника i-й категории по теме, дн. или мес.;

– коэффициент премий по премиальным системам,от 1,10 до 1,30.

Расчет основной заработной платы по теме приведен в таблице 6.3.

Таблица 6.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование категорий работников и должностей | Количество штатных единиц, чел. | Заработная плата за 1 день (месяц), руб. | Трудозатраты, дн. или мес. | Сумма, руб. |
| Техник – программист | 1 | 460,81 | 2 | 921,62 |
| Всего | | | | 921,62 |
| Всего с коэффициентом премий | | | | 1124,38 |

Определяется дополнительная заработная плата исполнителей, включающая разнообразные предусмотренные трудовым законодательством выплаты, по формуле (6.4)

, (6.4)

где Ндз– норматив дополнительной заработной платы, Ндз ≈ от 10 до 20%.

Рассчитываются отчисления органам социальной защиты по формуле (6.5)

, (6.5)

где Рос – норма отчислений на социальную защиту, Рос=34%.

Также рассчитываются отчисления на страхование от несчастных случаев на производстве и профессиональных заболеваний (Рстр) по ставке действующего законодательства (Нбгс принимается равным от 0,3 до 0,9%). Для расчетов среднее значение Нбгс принимается равным 0,6%. Рстр рассчитывается по формуле (6.6)

(6.6)

По статье «Амортизация основных средств и нематериальных активов» рассчитываются амортизационные отчисления (АО), исходя из стоимости основных средств (ОС), используемых в процессе разработки программного обеспечения, сроков эксплуатации оборудования (Тс) и годовой нормы амортизации (На).

Для определения затрат по данному элементу будет использоваться линейный способ начисления амортизации. Нормативные сроки службы машин и оборудования составляют 5 лет.

Норма амортизации для линейного способа начисления вычисляется по формуле (6.7)

, (6.7)

где Тс – срок службы оборудования, лет.

Для линейного способа начисления амортизационные отчисления равномерно распределены на весь период службы оборудования и вычисляются на один год. Сумма амортизационных отчислений (АО) рассчитывается по формуле (6.8) с учетом того, что разработка программного средства длилась 40 дней

(6.8)

В расчетах принимаем условно размер прочих затрат равных от 30 до 50% от суммы всех остальных затрат на разработку по формуле (6.9)

, (6.9)

где Нпр – норматив прямых расходов, Нпр≈ от 20 до 30%.

Определяется полная себестоимость научно-технической продукции как сумма всех групп затрат по формуле (6.10)

(6.10)

Все приведенные выше расчеты целесообразно объединить в сводную таблицу 6.4.

Таблица 6.4

|  |  |
| --- | --- |
| Элемент затрат | Затраты, руб. |
| Материальные затраты () | 42,96 |
| Электроэнергия () | 62,79 |

Продолжение таблицы 6.4

|  |  |
| --- | --- |
| Элемент затрат | Затраты, руб. |
| Затраты на оплату труда () | 1124,38 |
| Дополнительная заработная плата () | 168,66 |
| Отчисления органам социальной защиты (); | 439,63 |
| Отчисления на страхования от несчастных случаев () | 7,76 |
| Амортизация () | 66,70 |
| Прочие затраты () | 224,88 |
| Общая сумма затрат () | 2137,76 |

Таблицу 6.4 удобно представить в виде диаграммы, представленной на рисунке 6.2, отображающей элементы затрат программного средства.

Рисунок 6.2

Исходя из представленных данных, общая стоимость разработки программного средства составляет 2137,76 рублей. Основными составляющими себестоимости являются затраты на оплату труда (1124,38 руб.), отчисления в организации социального страхования (439,63 руб.) и амортизация (66,7 руб.). Материальные затраты и электроэнергия составляют лишь небольшую часть общих затрат, в то время как дополнительная заработная плата и отчисления на страхование от несчастных случаев также вносят значительный вклад. Наконец, прочие расходы (224,88 руб.) составляют умеренную часть общей стоимости.

В заключение, очевидно, что затраты на оплату труда и отчисления в организации социального обеспечения являются самыми крупными компонентами общей стоимости, в то время как прочие затраты и амортизация также играют значительную роль. Стоимость материалов и электроэнергии, с другой стороны, относительно невелика по сравнению с общей стоимостью. Исходя из этой информации, важно сосредоточиться на управлении затратами на оплату труда и отчислениями в организации социального обеспечения, чтобы контролировать общую стоимость разработки программного продукта.

## Расчет экономического эффекта у разработчика и пользователя программы

### Экономический эффект у разработчика программы

Заказчик оплачивает разработчику всю сумму расходов по проекту, включая прибыль. После уплаты налогов из прибыли в распоряжении заказчика остается чистая прибыль от проекта [2]. Ввиду того, что программное обеспечение разрабатывается для одного объекта, чистую прибыль можно считать в качестве экономического эффекта разработчика от реализованного программного средства.

В дипломном проекте отпускная цена программного средства, представляет собой не цену за единицу продукции, а цену проекта, за которую его можно продать и получить определённую выгоду.

Отпускная цена продукции формируется исходя из плановой себестоимости производства продукции, всех видов установленных налогов и прибыли, а также качества, потребительских свойств, продукции и конъюнктуры рынка.

С учетом действующих в республике нормативных документов отпускная цена на продукцию рассчитывается по формуле (6.11)

, (6.11)

где ОЦ – отпускная цена разработчика, руб.;

З – затраты на разработку, руб.;

П – прибыль, руб.

Прибыль рассчитывается по формуле (6.12)

, (6.12)

где R –уровень рентабельности (от 10 до 30%),

Стоимость проекта с учётом налога на добавленную стоимость (НДС), представляет собой сумму отпускной цены и налога на добавленную стоимость и высчитывается по формуле(6.13)

, (6.13)

где ставка НДС –20%

Отпускная цена рассчитывается по формуле (6.14)

(6.14)

Таким образом, разработчик программного средства может продать заказчику программное средство покрыв затраты и обеспечив прибыль за разработку проекта.

### Экономический эффект у пользователя программы

Экономический эффект у пользователя программного средства выражается в виде экономии трудовых, материальных и финансовых ресурсов, получаемой от:

* повышение производительности сервиса (увеличение числа выводимых за единицу времени документов, уменьшение среднего времени подготовки отчета и т.д.), что выражается в снижении трудоемкости выполнения операций, решении задач, подготовки данных, обработки информации и анализа результатов;
* сокращение затрат на оплату машинного времени и расходных материалов;
* повышения уровня сервиса (сокращение времени на устранение инцидентов);
* улучшения показателей основной деятельности предприятия в результате использования программного продукта и т.д.

Программное обеспечение для автоматизации сбора и обработки данных во время олимпиады было разработано за 320 часов, стоимость программного обеспечения была определена таким образом, чтобы покрыть расходы на разработку и получить прибыль, в результате чего окончательная цена составила 2821,31 руб.

Автоматизируя такие задачи, как сбор и обработки данных для подготовки и проведения олимпиады, программное обеспечение может помочь организаторам сэкономить ценное время, которое можно использовать, чтобы сосредоточиться на других аспектах олимпиады [2]. Кроме того, способность программного обеспечения предоставлять точные и своевременные результаты может повысить качество соревнований, что приведет к увеличению числа участников. Разработка и распространение программного обеспечения также могут создать рабочие места в индустрии программного обеспечения, способствуя росту экономики.

Таким образом, программное обеспечение имеет потенциал для обеспечения значительных преимуществ для образовательных учреждений и экономики в целом.

# Заключение

В рамках дипломного проектирования на тему «Создание программного средства по автоматизации сбора и обработки данных для подготовки и проведения олимпиады» было разработано программное средство «QuizHut.exe» со следующими возможностями: регистрация участников олимпиады; выгрузка данных об итогах проведения олимпиады; публикация заданий для участников олимпиады:

* активации задания;
* получение решений от участников в заданные сроки;
* закрепление полученных решений за участником.

В процессе разработки данного программного средства были применены и закреплены знания по уже изученному материалу, были отработаны навыки владения методами надежного программирования и эффективности разработки программного обеспечения в Microsoft Visual Studio 2022 с использованием языка программирования C#, разработана база данных средствами системы управления базами данных MySQL.

В разделе охраны труда были рассмотрены следующие вопросы:

* правовые, нормативные, социально-экономические и организационные вопросы охраны труда;
* разработка мер по нормализации и защиты работающих от статического электричества при эксплуатации ПЭВМ;
* пожарная безопасность;
* охрана окружающей среды.

В экономическом разделе была рассчитана стоимость разработки программного средства, которая составила 2821,31 рубля.

Программное средство не имеет недостатков, однако обладает рядом преимуществ:

* отсутствие прямого доступа пользователей к информационным ресурсам программы;
* возможность одновременного использования одного серверного ресурса различными пользователями;
* сокращение времени и объема работы специалистов.

Программа реализована в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

Программное средство готово к практическому использованию и может быть дополнено и модернизировано.

**Список информационных источников**

1. Багласова, Т.Г. Методические указания по дипломному проектированию / Т.Г. Багласова – Минск: КБП, 2023. 40 с.
2. Григораш, М.А. Методические указания по технико-экономическому обоснованию дипломных проектов / М.А. Григораш, Д.И. Кошепарова ­ Минск: КБП, 2021. 20 с.
3. Категорирование помещений, зданий и наружных установок по взрывопожарной и пожарной безопасности : ТКП 474-2013. Введ. 29.01.13. – Минск : Научно-исследовательский ин-т пожарной безопасности и проблем чрезвычайных ситуаций, 2013. ­ 53 с.
4. Об охране окружающей среды: Закон Респ. Беларусь, 26 янв. 1992 г., № 1982-XII : в ред. Закона Респ. Беларусь от 29.01.1993 г.
5. Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны : ГОСТ 12.1.005-88.­ Введ. 17.12.92. – Минск : Госстандарт : Государственный комитет по стандартизации Республики Беларусь, 1992. 49 с.
6. Документация по Windows | Microsoft Learn [Электронный ресурс]. – Microsoft, 2023. – Режим доступа : <https://learn.microsoft.com/en-us/windows/>. –Дата доступа :05.05.2023.
7. Документация по Visual Studio | Microsoft Learn [Электронный ресурс]. – Microsoft, 2023. – Режим доступа : https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022. – Дата доступа :10.05.2023.
8. Документация по C#. Начало работы, руководства, справочные материалы. | Microsoft Learn [Электронный ресурс]. – Microsoft, 2023. – Режим доступа : view-source:https://learn.microsoft.com/ru-ru/dotnet/csharp/. – Дата доступа :18.05.2023.
9. Документация по Windows Presentation Foundation для .NET 7 | Microsoft Learn [Электронный ресурс]. – Microsoft, 2023. – Режим доступа : https://learn.microsoft.com/ru-u/dotnet/desktop/wpf/?view=netdesktop-7.0. – Дата доступа :18.05.2023.
10. Центр документации по Entity Framework | Microsoft Learn [Электронный ресурс]. – Microsoft, 2023. – Режим доступа :  https://learn.microsoft.com/ru-ru/ef/. – Дата доступа :18.04.2023.
11. draw.io Документация [Электронный ресурс]. – JGraph Ltd, 2023. – Режим доступа :  https://www.drawio.com/doc/. – Дата доступа :18.04.2023.
12. Dr.Explain – программа для быстрого создания help файлов, online справки, руководств пользователей и документации [Электронный ресурс]. – [Indigo Byte Systems, LLC](http://www.indigobyte.com/), All rights reserved,  2004 - 2023. – Режим доступа : https://www.drexplain.ru/. ­ Дата доступа :05.05.2023.
13. Microsoft Office Excel­ [Электронный ресурс]. – ОфисГуруНьюс, 2023. – Режим доступа : <https://office-guru.ru/excel/microsoft-office-excel-chto-eto-59.html>.– Дата доступа :05.05.2023.
14. MySQL Документация [Электронный ресурс]. – [Oracle](https://www.oracle.com/legal/copyright.html), 2023. – Режим доступа :  <https://dev.mysql.com/doc/>. – Дата доступа :18.04.2023.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программы**

public class ApplicationDbContext : IdentityDbContext<ApplicationUser, IdentityRole, string>

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Answer> Answers { get; set; } // Коллекция ответов

public DbSet<Question> Questions { get; set; } // Коллекция вопросов

public DbSet<Quiz> Quizzes { get; set; } // Коллекция викторин

public DbSet<Category> Categories { get; set; } // Коллекция категорий

public DbSet<Group> Groups { get; set; } // Коллекция групп

public DbSet<Event> Events { get; set; } // Коллекция событий

public DbSet<Result> Results { get; set; } // Коллекция результатов

public DbSet<StudentGroup> StudentsGroups { get; set; } // Коллекция студентов в группах

public DbSet<EventGroup> EventsGroups { get; set; } // Коллекция событий в группах

public DbSet<ScheduledJob> ScheduledJobs { get; set; } // Коллекция запланированных заданий

}

//Класс для работы с викторинами

public class QuizzesService : IQuizzesService

{

//Репозиторий типа викторины

private readonly IRepository<Quiz> quizRepository;

private readonly IExpressionBuilder expressionBuilder;

//Конструктор класса

public QuizzesService(

IRepository<Quiz> quizRepository,

IExpressionBuilder expressionBuilder)

{

this.quizRepository = quizRepository;

this.expressionBuilder = expressionBuilder;

}

//Метод получения всех викторин по Id создателя, Id категории и критериям поиска

public async Task<IEnumerable<T>> GetAllQuizzesAsync<T>(

string creatorId = null,

string searchCriteria = null,

string searchText = null,

string categoryId = null)

{

//Получаем все викторины

var query = quizRepository.AllAsNoTracking();

//Если Id создателя указано, то добавить в выборку

if (creatorId != null)

{

query = query.Where(x => x.CreatorId == creatorId);

}

//Если Id категории указано, то добавить в выборку

if (categoryId != null)

{

query = query.Where(x => x.CategoryId == categoryId);

}

//Если текст для поиска указан, то добавить в выборку

var emptyNameInput = searchText == null && searchCriteria == "Name";

if (searchCriteria != null && !emptyNameInput)

{

var filter = expressionBuilder.GetExpression<Quiz>(searchCriteria, searchText);

query = query.Where(filter);

}

//Отсортировать по убыванию и вернуть

return await query.OrderByDescending(x => x.CreatedOn)

.To<T>()

.ToListAsync();

}

//Метод получения всех викторин по Id категории

public async Task<IList<T>> GetQuizzesByCategoryIdAsync<T>(string id)

{

return await quizRepository

.AllAsNoTracking()

.Where(x => x.CategoryId == id)

.To<T>()

.ToListAsync();

}

//Метод получения всех викторин, которые не назначены на категорию с переданным Id

public async Task<IList<T>> GetUnAssignedQuizzesToCategoryByCreatorIdAsync<T>(string categoryId, string creatorId)

{

return await quizRepository

.AllAsNoTracking()

.Where(x => x.CreatorId == creatorId && x.CategoryId != categoryId)

.To<T>()

.ToListAsync();

}

//Метод получения всех викторин, которые не назначены на событие

public async Task<IList<T>> GetUnAssignedQuizzesToEventAsync<T>(string creatorId = null)

{

var query = quizRepository

.AllAsNoTracking()

.Where(x => x.EventId == null);

if (creatorId != null)

{

query = query.Where(x => x.CreatorId == creatorId);

}

return await query.OrderByDescending(x => x.CreatedOn)

.To<T>()

.ToListAsync();

}

//Метод получения всех викторин, которые назначены на событие с переданным Id

public async Task<IList<T>> GetQuizzesByEventId<T>(string eventId)

{

return await quizRepository

.AllAsNoTracking()

.Where(x => x.EventId == eventId)

.To<T>()

.ToListAsync();

}

//Метод получения викторины с переданным паролем

public async Task<T> GetQuizByPasswordAsync<T>(string password)

{

return await quizRepository

.AllAsNoTracking()

.Where(x => x.Password == password)

.To<T>()

.FirstOrDefaultAsync();

}

//Метод создания викторины

public async Task<string> CreateQuizAsync(string name, string description, int? timer, string creatorId, string password)

{

//Создание нового объекта викторины

var quiz = new Quiz

{

Name = name,

Description = description,

Timer = timer,

CreatorId = creatorId,

Password = password

};

//Добавление новой викторины

await quizRepository.AddAsync(quiz);

//Сохранение изменений в базе данных

await quizRepository.SaveChangesAsync();

return quiz.Id;

}

//Метод обновления викторины

public async Task UpdateQuizAsync(string id, string name, string description, int? timer, string password)

{

var quiz = await quizRepository

.All()

.FirstOrDefaultAsync(x => x.Id == id);

//Установка новых значений для полей

quiz.Name = name;

quiz.Description = description;

quiz.Timer = timer;

quiz.Password = password;

//Обновление викторины

quizRepository.Update(quiz);

//Сохранение изменений в базе данных

await quizRepository.SaveChangesAsync();

}

//Метод удаления викторины

public async Task DeleteQuizAsync(string id)

{

var quiz = await quizRepository

.All()

.FirstOrDefaultAsync(x => x.Id == id);

quizRepository.Delete(quiz);

await quizRepository.SaveChangesAsync();

}

}

public class GroupsService : IGroupsService

{

private readonly IRepository<Group> repository; // Репозиторий для работы с группами

private readonly IStudentsGroupsService studentsGroupsService; // Сервис для работы со студентами в группах

private readonly IEventsGroupsService eventsGroupsService; // Сервис для работы с событиями в группах

private readonly IExpressionBuilder expressionBuilder; // Построитель выражений для фильтрации

public GroupsService(

IRepository<Group> repository,

IStudentsGroupsService studentsGroupsService,

IEventsGroupsService eventsGroupsService,

IExpressionBuilder expressionBuilder)

{

this.repository = repository;

this.studentsGroupsService = studentsGroupsService;

this.eventsGroupsService = eventsGroupsService;

this.expressionBuilder = expressionBuilder;

}

public async Task<IList<T>> GetAllGroupsAsync<T>(string creatorId = null, string eventId = null, string searchText = null)

{

var query = repository.AllAsNoTracking(); // Получение всех групп без отслеживания изменений

if (creatorId != null)

{

query = query.Where(x => x.CreatorId == creatorId); // Фильтрация по идентификатору создателя

}

if (eventId != null)

{

query = query.Where(x => !x.EventsGroups.Any(x => x.EventId == eventId)); // Фильтрация по идентификатору события

}

if (searchText != null)

{

var filter = expressionBuilder.GetExpression<Group>("Name", searchText); // Получение выражения для фильтрации по имени

query = query.Where(filter); // Применение фильтра

}

return await query.Include(x => x.StudentsGroups).ThenInclude(x => x.Student).OrderByDescending(x => x.CreatedOn).To<T>().ToListAsync(); // Получение списка групп с подключенными студентами и их сортировка по дате создания

}

public async Task<IEnumerable<T>> GetAllGroupsByEventIdAsync<T>(string eventId)

{

return await repository

.AllAsNoTracking()

.Where(x => x.EventsGroups.Any(x => x.EventId == eventId))

.To<T>()

.ToListAsync(); // Получение списка всех групп, связанных с указанным событием

}

public async Task<string> CreateGroupAsync(string name, string creatorId)

{

var group = new Group() { Name = name, CreatorId = creatorId }; // Создание новой группы

await repository.AddAsync(group); // Добавление группы в репозиторий

await repository.SaveChangesAsync(); // Сохранение изменений в репозитории

return group.Id; // Возврат идентификатора созданной группы

}

public async Task AssignStudentsToGroupAsync(string groupId, IList<string> studentsIds)

{

foreach (var studentId in studentsIds)

{

await studentsGroupsService.CreateStudentGroupAsync(groupId, studentId); // Назначение студентов в группу

}

}

public async Task AssignEventsToGroupAsync(string groupId, IList<string> evenstIds)

{

foreach (var eventId in evenstIds)

{

await eventsGroupsService.CreateEventGroupAsync(eventId, groupId); // Назначение событий в группу

}

}

public async Task UpdateGroupNameAsync(string groupId, string newName)

{

var group = await repository

.All()

.Where(x => x.Id == groupId)

.FirstOrDefaultAsync(); // Получение группы по идентификатору

group.Name = newName; // Обновление имени группы

repository.Update(group); // Обновление группы в репозитории

await repository.SaveChangesAsync(); // Сохранение изменений в репозитории

}

public async Task DeleteGroupAsync(string groupId)

{

var group = await repository

.All()

.Where(x => x.Id == groupId)

.FirstOrDefaultAsync(); // Получение группы по идентификатору

repository.Delete(group); // Удаление группы из репозитория

await repository.SaveChangesAsync(); // Сохранение изменений в репозитории

}

public async Task DeleteStudentFromGroupAsync(string groupId, string studentId)

{

await studentsGroupsService.DeleteStudentGroupAsync(groupId, studentId); // Удаление студента из группы

}

public async Task DeleteEventFromGroupAsync(string groupId, string eventId)

{

await eventsGroupsService.DeleteEventGroupAsync(eventId, groupId); // Удаление события из группы

}

}

public class StudentsService : IStudentsService

{

private readonly IRepository<ApplicationUser> userRepository; // Репозиторий для работы со студентами

private readonly IExpressionBuilder expressionBuilder; // Построитель выражений для фильтрации

public StudentsService(

IRepository<ApplicationUser> userRepository,

IExpressionBuilder expressionBuilder)

{

this.userRepository = userRepository;

this.expressionBuilder = expressionBuilder;

}

public async Task<IList<T>> GetAllStudentsAsync<T>(

string teacherId = null,

string searchCriteria = null,

string searchText = null)

{

var query = userRepository.AllAsNoTracking(); // Получение всех студентов без отслеживания изменений

if (teacherId != null)

{

query = query.Where(x => x.TeacherId == teacherId); // Фильтрация по идентификатору учителя

}

if (searchCriteria != null && searchText != null)

{

var filter = expressionBuilder.GetExpression<ApplicationUser>(searchCriteria, searchText); // Получение выражения для фильтрации

query = query.Where(filter); // Применение фильтра

}

return await query

.To<T>()

.ToListAsync(); // Получение списка студентов

}

public async Task<IList<T>> GetAllStudentsByGroupIdAsync<T>(string groupId)

{

return await userRepository

.AllAsNoTracking()

.Where(x => x.StudentsInGroups.Select(x => x.GroupId).Contains(groupId)) // Получение студентов, связанных с указанной группой

.To<T>()

.ToListAsync();

}

public async Task<IList<T>> GetAllStudentsUnAssignedToGroup<T>(string groupId)

{

return await userRepository

.AllAsNoTracking()

.Where(x => !x.StudentsInGroups.Select(x => x.GroupId).Contains(groupId)) // Получение студентов, не связанных с указанной группой

.To<T>()

.ToListAsync();

}

public async Task<bool> AddStudentAsync(string email, string teacherId)

{

var user = await userRepository

.All()

.Where(x => x.Email == email)

.FirstOrDefaultAsync(); // Получение пользователя по электронной почте

if (user != null && user.TeacherId != teacherId)

{

user.TeacherId = teacherId; // Назначение учителя студенту

userRepository.Update(user); // Обновление пользователя в репозитории

await userRepository.SaveChangesAsync(); // Сохранение изменений в репозитории

return true;

}

return false;

}

public async Task DeleteStudentFromTeacherListAsync(string studentId, string teacherId)

{

var studentToRemove = await userRepository

.All()

.Where(x => x.Id == studentId)

.FirstOrDefaultAsync(); // Получение студента по идентификатору

if (studentToRemove != null)

{

studentToRemove.TeacherId = null; // Удаление связи учителя со студентом

userRepository.Update(studentToRemove); // Обновление студента в репозитории

await userRepository.SaveChangesAsync(); // Сохранение изменений в репозитории

}

}

}

public class EventsService : IEventsService

{

private readonly IRepository<Event> repository; // Репозиторий для событий

private readonly IRepository<Quiz> quizRepository; // Репозиторий для викторин

private readonly IEventsGroupsService eventsGroupsService; // Сервис для групп событий

private readonly IScheduledJobsService scheduledJobsService; // Сервис для запланированных задач

private readonly IEmailSenderService emailSenderService; // Сервис для отправки электронной почты

private readonly IExpressionBuilder expressionBuilder; // Построитель выражений

public EventsService(

IRepository<Event> repository,

IRepository<Quiz> quizRepository,

IEventsGroupsService eventsGroupsService,

IScheduledJobsService scheduledJobsService,

IEmailSenderService emailSenderService,

IExpressionBuilder expressionBuilder)

{

this.repository = repository;

this.quizRepository = quizRepository;

this.eventsGroupsService = eventsGroupsService;

this.scheduledJobsService = scheduledJobsService;

this.emailSenderService = emailSenderService;

this.expressionBuilder = expressionBuilder;

}

public async Task<IList<T>> GetAllEventsAsync<T>(

string creatorId = null,

string searchCriteria = null,

string searchText = null)

{

var query = repository.AllAsNoTracking(); // Получить все события из репозитория

if (creatorId != null)

{

query = query.Where(x => x.CreatorId == creatorId); // Фильтр по создателю события

}

var emptyNameInput = searchText == null && searchCriteria == "Name"; // Проверка на пустой ввод имени

if (searchCriteria != null && !emptyNameInput)

{

var filter = expressionBuilder.GetExpression<Event>(searchCriteria, searchText); // Получить выражение для фильтрации

query = query.Where(filter); // Применить фильтр

}

return await query

.OrderByDescending(x => x.CreatedOn) // Сортировка по убыванию даты создания

.To<T>()

.ToListAsync();

}

public async Task<IList<T>> GetAllEventsByCreatorIdAndStatusAsync<T>(

Status status,

string creatorId,

string searchText = null)

{

var query = repository

.AllAsNoTracking()

.Where(x => x.Status == status && x.CreatorId == creatorId); // Фильтр по статусу и создателю события

if (searchText != null)

{

var filter = expressionBuilder.GetExpression<Event>("Name", searchText); // Получить выражение для фильтрации по имени

query = query.Where(filter); // Применить фильтр

}

return await query

.OrderByDescending(x => x.CreatedOn) // Сортировка по убыванию даты создания

.To<T>()

.ToListAsync();

}

public async Task<IList<T>> GetAllEventsFilteredByStatusAndGroupAsync<T>(

Status status,

string groupId,

string creatorId = null)

{

var query = repository

.AllAsNoTracking()

.Where(x => !x.EventsGroups.Any(x => x.GroupId == groupId)); // Фильтр по группе событий

if (creatorId != null)

{

query = query.Where(x => x.CreatorId == creatorId); // Фильтр по создателю события

}

return await query

.Where(x => x.Status != status) // Фильтр по статусу

.OrderByDescending(x => x.CreatedOn) // Сортировка по убыванию даты создания

.To<T>()

.ToListAsync();

}

public async Task<IList<T>> GetAllEventsByStatusAndStudentIdAsync<T>(

Status status,

string studentId,

string searchText = null)

{

var query = repository

.AllAsNoTracking()

.Include(x => x.Quizzes)

.ThenInclude(q => q.Results)

.Where(x => x.EventsGroups.Any(eg => eg.Group.StudentsGroups.Any(sg => sg.StudentId == studentId))); // Фильтр по студенту

if (searchText != null)

{

var filter = expressionBuilder.GetExpression<Event>("Name", searchText); // Получить выражение для фильтрации по имени

query = query.Where(filter); // Применить фильтр

}

return await query

.Where(x => x.Status == status) // Фильтр по статусу

.OrderByDescending(x => x.CreatedOn) // Сортировка по убыванию даты создания

.To<T>()

.ToListAsync();

}

public async Task<IList<T>> GetAllEventsByGroupIdAsync<T>(string groupId)

{

return await repository

.AllAsNoTracking()

.Where(x => x.EventsGroups.Any(x => x.GroupId == groupId)) // Фильтр по идентификатору группы

.To<T>()

.ToListAsync();

}

public async Task<IList<T>> GetAllEventsByGroupIdAsync<T>(string groupId)

{

return await repository

.AllAsNoTracking()

.Where(x => x.EventsGroups.Any(x => x.GroupId == groupId)) // Фильтр по идентификатору группы

.To<T>()

.ToListAsync();

}

public async Task AssignQuizzesToEventAsync(IList<string> quizIds, string eventId)

{

var @event = await repository

.Where(e => e.Id == eventId)

.FirstOrDefaultAsync();

foreach (var quizId in quizIds)

{

var quiz = await quizRepository

.All()

.Where(x => x.Id == quizId)

.FirstOrDefaultAsync();

@event.Quizzes.Add(quiz);

@event.Status = GetStatus(@event.ActivationDateAndTime, @event.DurationOfActivity, quizId);

if (@event.Quizzes.Count == 1)

{

await SheduleStatusChangeAsync(@event.ActivationDateAndTime, @event.DurationOfActivity, @event.Id, @event.Status);

}

}

repository.Update(@event);

await repository.SaveChangesAsync();

}

public async Task AssignGroupsToEventAsync(IList<string> groupIds, string eventId)

{

foreach (var groupId in groupIds)

{

await eventsGroupsService.CreateEventGroupAsync(eventId, groupId); // Создание связи между событием и группой

}

}

public async Task<string> CreateEventAsync(

string name,

string activationDate,

string activeFrom,

string activeTo,

string creatorId)

{

var activationDateAndTime = GetActivationDateAndTimeUtc(activationDate, activeFrom);

var durationOfActivity = GetDurationOfActivity(activationDate, activeFrom, activeTo);

var @event = new Event

{

Name = name,

Status = Status.Pending,

ActivationDateAndTime = activationDateAndTime,

DurationOfActivity = durationOfActivity,

CreatorId = creatorId,

};

await repository.AddAsync(@event); // Добавление нового события

await repository.SaveChangesAsync();

return @event.Id;

}

public async Task UpdateEventAsync(string id, string name, string activationDate, string activeFrom, string activeTo)

{

var @event = await repository

.All()

.Include(e => e.Quizzes)

.Where(x => x.Id == id)

.FirstOrDefaultAsync();

var activationDateAndTime = GetActivationDateAndTimeUtc(activationDate, activeFrom);

var durationOfActivity = GetDurationOfActivity(activationDate, activeFrom, activeTo);

@event.Name = name;

@event.ActivationDateAndTime = activationDateAndTime;

@event.DurationOfActivity = durationOfActivity;

@event.Status = GetStatus(activationDateAndTime, durationOfActivity, @event.Quizzes.FirstOrDefault()?.Id);

repository.Update(@event); // Обновление информации о событии

await repository.SaveChangesAsync();

if (@event.Quizzes.Any())

{

await SheduleStatusChangeAsync(activationDateAndTime, durationOfActivity, id, @event.Status);

}

}

class EventsViewModel : ViewModel, IMenuView

{

public static string Title { get; } = "События"; // Заголовок представления

public static IconChar IconChar { get; } = IconChar.CalendarDays; // Иконка представления

public Dictionary<string, string> SearchCriteriasInEnglish => new() // Словарь для перевода критериев поиска на английский язык

{

{ "Название", "Name" },

{ "Активные", "Active" },

{ "В ожидании", "Pending" },

{ "Завершенные", "Ended" }

};

private readonly IEventsService eventsService; // Сервис для работы с событиями

private readonly ISharedDataStore sharedDataStore; // Общее хранилище данных

private readonly IDateTimeConverter dateTimeConverter; // Конвертер даты и времени

private readonly IExporter exporter; // Экспортер данных

public EventsViewModel(

IEventsService eventsService,

ISharedDataStore sharedDataStore,

IDateTimeConverter dateTimeConverter,

IRenavigator eventActionsRenavigator,

IRenavigator eventSettingRenavigator,

IViewDisplayTypeService viewDisplayTypeService,

IExporter exporter)

{

// Инициализация зависимостей через конструктор

this.eventsService = eventsService;

this.sharedDataStore = sharedDataStore;

this.dateTimeConverter = dateTimeConverter;

this.exporter = exporter;

// Инициализация команд для навигации и выполнения операций

NavigateCreateEventCommand = new RenavigateCommand(eventActionsRenavigator, ViewDisplayType.Create, viewDisplayTypeService);

NavigateEditEventCommand = new RenavigateCommand(eventActionsRenavigator, ViewDisplayType.Edit, viewDisplayTypeService);

NavigateEventSettingsCommand = new RenavigateCommand(eventSettingRenavigator);

LoadDataCommandAsync = new ActionCommandAsync(OnLoadDataCommandExecutedAsync);

SearchCommandAsync = new ActionCommandAsync(OnSearchCommandAsyncExecute, CanSearchCommandAsyncExecute);

DeleteEventCommandAsync = new ActionCommandAsync(OnDeleteEventCommandExecutedAsync);

}

#region FieldsAndProperties

public ObservableCollection<EventListViewModel> events; // Коллекция событий

public ObservableCollection<EventListViewModel> Events

{

get => events;

set => Set(ref events, value);

}

private EventListViewModel selectedEvent; // Выбранное событие

public EventListViewModel SelectedEvent

{

get

{

sharedDataStore.SelectedEvent = selectedEvent;

return selectedEvent;

}

set => Set(ref selectedEvent, value);

}

private string searchCriteria; // Критерий поиска

public string SearchCriteria

{

get => searchCriteria;

set => Set(ref searchCriteria, value);

}

private string searchText; // Текст для поиска

public string SearchText

{

get => searchText;

set => Set(ref searchText, value);

}

private string? errorMessage; // Сообщение об ошибке

public string? ErrorMessage

{

get => errorMessage;

set => Set(ref errorMessage, value);

}

#endregion

#region NavigationCommands

public ICommand NavigateCreateEventCommand { get; } // Команда для создания события

public ICommand NavigateEditEventCommand { get; } // Команда для редактирования события

public ICommand NavigateEventSettingsCommand { get; } // Команда для настройки события

#endregion

#region LoadDataCommandAsync

public ICommandAsync LoadDataCommandAsync { get; } // Асинхронная команда для загрузки данных

private async Task OnLoadDataCommandExecutedAsync(object p)

{

await LoadEventsData(); // Загрузка данных событий

}

#endregion

#region SearchCommandAsync

public ICommandAsync SearchCommandAsync { get; } // Асинхронная команда для поиска данных

private bool CanSearchCommandAsyncExecute(object p) => !string.IsNullOrEmpty(SearchCriteria);

private async Task OnSearchCommandAsyncExecute(object p)

{

await LoadEventsData(SearchCriteriasInEnglish[SearchCriteria], SearchText); // Загрузка данных событий с применением фильтра поиска

}

#endregion

#region DeleteEventCommandAsync

public ICommandAsync DeleteEventCommandAsync { get; } // Асинхронная команда для удаления события

private async Task OnDeleteEventCommandExecutedAsync(object p)

{

await eventsService.DeleteEventAsync(SelectedEvent.Id); // Удаление выбранного события

await LoadEventsData(); // Загрузка данных событий

}

#endregion

private async Task LoadEventsData(string searchCriteria = null, string searchText = null)

{

var events = await eventsService.GetAllEventsAsync<EventListViewModel>(sharedDataStore.CurrentUser.Id, searchCriteria, searchText); // Получение всех событий

foreach (var @event in events)

{

@event.Duration = dateTimeConverter.GetDurationString(@event.ActivationDateAndTime, @event.DurationOfActivity); // Вычисление продолжительности события

@event.StartDate = dateTimeConverter.GetDate(@event.ActivationDateAndTime); // Вычисление даты начала события

}

Events = new(events); // Обновление коллекции событий

}

}

class EventActionsViewModel : ViewModel

{

private readonly IEventsService eventsService;

private readonly IQuizzesService quizzesService;

private readonly IGroupsService groupsService;

private readonly ISharedDataStore sharedDataStore;

private readonly IViewDisplayTypeService viewDisplayTypeService;

public EventActionsViewModel(

IEventsService eventsService,

IQuizzesService quizzesService,

IGroupsService groupsService,

ISharedDataStore sharedDataStore,

IRenavigator eventRenavigator,

IRenavigator eventSettingRenavigator,

IRenavigator addQuizRenavigator,

IRenavigator addGroupRenavigator,

IViewDisplayTypeService viewDisplayTypeService)

{

this.eventsService = eventsService;

this.quizzesService = quizzesService;

this.groupsService = groupsService;

this.sharedDataStore = sharedDataStore;

this.viewDisplayTypeService = viewDisplayTypeService;

// Подписка на событие изменения состояния отображения

viewDisplayTypeService.StateChanged += ViewDisplayTypeService\_StateChanged;

// Инициализация команд навигации

NavigateEventCommand = new RenavigateCommand(eventRenavigator);

NavigateEventSettingsCommand = new RenavigateCommand(eventSettingRenavigator);

NavigateAddQuizCommand = new RenavigateCommand(addQuizRenavigator, ViewDisplayType.Create, viewDisplayTypeService);

NavigateAddGroupCommand = new RenavigateCommand(addGroupRenavigator, ViewDisplayType.Create, viewDisplayTypeService);

// Инициализация асинхронных команд

LoadDataCommandAsync = new ActionCommandAsync(OnLoadDataCommandExecutedAsync, CanLoadDataCommandExecute);

CreateEventCommandAsync = new ActionCommandAsync(OnCreateEventCommandExecutedAsync, CanCreateUpdateEventCommandExecute);

UpdateEventCommandAsync = new ActionCommandAsync(OnUpdateEventCommandExecutedAsync, CanCreateUpdateEventCommandExecute);

AssignQuizToEventCommandAsync = new ActionCommandAsync(OnAssignQuizToEventCommandExecute, CanAssignQuizToEventCommandExecute);

AssignGroupsToEventCommandAsync = new ActionCommandAsync(OnAssignGroupsToEventCommandExecute, CanAssignGroupsToEventCommandExecute);

}

#region Поля и свойства

private bool isQuizzesEmpty;

public bool IsQuizzesEmpty

{

get => isQuizzesEmpty;

set => Set(ref isQuizzesEmpty, value);

}

private bool isGroupsEmpty;

public bool IsGroupsEmpty

{

get => isGroupsEmpty;

set => Set(ref isGroupsEmpty, value);

}

// Текущий тип отображения

public ViewDisplayType? CurrentViewDisplayType

{

get

{

if (viewDisplayTypeService.CurrentViewDisplayType == ViewDisplayType.Edit)

{

// Если тип отображения - редактирование, заполняем соответствующие поля

var timeParts = sharedDataStore.SelectedEvent.Duration.Split('-');

EventNameToCreate = sharedDataStore.SelectedEvent.Name;

EventActivationDate = sharedDataStore.SelectedEvent.StartDate;

EventAvaliableFrom = timeParts[0].Trim();

EventAvaliableTo = timeParts[1].Trim();

}

return viewDisplayTypeService.CurrentViewDisplayType;

}

}

private string eventNameToCreate;

public string EventNameToCreate

{

get => eventNameToCreate;

set => Set(ref eventNameToCreate, value);

}

private string eventActivationDate;

public string EventActivationDate

{

get => eventActivationDate;

set => Set(ref eventActivationDate, value);

}

private string eventAvaliableFrom;

public string EventAvaliableFrom

{

get => eventAvaliableFrom;

set => Set(ref eventAvaliableFrom, value);

}

private string eventAvaliableTo;

public string EventAvaliableTo

{

get => eventAvaliableTo;

set => Set(ref eventAvaliableTo, value);

}

// Список викторин

public ObservableCollection<QuizAssignViewModel> quizzes;

public ObservableCollection<QuizAssignViewModel> Quizzes

{

get => quizzes;

set => Set(ref quizzes, value);

}

// Список групп

public ObservableCollection<GroupAssignViewModel> groups;

public ObservableCollection<GroupAssignViewModel> Groups

{

get => groups;

set => Set(ref groups, value);

}

private string? errorMessage;

public string? ErrorMessage

{

get => errorMessage;

set => Set(ref errorMessage, value);

}

#endregion

#region Команды навигации

public ICommand NavigateEventCommand { get; }

public ICommand NavigateEventSettingsCommand { get; }

public ICommand NavigateAddQuizCommand { get; }

public ICommand NavigateAddGroupCommand { get; }

#endregion

#region Команда загрузки данных

public ICommandAsync LoadDataCommandAsync { get; }

private bool CanLoadDataCommandExecute(object p)

{

// Можно загружать данные, если тип отображения - добавление викторин или добавление групп

if (CurrentViewDisplayType == ViewDisplayType.AddQuizzes

||

CurrentViewDisplayType == ViewDisplayType.AddGroups)

{

return true;

}

return false;

}

private async Task OnLoadDataCommandExecutedAsync(object p)

{

// Загрузка данных в зависимости от типа отображения

if (CurrentViewDisplayType == ViewDisplayType.AddQuizzes)

{

await LoadQuizzesData();

}

if (CurrentViewDisplayType == ViewDisplayType.AddGroups)

{

await LoadGroupsData();

}

}

#endregion

#region Команда создания события

public ICommandAsync CreateEventCommandAsync { get; }

private bool CanCreateUpdateEventCommandExecute(object p)

{

// Можно создать или обновить событие, если заполнены все необходимые поля

if (!string.IsNullOrEmpty(EventNameToCreate) &&

!string.IsNullOrEmpty(EventActivationDate) &&

!string.IsNullOrEmpty(EventAvaliableFrom) &&

!string.IsNullOrEmpty(EventAvaliableTo))

{

return true;

}

return false;

}

private async Task OnCreateEventCommandExecutedAsync(object p)

{

// Проверка времени события и вывод сообщения об ошибке, если необходимо

var timeErrorMessage = eventsService.GetTimeErrorMessage(EventAvaliableFrom, EventAvaliableTo, EventActivationDate);

if (timeErrorMessage != null)

{

return;

}

// Создание события

await eventsService.CreateEventAsync(EventNameToCreate, EventActivationDate, EventAvaliableFrom, EventAvaliableTo, sharedDataStore.CurrentUser.Id);

NavigateEventCommand.Execute(p);

}

#endregion

#region Команда обновления события

public ICommandAsync UpdateEventCommandAsync { get; }

private async Task OnUpdateEventCommandExecutedAsync(object p)

{

// Получение информации о времени предыдущего доступа к событию

var timeParts = sharedDataStore.SelectedEvent.Duration.Split('-');

var oldEventAvaliableFrom = timeParts[0].Trim();

// Проверка времени события и вывод сообщения об ошибке, если необходимо

var timeErrorMessage = eventsService.GetTimeErrorMessage(EventAvaliableFrom, EventAvaliableTo, EventActivationDate, oldEventAvaliableFrom);

if (timeErrorMessage != null)

{

return;

}

// Обновление события

await eventsService.UpdateEventAsync(sharedDataStore.SelectedEvent.Id, EventNameToCreate, EventActivationDate, EventAvaliableFrom, EventAvaliableTo);

NavigateEventCommand.Execute(p);

}

#endregion

#region Команда назначения викторин к событию

public ICommandAsync AssignQuizToEventCommandAsync { get; }

private bool CanAssignQuizToEventCommandExecute(object p) => true;

private async Task OnAssignQuizToEventCommandExecute(object p)

{

// Проверка статуса события и вывод сообщения об ошибке, если необходимо

if (sharedDataStore.SelectedEvent.Status == Status.Ended)

{

return;

}

// Получение выбранных викторин

var selectedQuizes = Quizzes.Where(s => s.IsAssigned).Select(x => x.Id).ToList();

if (!selectedQuizes.Any())

{

return;

}

// Назначение викторин к событию

await eventsService.AssignQuizzesToEventAsync(selectedQuizes, sharedDataStore.SelectedEvent.Id);

NavigateEventSettingsCommand.Execute(p);

}

#endregion

#region Команда назначения групп к событию

public ICommandAsync AssignGroupsToEventCommandAsync { get; }

private bool CanAssignGroupsToEventCommandExecute(object p) => true;

private async Task OnAssignGroupsToEventCommandExecute(object p)

{

// Проверка статуса события и вывод сообщения об ошибке, если необходимо

if (sharedDataStore.SelectedEvent.Status == Status.Ended)

{

return;

}

// Получение выбранных групп

var selectedGroupIds = Groups.Where(s => s.IsAssigned).Select(x => x.Id).ToList();

if (!selectedGroupIds.Any())

{

return;

}

// Назначение групп к событию

await eventsService.AssignGroupsToEventAsync(selectedGroupIds, sharedDataStore.SelectedEvent.Id);

NavigateEventSettingsCommand.Execute(p);

}

#endregion

private async Task LoadQuizzesData()

{

// Загрузка неназначенных викторин

var unassignedQuizzes = await quizzesService.GetUnassignedQuizzesAsync();

// Загрузка назначенных викторин для выбранного события

var assignedQuizzes = await quizzesService.GetAssignedQuizzesAsync(sharedDataStore.SelectedEvent.Id);

// Создание списка викторин с информацией о назначении

var quizAssignViewModels = new ObservableCollection<QuizAssignViewModel>();

foreach (var quiz in unassignedQuizzes)

{

var isAssigned = assignedQuizzes.Any(q => q.Id == quiz.Id);

quizAssignViewModels.Add(new QuizAssignViewModel(quiz, isAssigned));

}

Quizzes = quizAssignViewModels;

// Проверка наличия викторин

IsQuizzesEmpty = !Quizzes.Any();

}

private async Task LoadGroupsData()

{

// Загрузка неназначенных групп

var unassignedGroups = await groupsService.GetUnassignedGroupsAsync();

// Загрузка назначенных групп для выбранного события

var assignedGroups = await groupsService.GetAssignedGroupsAsync(sharedDataStore.SelectedEvent.Id);

// Создание списка групп с информацией о назначении

var groupAssignViewModels = new ObservableCollection<GroupAssignViewModel>();

foreach (var group in unassignedGroups)

{

var isAssigned = assignedGroups.Any(g => g.Id == group.Id);

groupAssignViewModels.Add(new GroupAssignViewModel(group, isAssigned));

}

Groups = groupAssignViewModels;

// Проверка наличия групп

IsGroupsEmpty = !Groups.Any();

}

private void ViewDisplayTypeService\_StateChanged(object sender, ViewDisplayTypeChangedEventArgs e)

{

// Обновление типа отображения при его изменении

RaisePropertyChanged(nameof(CurrentViewDisplayType));

// Очистка полей при изменении типа отображения

EventNameToCreate = string.Empty;

EventActivationDate = string.Empty;

EventAvaliableFrom = string.Empty;

EventAvaliableTo = string.Empty;

// Очистка списков викторин и групп при изменении типа отображения

Quizzes = new ObservableCollection<QuizAssignViewModel>();

Groups = new ObservableCollection<GroupAssignViewModel>();

IsQuizzesEmpty = false;

IsGroupsEmpty = false;

}

}

**ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Результаты работы программы**

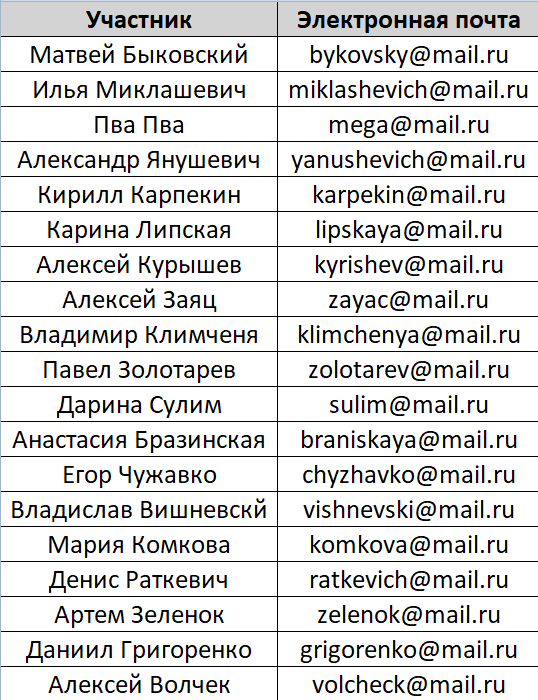


Рисунок Б.1

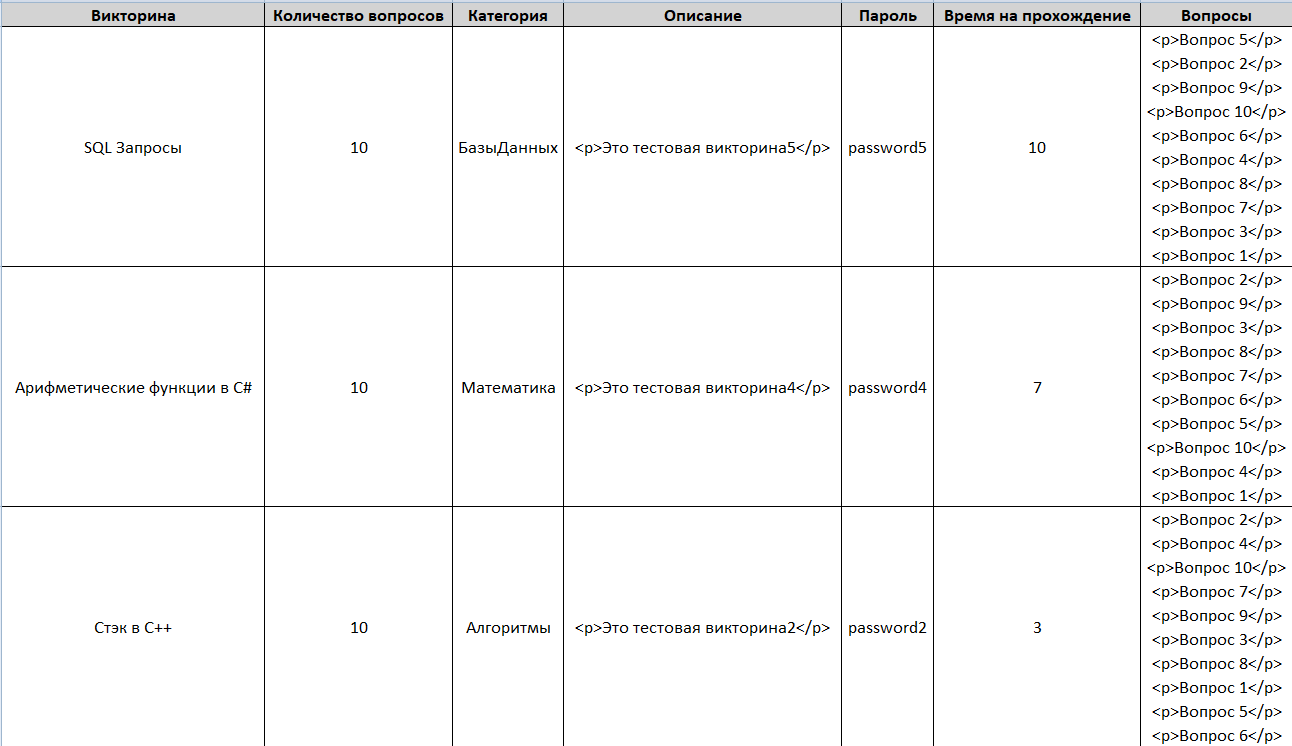


Рисунок Б.2



Рисунок Б.3

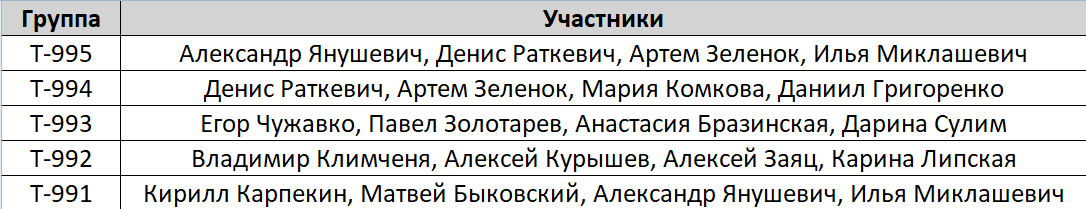


Рисунок Б.4

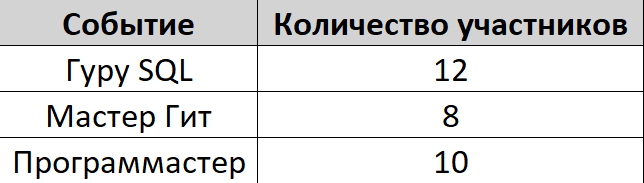


Рисунок Б.5

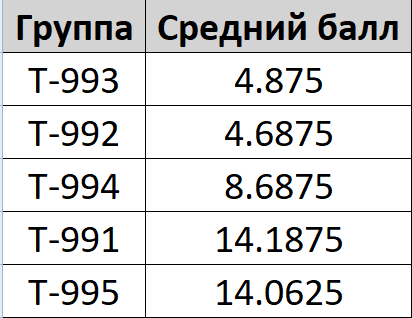


Рисунок Б.6

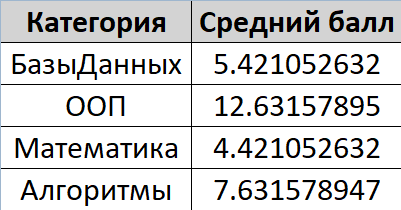


Рисунок Б.7

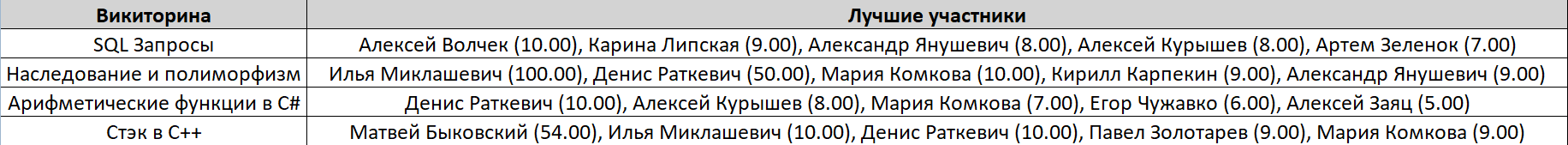


Рисунок Б.8

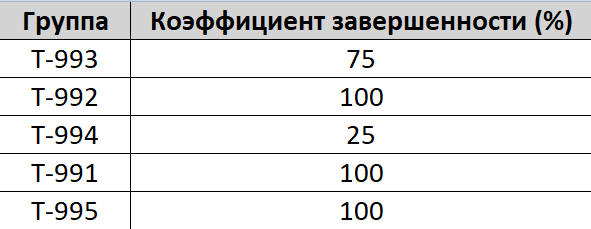


Рисунок Б.9

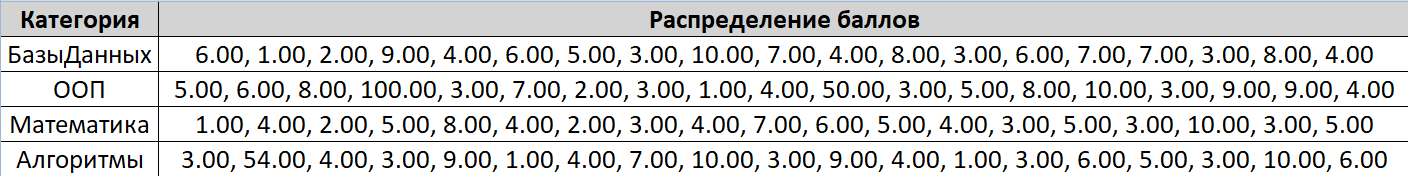


Рисунок Б.10



Рисунок Б.11

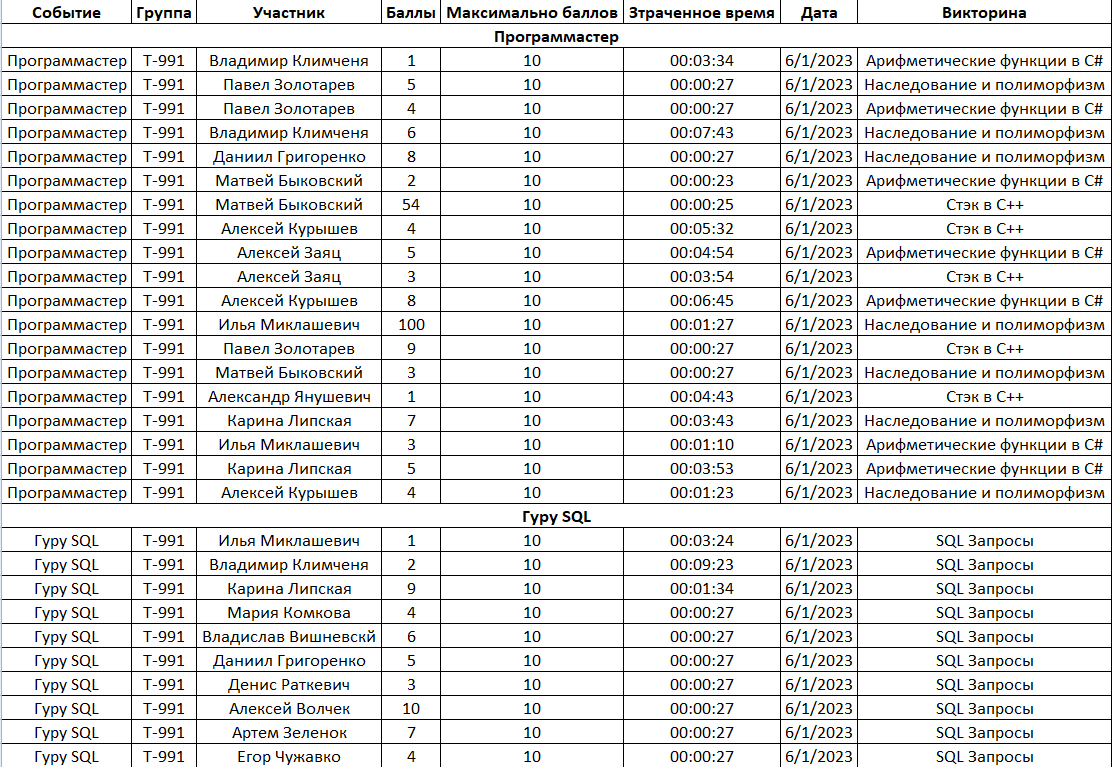


Рисунок Б.12

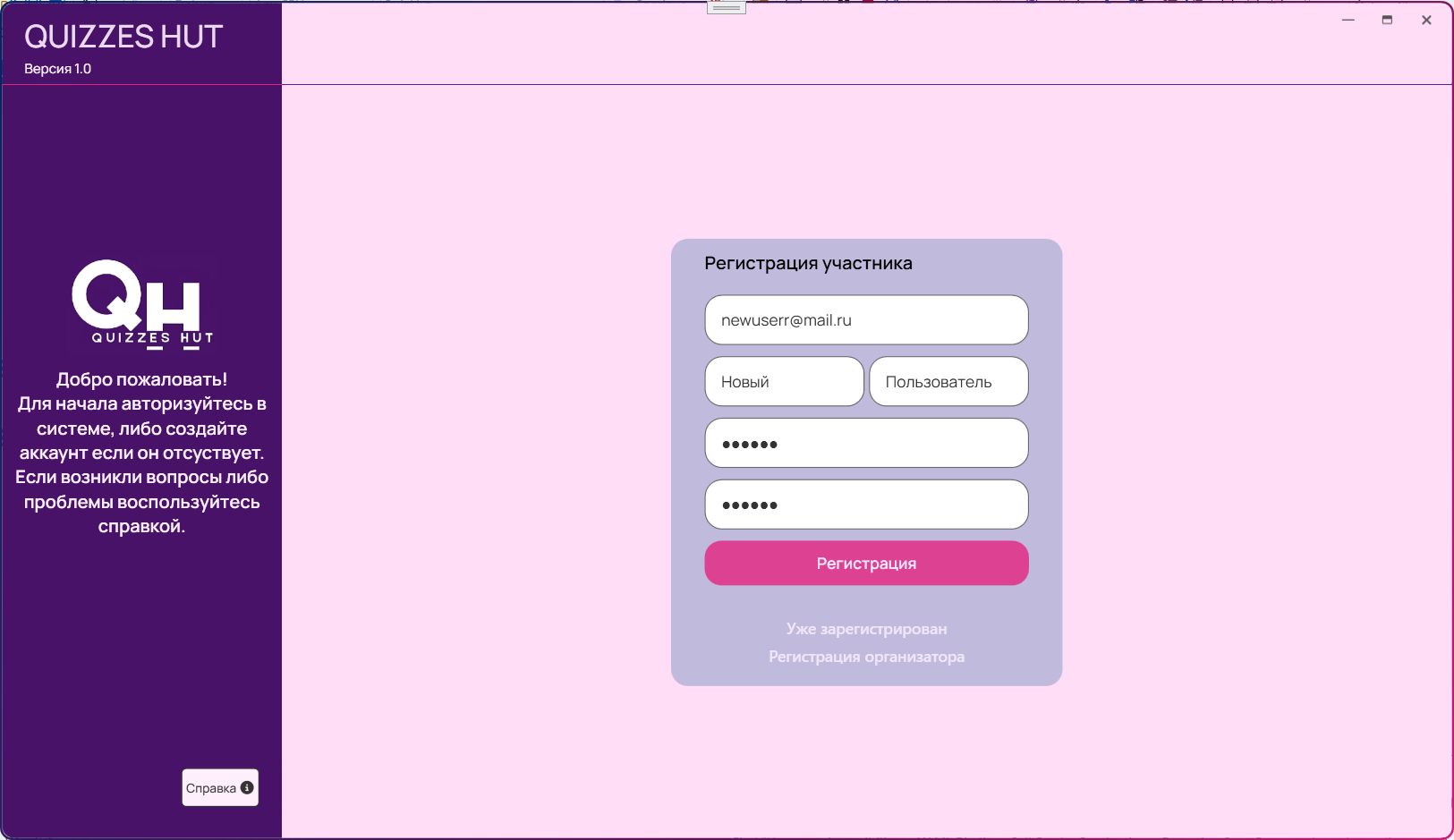


Рисунок Б.13

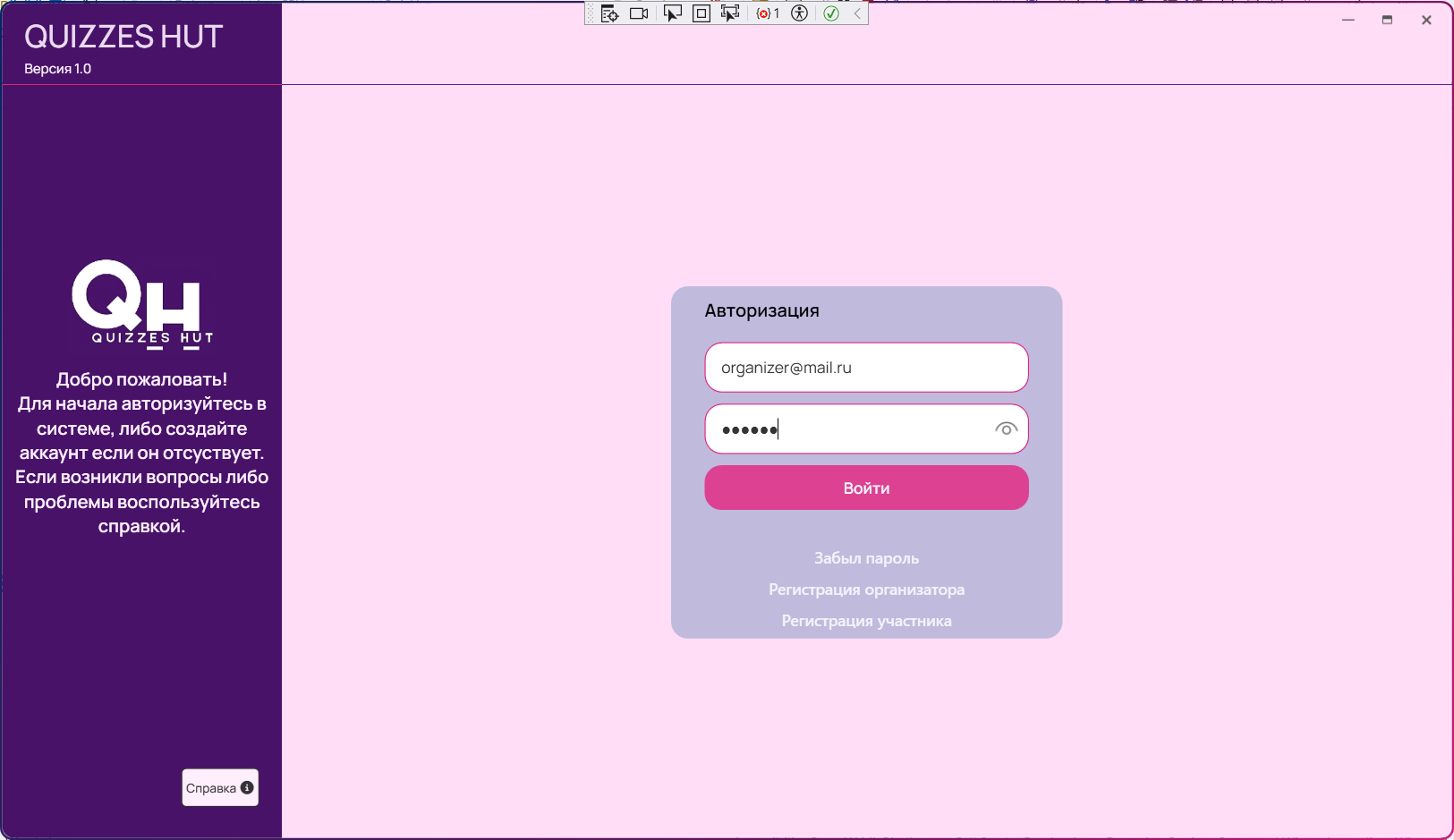


Рисунок Б.14

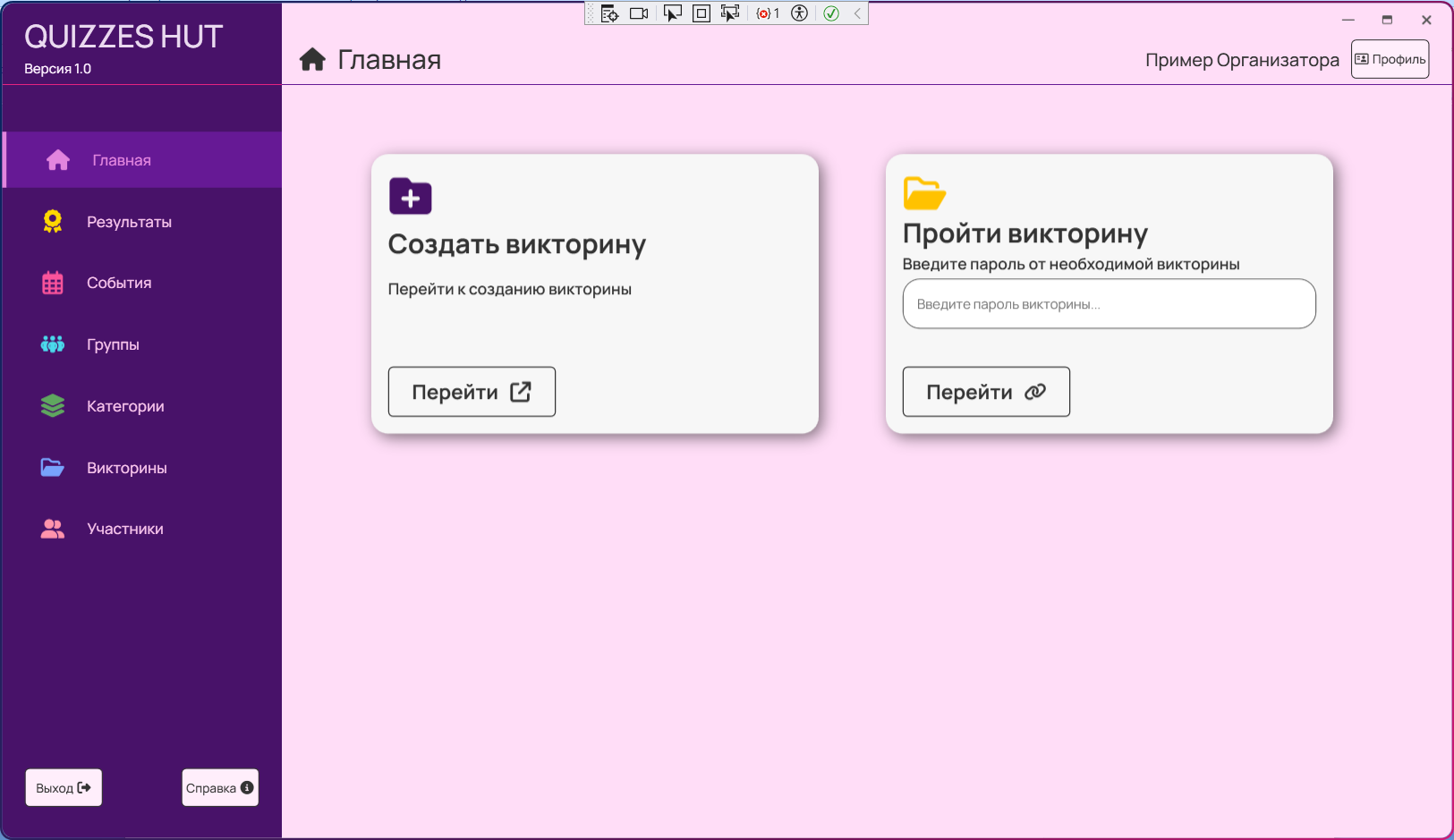


Рисунок Б.15

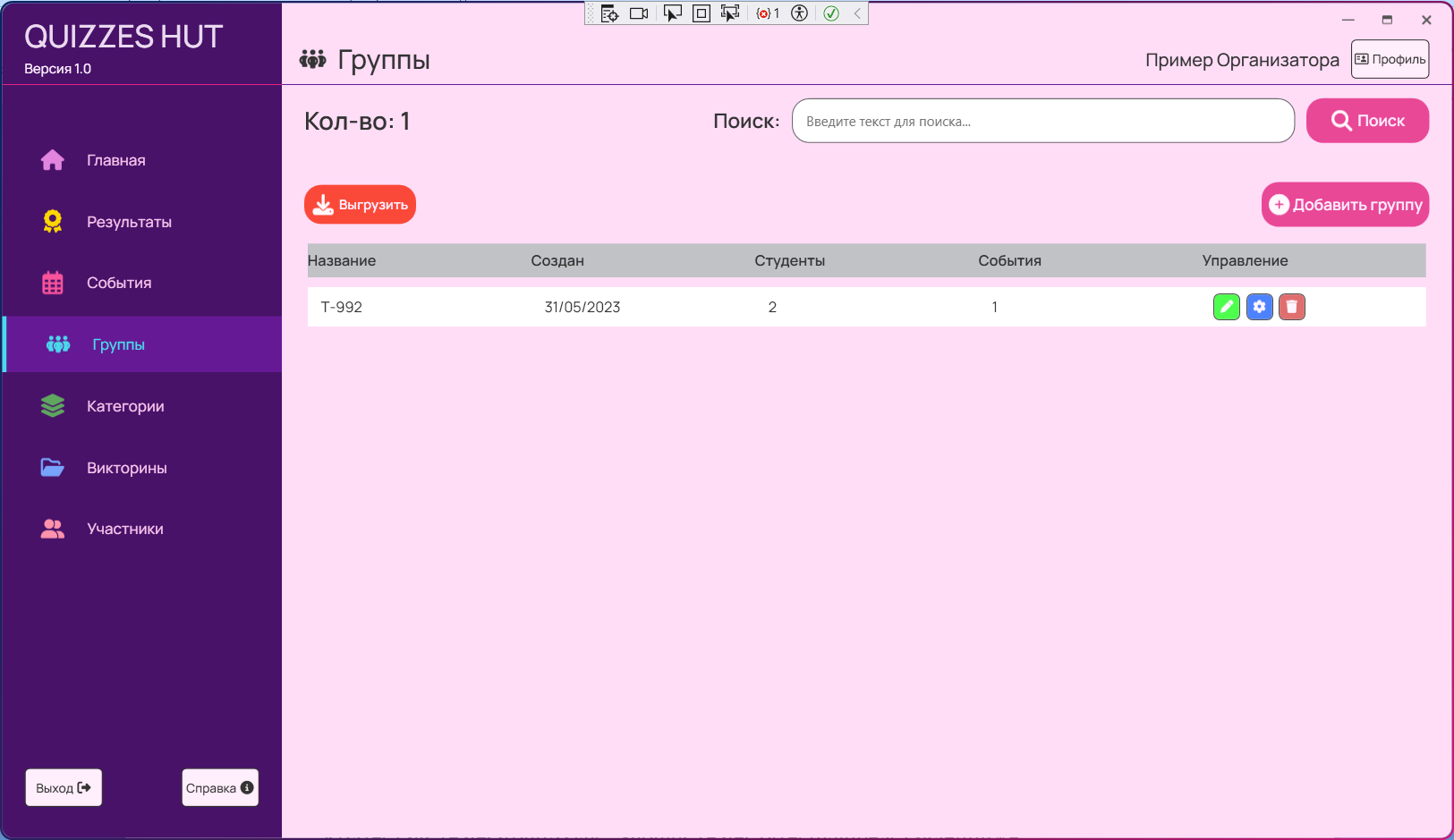


Рисунок Б.16

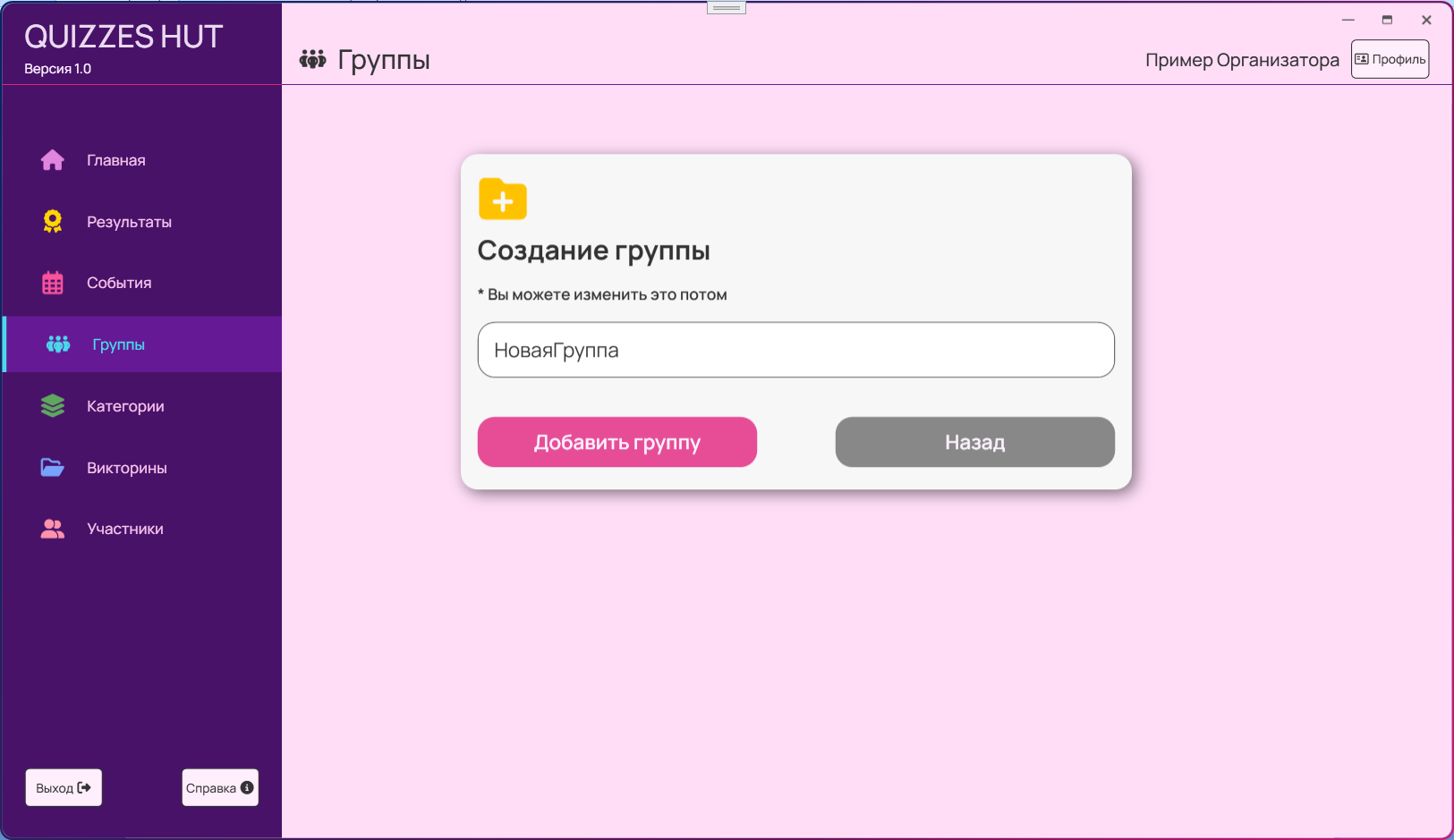


Рисунок Б.17

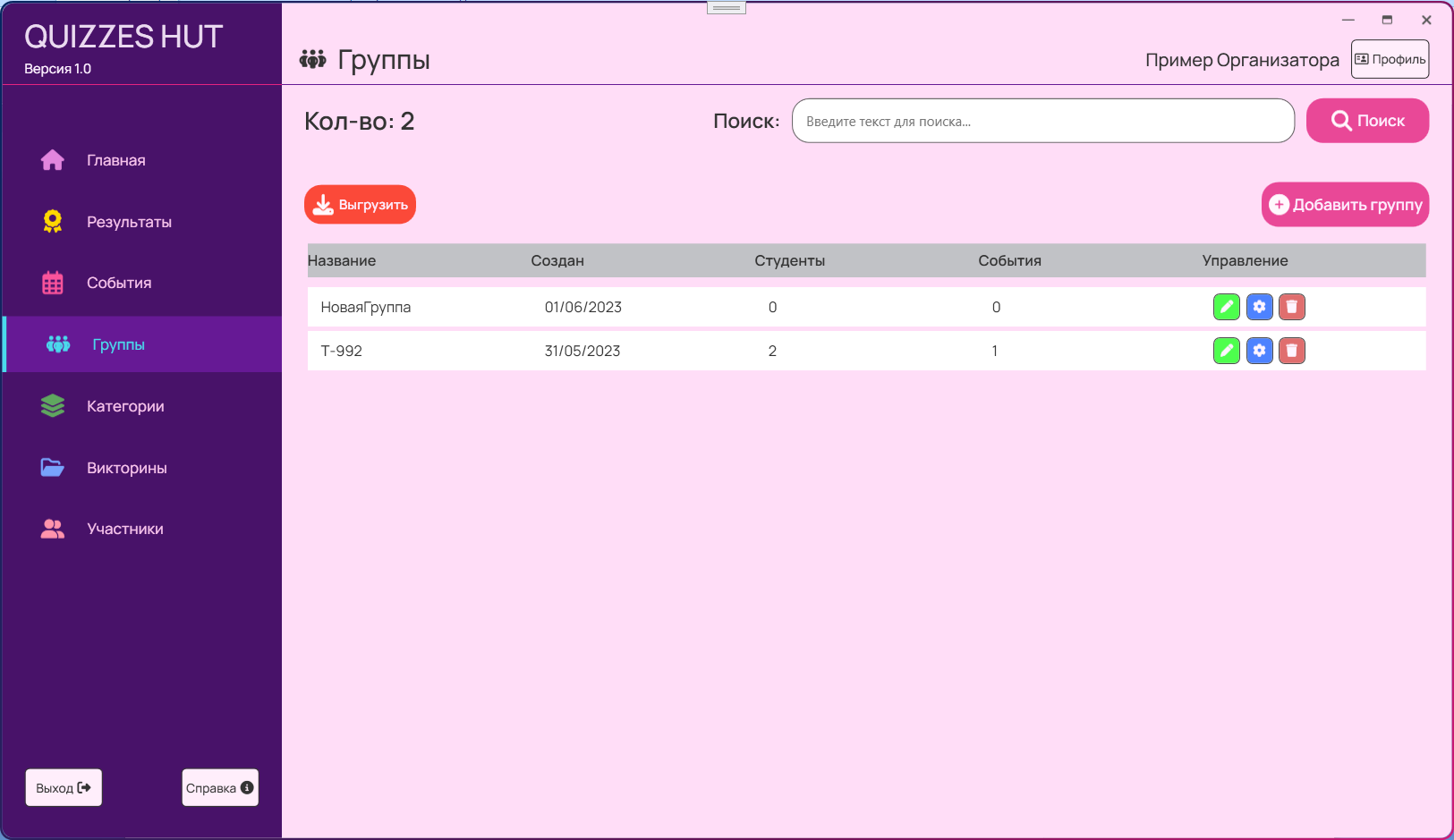


Рисунок Б.18

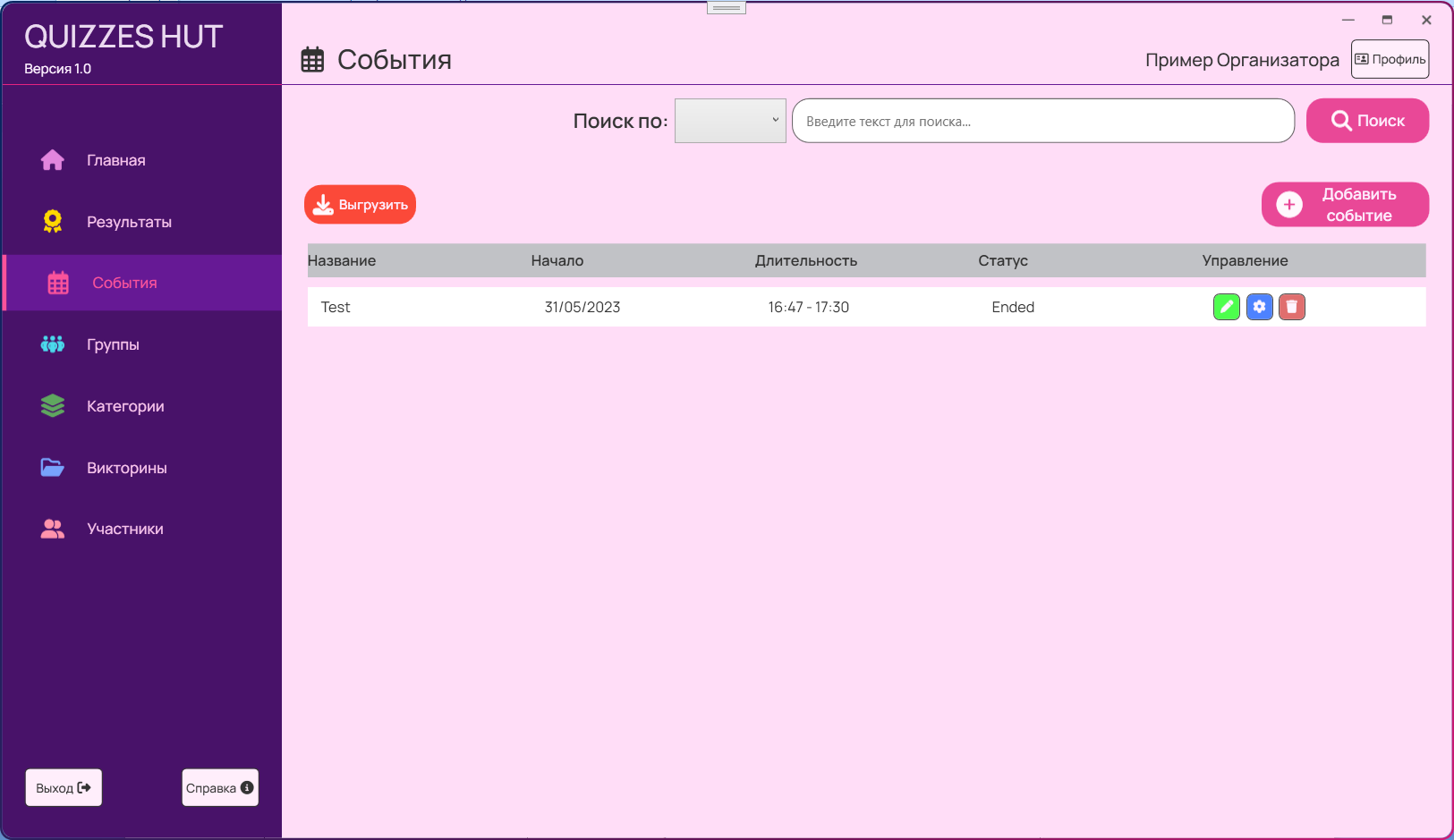


Рисунок Б.19

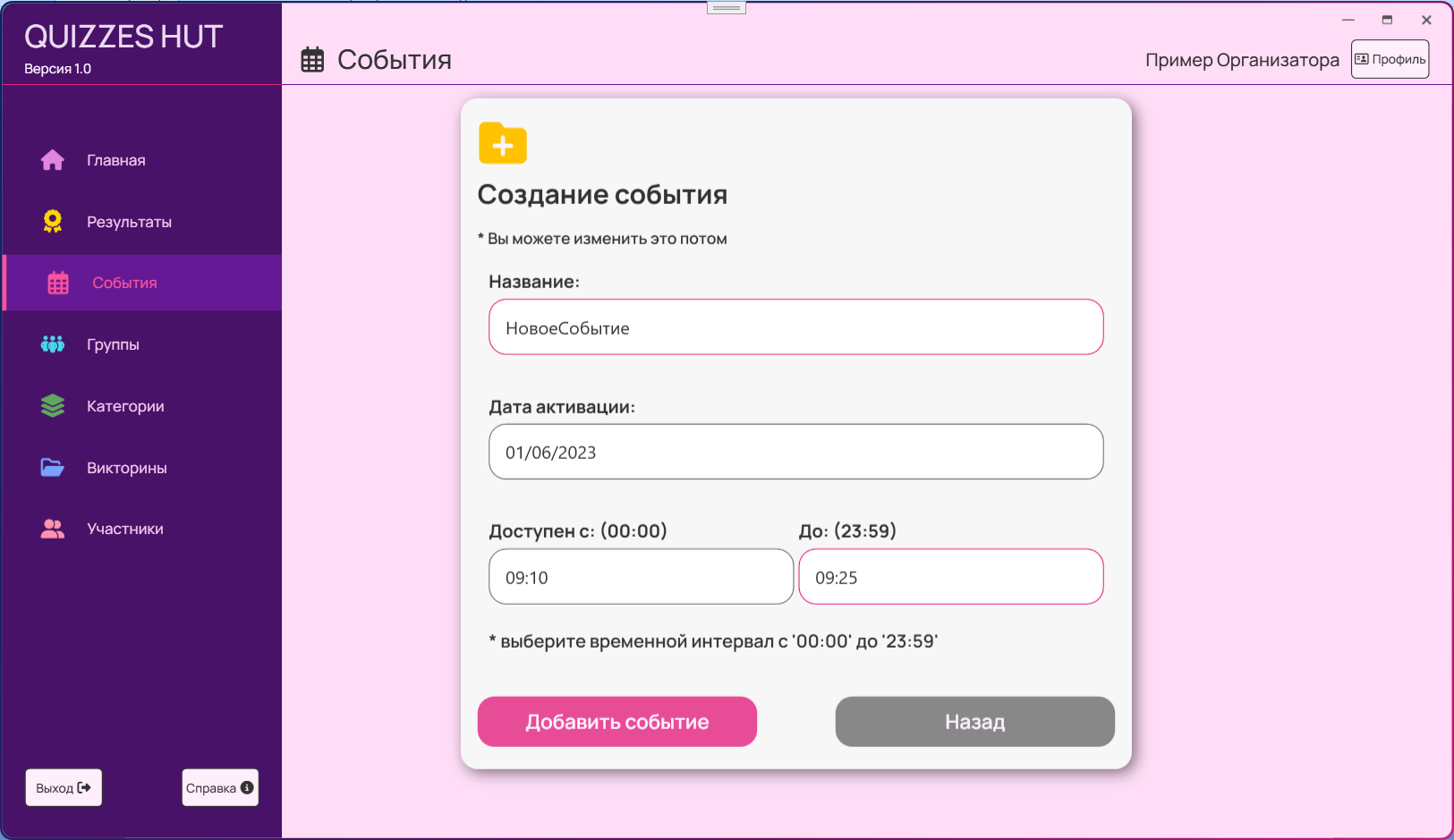


Рисунок Б.20

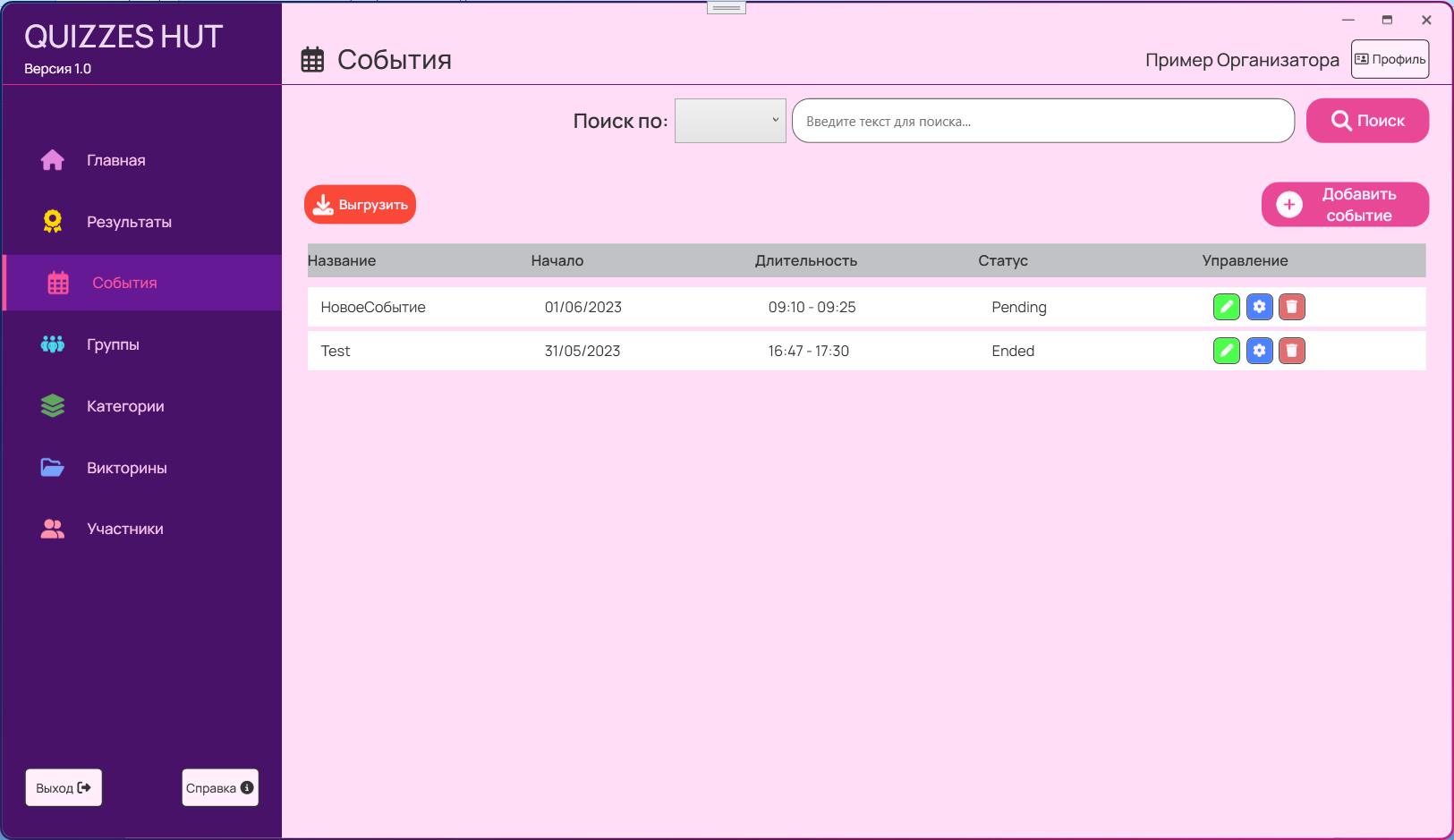


Рисунок Б.21

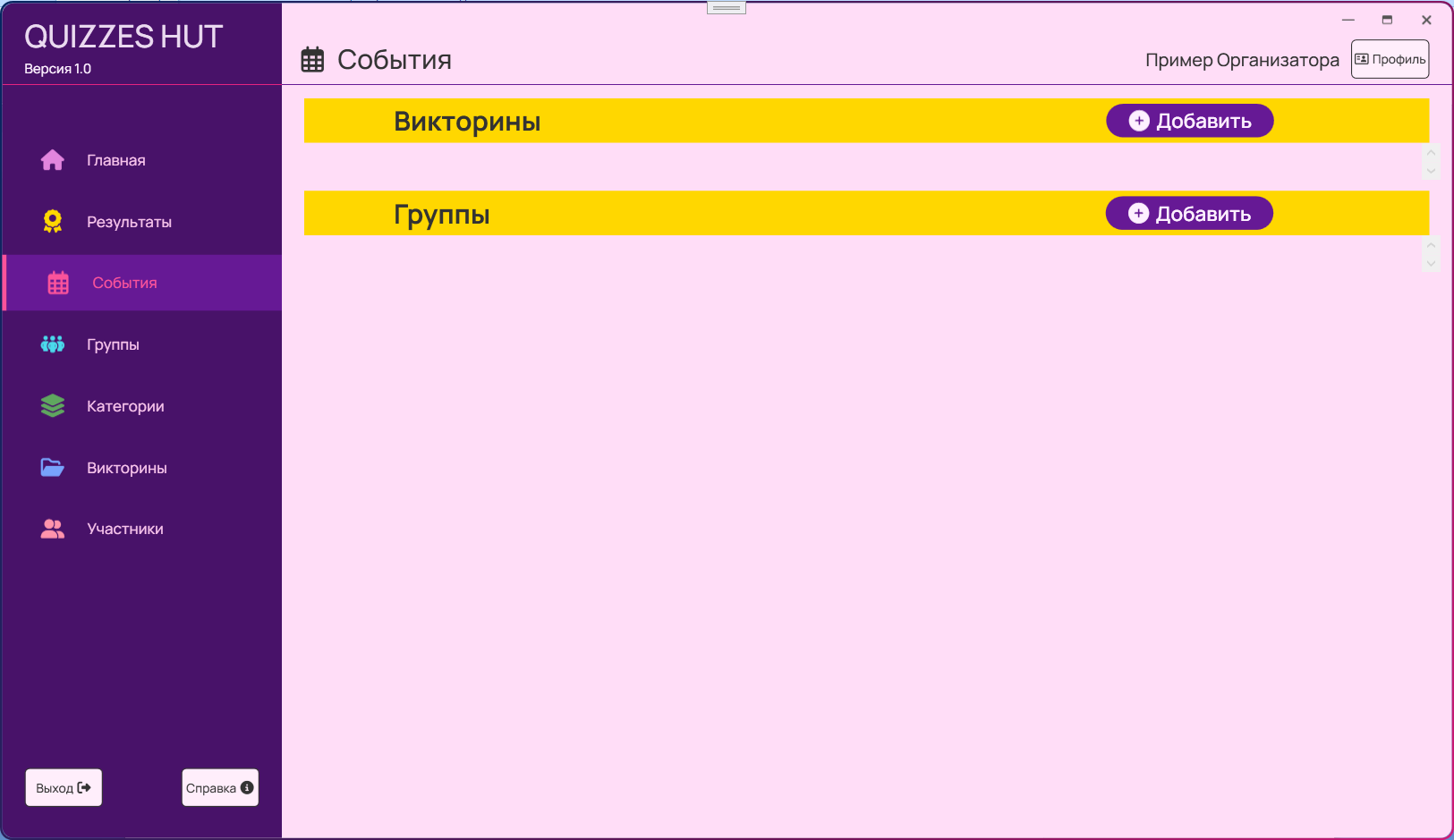


Рисунок Б.22

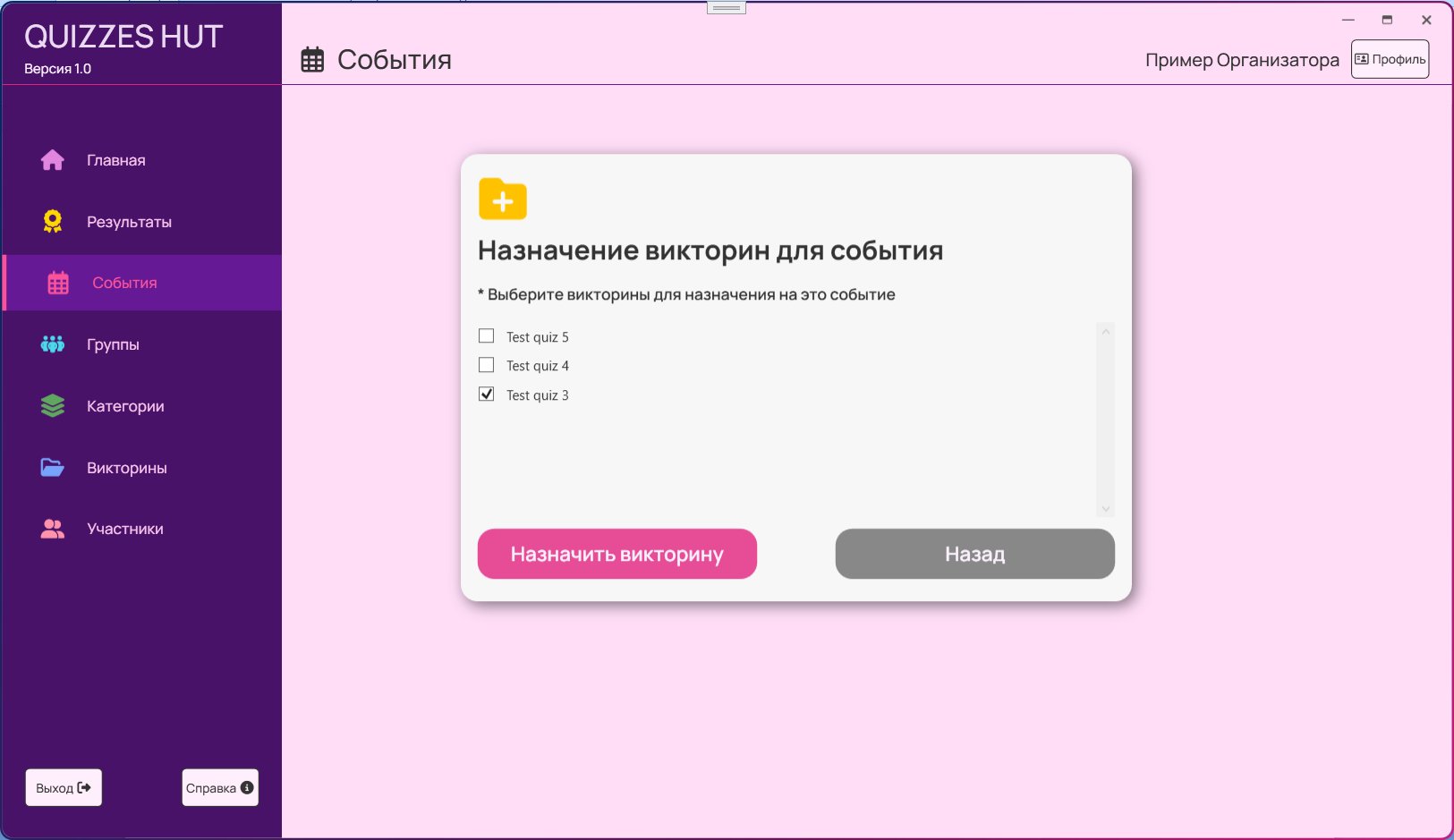


Рисунок Б.23

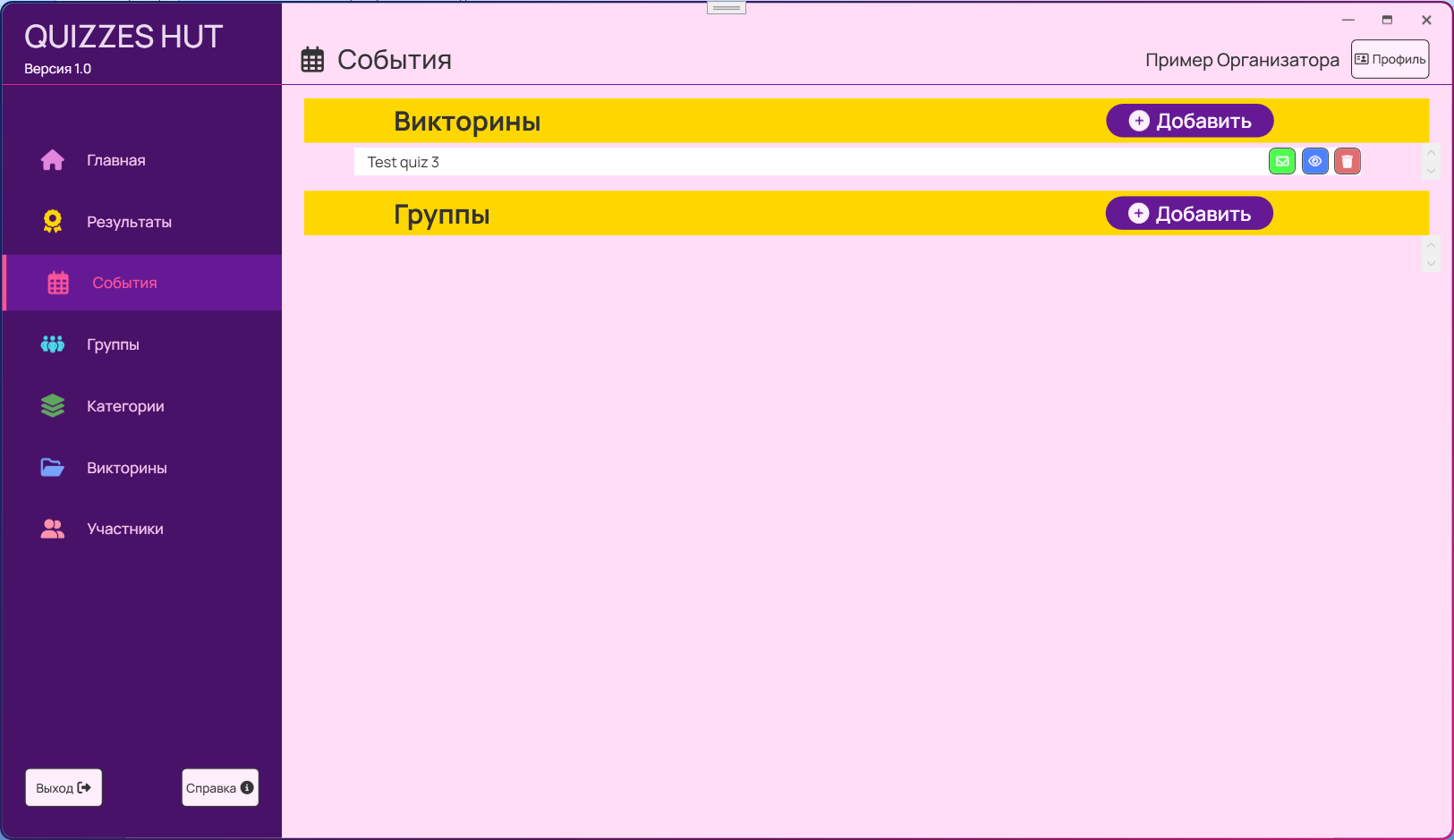


Рисунок Б.24

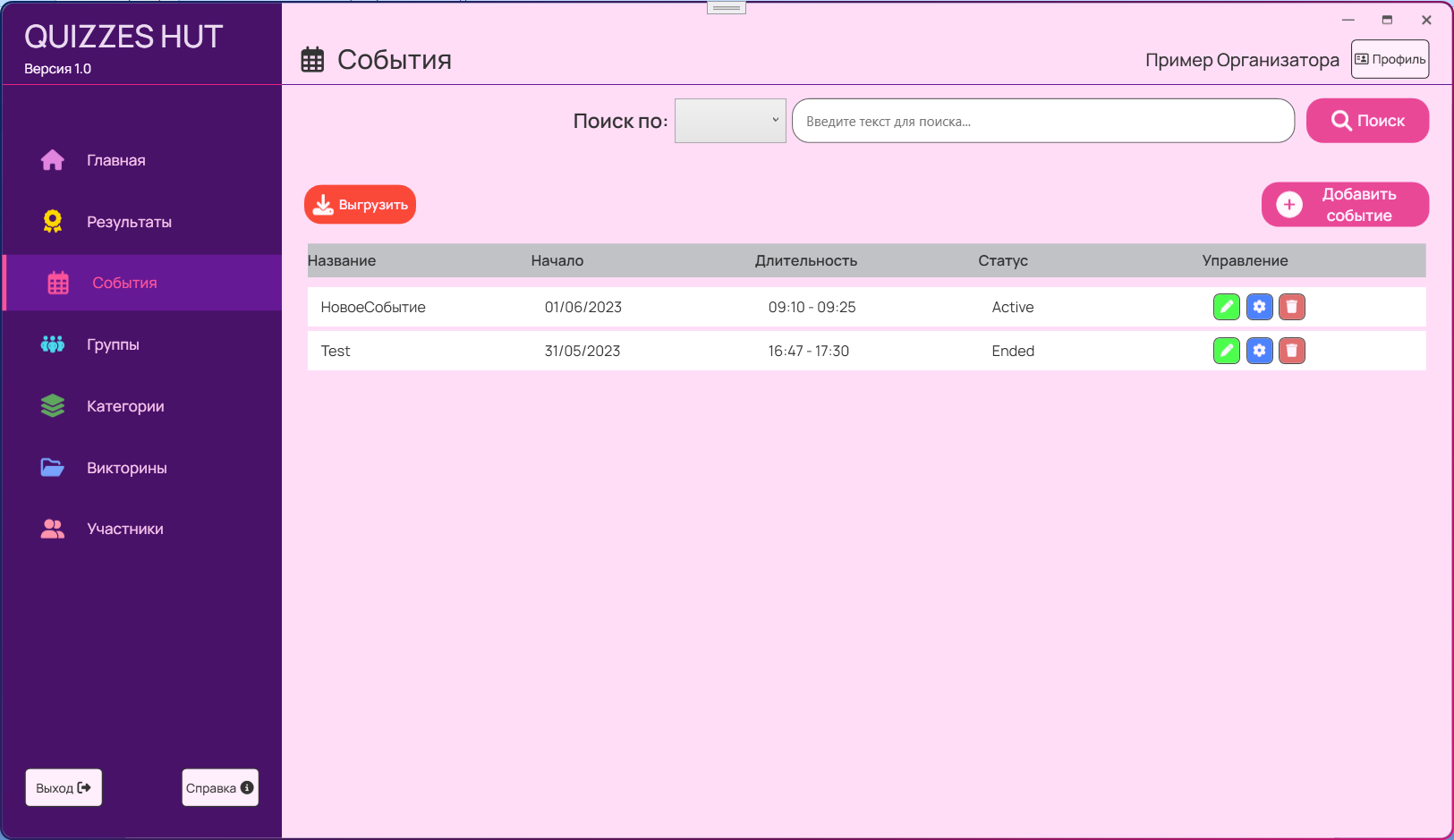


Рисунок Б.25

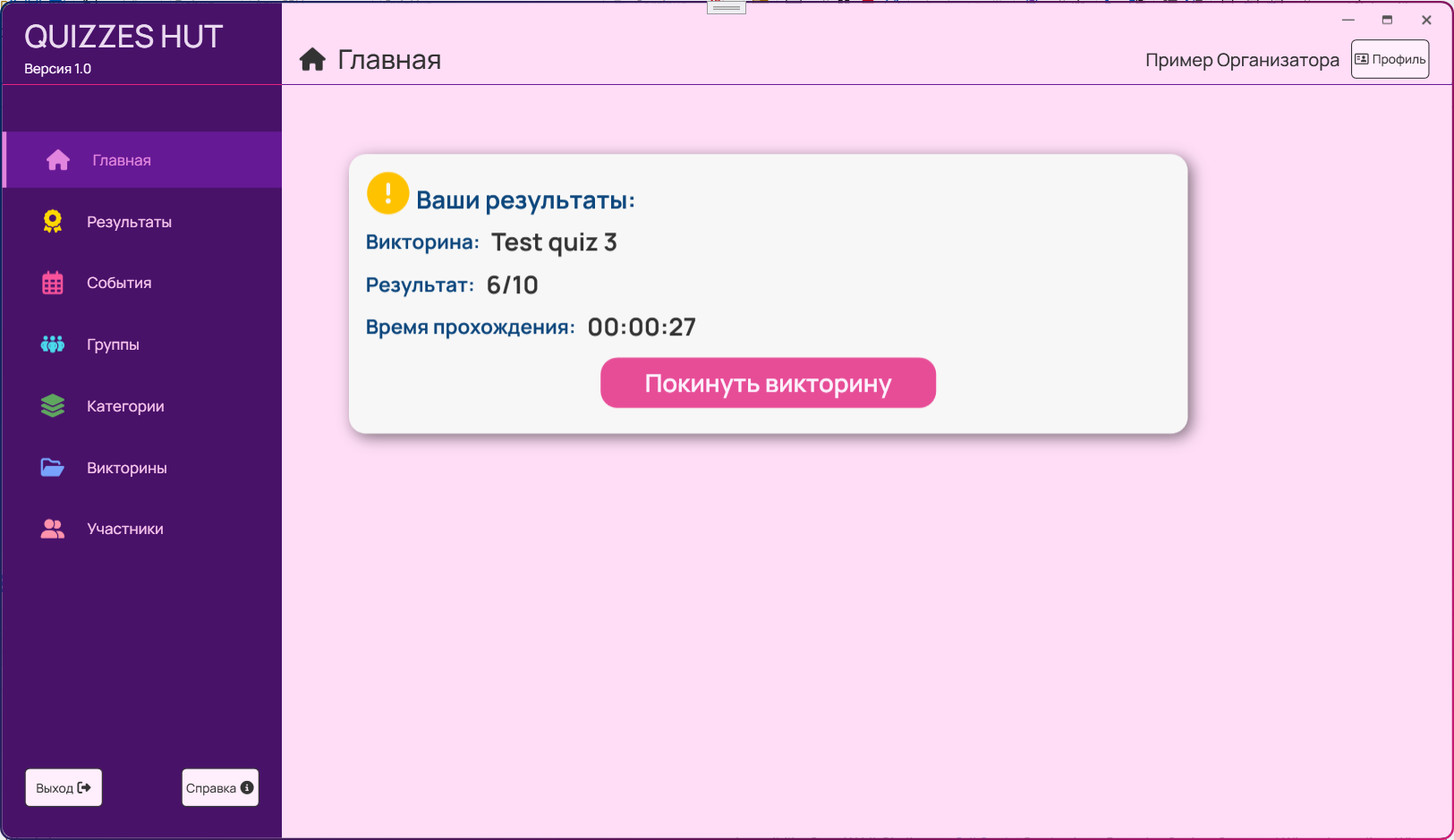


Рисунок Б.26



ДП Т.992011.401 ГЧ

Т. контр.

Н. контр.

Утверд.

КБП

Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.

№докум

Дата

Изм.

Лист

ист

Взам.инв.№

Подп. и дата

*СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ПО АВТОМАТИЗАЦИИ СБОРА И ОБРАБОТКИ ДАННЫХ ДЛЯ ПОДГОТОВКИ И ПРОВЕДЕНИЯ ОЛИМПИАДЫ*

*Диаграмма ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ*

У

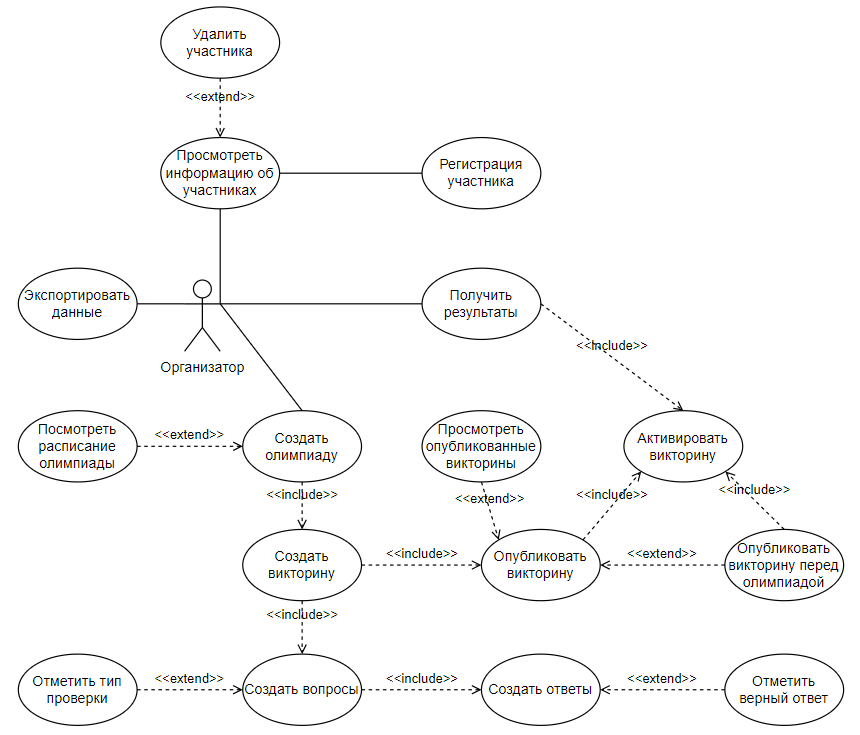
Масса

Лит.

Масштаб

Лист 1

Реценз.



Подпись

У

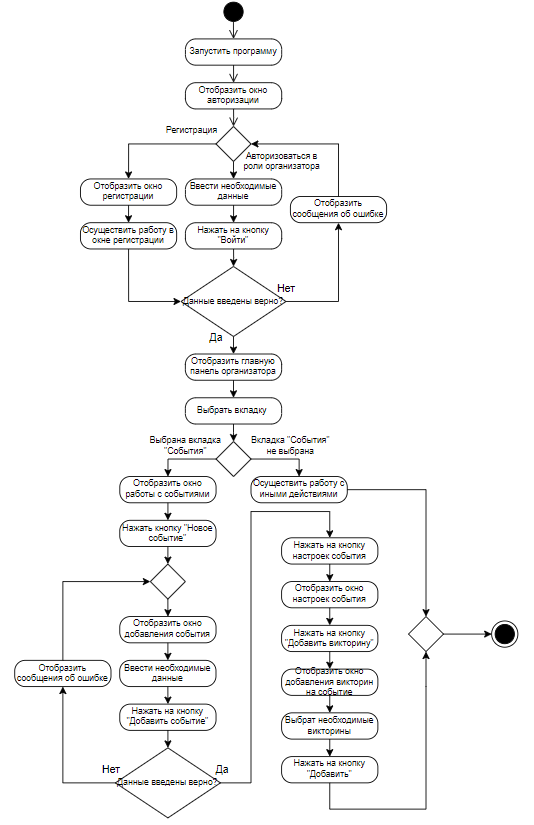
Багласова Т.Г.

Багласова Т.Г.

Басалыга Л.В.

Сватко И.Б.

Быковский М.М.

У

Басалыга Л.В.

Быковский М.М.

Багласова Т.Г.

Багласова Т.Г.

Сватко И.Б.

Подпись

Подпись

ДП Т.992011.401 ГЧ

Т. контр.

Н. контр.

Утверд.

КБП

Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.

№докум

Дата

Изм.

Лист

ист

Взам.инв.№

Подп. и дата

*СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ПО АВТОМАТИЗАЦИИ СБОРА И ОБРАБОТКИ ДАННЫХ ДЛЯ ПОДГОТОВКИ И ПРОВЕДЕНИЯ ОЛИМПИАДЫ*

*Диаграмма деятельности для создания и активации события*

У

Масса

Лит.

Масштаб

Лист 2

Реценз.



ДП Т.992011.401 ГЧ

Т. контр.

Н. контр.

Утверд.

КБП

Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.

№докум

Дата

Изм.

Лист

Взам.инв.№

Подп. и дата

*СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ПО АВТОМАТИЗАЦИИ СБОРА И ОБРАБОТКИ ДАННЫХ ДЛЯ ПОДГОТОВКИ И ПРОВЕДЕНИЯ ОЛИМПИАДЫ*

*Диаграмма*

«*Сущность-связь*»

У

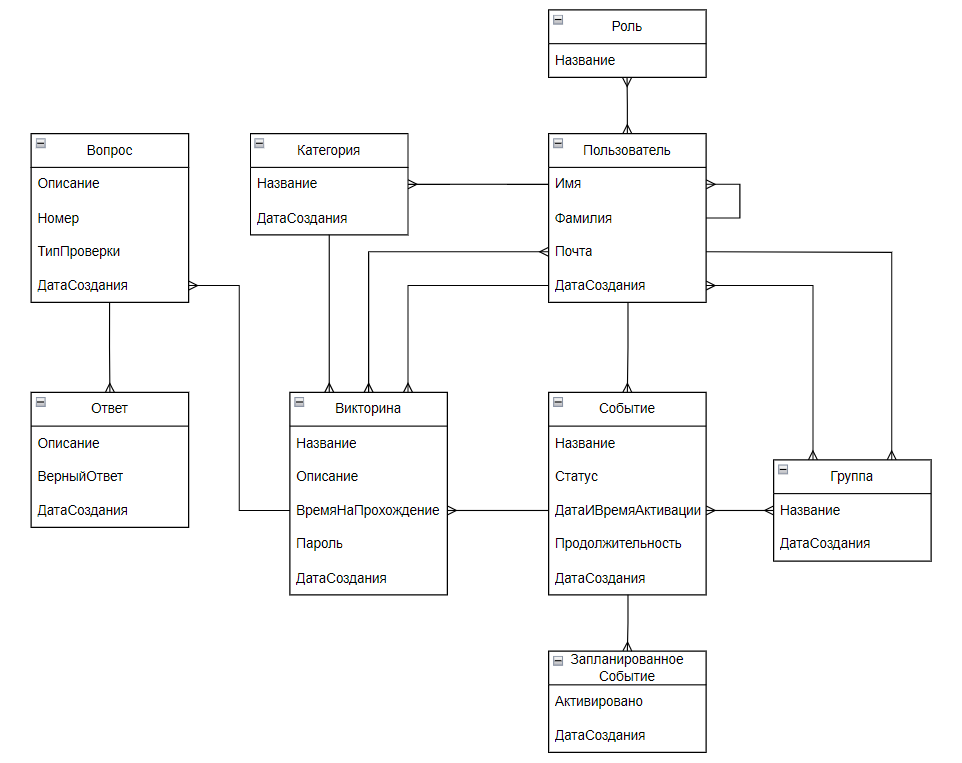
Масса

Лит.

Масштаб

Лист 3

Реценз.



У

Сватко И.Б.

Багласова Т.Г.

Подпись

Быковский М.М.

Басалыга Л.В.

Багласова Т.Г.