Частное учреждение образования

«Колледж бизнеса и права»

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_ И. В. Малафей

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023

СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ ЗНАНИЙ УЧАСТНИКОВ ОЛИМПИАДЫ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ДП Т.992103.401

Председатель цикловой комиссии (Т.Г. Багласова )

Руководитель проекта (К. О. Якимович )

Консультант по экономической части (Е. А. Андреева )

Консультант по охране труда (В. С. Кудласевич )

Учащийся (А. С. Парфянович)

Рецензент ( )

2023

СОДЕРЖАНИЕ

Изм.

Лист

№докум.

Подпись

Дата

Лист

3

ДП Т.992103.401 ПЗ

Разраб.

Парфянович А.С.

Провер.

Сватко И.Б.

Т. контр.

*Багласова Т.Г.*

Н. контр.

Басалыга Л.В.

Утверд.

Багласова Т.Г.

*СОздание программного средства для проведения тестирования знаний участников олимпиады*

Лит.

Листов

КБП

103

у

у

[Введение 4](#_Toc136815669)

[1 Описание задачи 6](#_Toc136815670)

[1.1 Анализ предметной области 6](#_Toc136815671)

[1.2 Постановка задачи 8](#_Toc136815672)

[2 Проектирование системы 11](#_Toc136815673)

[2.1 Требования к приложению 11](#_Toc136815674)

[2.2 Проектирование модели 13](#_Toc136815675)

[2.3 Проектирование структуры базы данных 13](#_Toc136815676)

[2.4 Концептуальный прототип 15](#_Toc136815677)

[3 Описание реализации программного средства 21](#_Toc136815678)

[3.1 Инструменты разработки и применяемые технологии 21](#_Toc136815679)

[3.2 Порядок авторизации пользователей 22](#_Toc136815680)

[3.3 Организация данных 23](#_Toc136815681)

[3.4 Функции: логическая и физическая организация 29](#_Toc136815682)

[3.5 Входные и выходные данные 32](#_Toc136815683)

[3.6 Функциональное тестирование 33](#_Toc136815684)

[3.7 Описание справочной системы 48](#_Toc136815685)

[4 Применение 49](#_Toc136815686)

[4.1 Назначение программного средства 49](#_Toc136815687)

[4.2 Условия применения 49](#_Toc136815688)

[5 Охрана окружающей среды 51](#_Toc136815689)

[5.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда 51](#_Toc136815690)

[5.2 Разработка мер по нормализации и защите от источников ионизирующих излучений при эксплуатации ПЭВМ 54](#_Toc136815691)

[5.3 Пожарная безопасность 58](#_Toc136815692)

[5.4 Охрана окружающей среды 60](#_Toc136815693)

[6 Экономический раздел 62](#_Toc136815694)

[6.1 Технико-экономическое обоснование разработки программного средства 62](#_Toc136815695)

[6.2 Составление плана по разработке программного средства 62](#_Toc136815696)

[6.3 Определение цены программного средства 63](#_Toc136815697)

[6.4 Расчет экономического эффекта у разработчика и пользователя программного средства 69](#_Toc136815698)

[6.4.1 Экономический эффект у разработчика программного средства 69](#_Toc136815699)

[6.4.2 Экономический эффект у пользователя программного средства 70](#_Toc136815700)

[Заключение 72](#_Toc136815701)

[Список информационных источников 73](#_Toc136815702)

[Приложение А Текст программы 75](#_Toc136815703)

[Приложение Б Результаты работы программы 78](#_Toc136815706)

**Введение**

Предметом деятельности любого учреждения образования является обучение учащихся, тестирование их знаний, полученных в процессе обучения, а также тестирование уровня интеллекта для оценки умственных способностей и нестандартного мышления учащихся. Сегодня прохождение электронного тестирования становится все более и более востребованным, поскольку представляет собой менее трудоемкий процесс, а анализ этой информации требует меньшего количества усилий и времени. В связи с этим, создание программного средства для проведения тестирования знаний является актуальным и затрагивает практически любое учреждение образования.

Целью дипломного проекта является создание программного средства для проведения тестирования знаний участников олимпиады, которое будет предоставлять следующие возможности: регистрация участников олимпиады и прохождение тестов; выгрузка данных об итогах проведения олимпиады; проведение тестирования с возможностями

* создания/редактирования различных видов тестовых заданий;
* задания критериев оценки тестовых заданий;
* задания ограничений на тестирование;
* просмотра информации об участниках олимпиады.

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 в области охраны труда, о требованиях пожарной безопасности и охраны окружающей среды. Кроме того, описывается разработка мер по нормализации и защите от источников ионизирующих излучений при эксплуатации персональной электронно-вычеслительной машины (ПЭВМ).

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 формы выходных документов.

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

# Описание задачи

## Анализ предметной области

В современной системе образования олимпиадные мероприятия – один из способов для учащихся продемонстрировать свои знания и навыки.

Олимпиада – это академическое мероприятие, которое проверяет знания и понимание учащихся по определенному предмету или области обучения.

Олимпиады организуются на различных уровнях, от местного до национального и международного, и могут быть открыты для учащихся из различных учебных заведений. Организаторами олимпиады могут быть различные учебные заведения с целью продвижения и признания академических достижений.

Одним из примеров олимпиад по программированию 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.

Для проверки знаний учащихся при проведении олимпиад в области программирования используются практические задания и задания в виде тестов, которые могут быть самыми разнообразными и иметь различную сложность: задания, проверяющие способность участника читать и понимать код; задания, требующие от участников написать собственный код для решения поставленной задачи; задания, требующие от участников оптимизировать существующий код для повышения производительности.

Одной из ключевых особенностей соревнований в области программирования является возможность использования онлайн-компилятора для выполнения практических заданий с использованием необходимого языка программирования, который позволяет участникам проверить свой код в режиме реального времени. Кроме того, участникам обычно предоставляется возможность приложить готовый файл с решением практических заданий.

Типы тестовых заданий на олимпиадах по программированию могут быть самыми разными:

* задания с одним вариантом ответа;
* задания с несколькими вариантами ответа;
* задания со свободным ответом;
* задания на сопоставление ответов.

Задания с одним вариантом ответа –­ тип тестовых заданий, в котором участника просят выбрать только один правильный ответ из списка вариантов.

Пример теста с одним вариантом ответа представлен на рисунке 1.1.

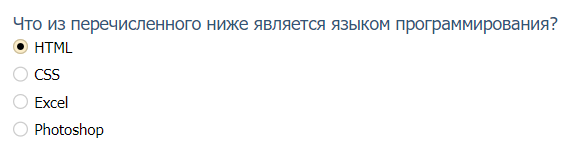


Рисунок 1.1

Задания с несколькими вариантами ответов –­ тип тестовых заданий, в котором участнику предлагается выбрать один или несколько правильных ответов из списка вариантов. Пример теста с несколькими вариантами ответов представлен на рисунке 1.2.

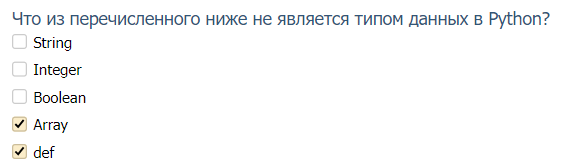


Рисунок 1.2

Задания со свободным ответом –­ тип тестовых заданий, в котором участника просят дать развернутый ответ на поставленный вопрос или определение.

Пример теста со свободным ответом представлен на рисунке 1.3.

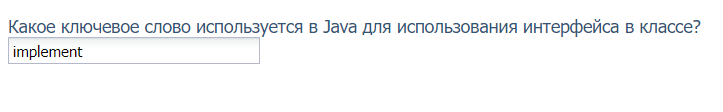


Рисунок 1.3

Задания на сопоставление –­ тип тестовых заданий, в котором участника просят сопоставить элементы из двух колонок на основе заданных критериев, например, определения с терминами или понятия с примерами.

Пример теста на сопоставление представлен на рисунке 1.4.

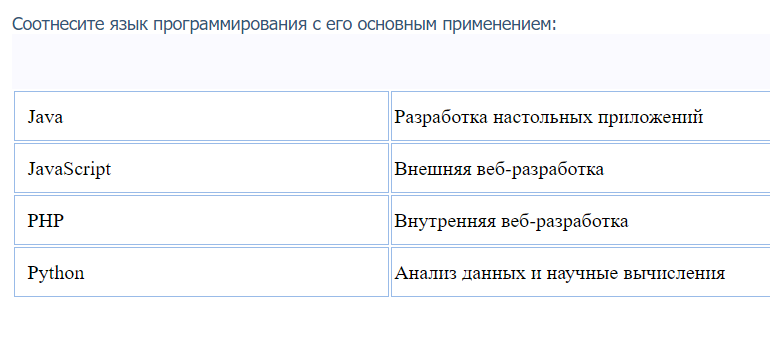


Рисунок 1.4

При проведении тестирования особое внимание уделяется и критериям оценки заданий, например, в зависимости от типа задания. Для заданий с одним правильным ответом, как правило, выставляется балл только за правильный ответ, для заданий с несколькими правильными ответами ­ за каждый правильный ответ, для заданий со свободным ответом – на основе критериев, заданных организатором олимпиады, а для заданий на сопоставление ­ за каждую правильно связанную пару. Существуют и другие алгоритмы оценки заданий, однако описанный выше применяется наиболее часто.

Выполнение тестового задания требует определенного времени. Общее время тестирования определяется количеством и сложностью заданий. Важно уделять внимание этому вопросу так как неверно установленное время тестирования не позволяет тестовым заданиям достичь своей цели ­ проверить знает ли  испытуемый тот или иной элемент.

При проведении тестирования знаний участников олимпиады не менее важным является предоставление обратной связи по результатам работы. Предоставление конструктивной обратной связи помогает

* участникам определить свои сильные и слабые стороны, понять, где они ошиблись;
* мотивировать участников приложить дополнительные усилия и улучшить свои результаты.

## Постановка задачи

Частное учреждение образования «Колледж бизнеса и права» в настоящее время для проверки знаний участников олимпиад по программированию использует бесплатный интернет-сервис «MasterTest». Функционал, доступный через интернет-сервис «MasterTest», не полностью отвечает всем необходимым требованиям учреждения образования:

* отсутствует возможность проверки знаний, требующих от участников написать собственный код для решения поставленной задачи;
* отсутствует возможность приложить готовый файл с решением практических заданий;
* отсутствует онлайн-компилятор для выполнения практических заданий с использованием необходимого языка программирования;
* не обеспечивает обратную связь с участниками олимпиады по завершении теста.

Разрабатываемое программное средство в рамках дипломного проекта будет представлять собой удобную и простую в использовании платформу для проверки знаний участников олимпиады и будет отвечать конкретным потребностям колледжа при проведении олимпиад различного уровня и масштаба:

* реализована работы программы в режиме различных пользователей: «Организатор», «Участник»;
* реализована возможность регистрации участников олимпиады (фамилия, имя, отчество, учреждение образования, курс, возраст, адрес электронной почты, контактный телефон);
* для роли «Организатор» реализованы следующие возможности: создание/редактирование различных видов тестовых заданий (с одним правильным ответом, с несколькими правильными ответами, свободный ответ, сопоставление); задание критериев оценки тестовых заданий; задание ограничений на тестирование (время на тест, сроки проведения, число попыток); просмотр информации об участниках олимпиады; выгрузка данных об итогах проведения олимпиады;
* для роли «Участник» реализована возможность прохождения тестов.

Результаты тестирования будут рассчитываться автоматически на основе ответов, предоставленных участниками: для заданий с одним правильным ответом будет начисляться балл только за правильный ответ, для заданий с несколькими правильными ответами ­ за каждый правильный ответ, для заданий с открытым ответом – на основе сложности вопроса, а для заданий на сопоставление ­ за каждую правильно связанную пару. Общее время тестирования будет определяться количеством и сложностью заданий. Результаты будут отображаться в личном кабинете участника.

На данный момент существуют аналоги разрабатываемого программного средства:

* онлайн-платформа для конкурсов по кодированию и программированию «HackerRank»;
* онлайн-платформа для конкурсного программирования «Codeforces»;
* онлайн-платформа для проведения соревнований по программированию и оценки навыков «TopCoder»;
* онлайн-платформа для проведения конкурсов по программированию «Braim».

Каждая из этих платформ имеет свои преимущества и недостатки: диапазон доступных типов тестовых заданий, простота использования, стоимость использования. Разрабатываемое программное средства в рамках дипломного проекта обеспечит баланс между всеми факторами, предоставляя собой удобный способ проверки знаний в рамках проведения олимпиады в Частном учреждении образования «Колледж бизнеса и права».

# 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.

Для минимизации умственной нагрузки на пользователей при выборе нужного раздела программного средства слева в главном окне будет разработано навигационное меню. Элементы управления навигационного меню, помимо текста, будут содержать векторные изображения (иконки) с сервиса «FontAwesome», которые помогут целевой аудитории в необходимой степени понимать какое действие последует при взаимодействии с элементом и ориентироваться в программе.

В качестве шрифта интерфейса будет использоваться геометрический шрифт без засечек «Manrope» в различных вариантах, который придаст программе современность, эстетическую привлекательность и профессиональность. Шрифт «Manrope» имеет различные варианты начертания, что позволит использовать его для заголовков, подзаголовков и основного текста.

Цветовая палитра будет представлена сочетающимся между собой набором цветов: светло-фиолетовый, светло-розовый, ярко-розовый, светло-зеленый, золотой, синий и так далее.

Все выводимые сообщения должны быть четким, лаконичным и информативным, передавая пользователю необходимую информацию. Оформление должно быть визуально привлекательным, а текстовый блок должен выделяться на фоне окружающих элементов интерфейса. Красный цвет текста и размер шрифта 14 пунктов должны быть одинаковыми для всех выходных сообщений, что обеспечит единообразие и сделает их легко узнаваемыми для пользователя. Сообщения будут размещены рядом с соответствующими элементами управления, так как должны обеспечивать пользователям немедленную обратную связь о результатах их действий. Кроме того, язык, используемый в выходных сообщениях, должен быть удобным и простым для понимания, избегая технического жаргона или сложной терминологии.

Форматы ввода данных должны быть интуитивно понятными, удобными для пользователя и согласованными во всей программе. Каждое поле ввода должно быть четко обозначено, с использованием описательного текста, который точно передает тип требуемых данных. Метки должны располагаться внутри поля ввода, чтобы пользователи могли легко понять, что нужно ввести.

Сами поля ввода должны быть соответствующего размера, чтобы пользователю было достаточно места для ввода необходимых данных. Кроме того, вводимые данные должны быть проверены с помощью встроенных средств валидации и фильтрации (длина строки, диапазон, регулярное выражение). Если вводимые данные неверные, пользователю должны выдаваться соответствующие сообщения об ошибках, объясняющие характер ошибки и способы ее устранения.

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 в виде диаграммы структурного анализа и проектирования (SADT), которая представляет функциональные требования, включающие методы, правила и процедуры. Она используется для создания функциональной модели объекта из определенной области и отображения его функциональной структуры, то есть действий, которые он выполняет, и связей между ними. Диаграмма SADT нулевого уровня представлена на рисунке 2.1.

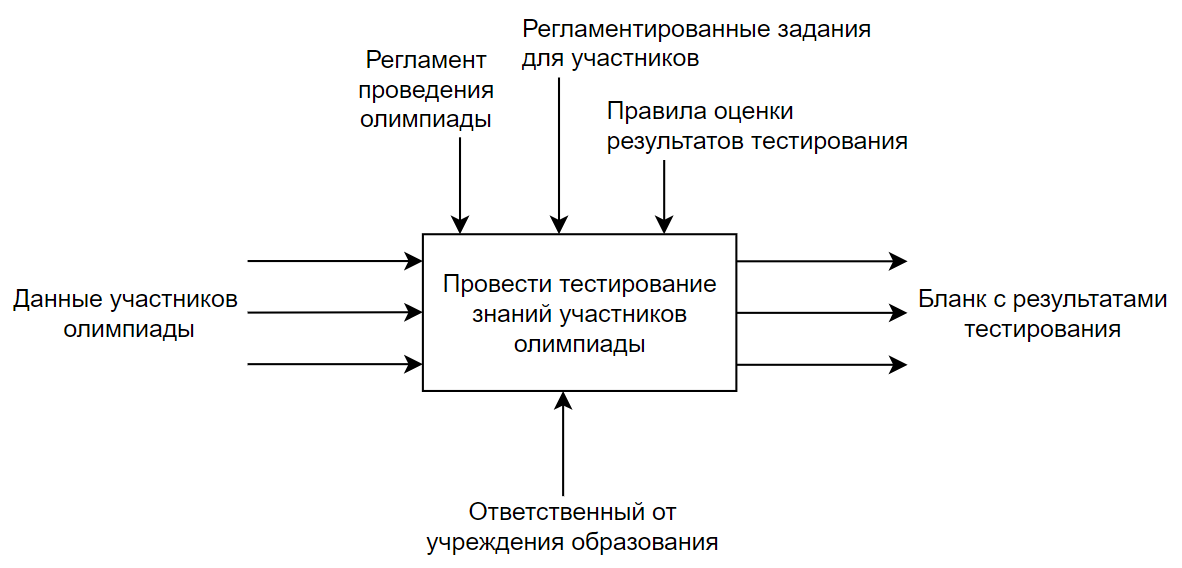


Рисунок 2.1

## Проектирование модели

Этап проектирования при разработке информационной системы включает моделирование данных. В процессе моделирования данных различные компоненты, такие как диаграммы вариантов использования, диаграммы классов, диаграммы последовательности и деятельности представляются в локальных моделях. Они позволяют разработчикам создать структурированную основу, которая точно представляет требования к данным информационной системы. В конечном итоге, хорошо выполненный этап моделирования данных способствует общему успеху информационной системы, обеспечивая эффективное управление данными и функциональность системы.

Диаграммы вариантов использования служит для отражения отношений между актерами и прецедентами. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. Основная цель использования таких диаграмм – визуализация возможностей пользователя в программном средстве.

Основными функциями средства для проведения тестирования знаний участников олимпиады являются:

* просмотр инфомарции о пользователях;
* прохождение викторины;
* просмотр личных результатов;
* просмот событий.

Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма деятельности – это поведенческая диаграмма, которая представляет последовательность действий или поток управления в системе. В диаграмме деятельности шаги будут представлены в виде серии связанных узлов. Каждый узел будет представлять собой действие или точку принятия решения, а стрелки будут указывать на поток управления между ними. Благодаря четкому и организованному представлению этих шагов диаграмма деятельности является ценным инструментом для разработчиков, дизайнеров и заинтересованных сторон, позволяющим понять процесс работы приложения.

Диаграмма деятельности для прохождения викторины представлена в графической части на листе 2.

## Проектирование структуры базы данных

Диаграмма «Сущность-связь» (Entity-relationship diagram (ERD)) – это графическое представление сущностей и их связей друг с другом. Диаграмма используется при проектировании баз данных для иллюстрации отношений между таблицами, сущностями, атрибутами и ключами.

ERD состоят из трех основных компонентов: сущностей, атрибутов и отношений. Сущности – это объекты или понятия в системе, такие как пользователи или тесты. Атрибуты описывают характеристики сущности, например, имя или электронная почта пользователя. Отношения описывают связи между сущностями, например, пользователь проходит тест.

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;
* 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 – систему меню и окон.

При запуске программы будет отображаться главное окно, поверх которого будет представлено окно входа в систему.

Макет окна входа в систему представлен на рисунке 2.2.

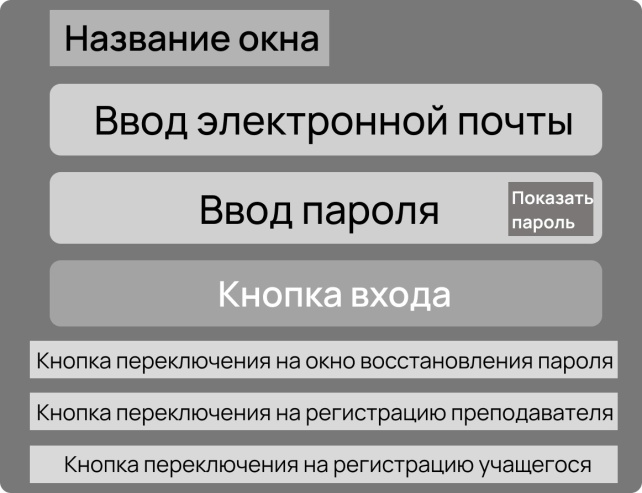


Рисунок 2.2

В данном окне будет представлены возможности входа в систему авторизованному пользователю, восстановления пароля, перехода к окну регистрации.

Макет окна восстановления пароля, на которое можно перейти путем нажатия кнопки переключения на окно восстановления пароля, представлен на рисунке 2.3.

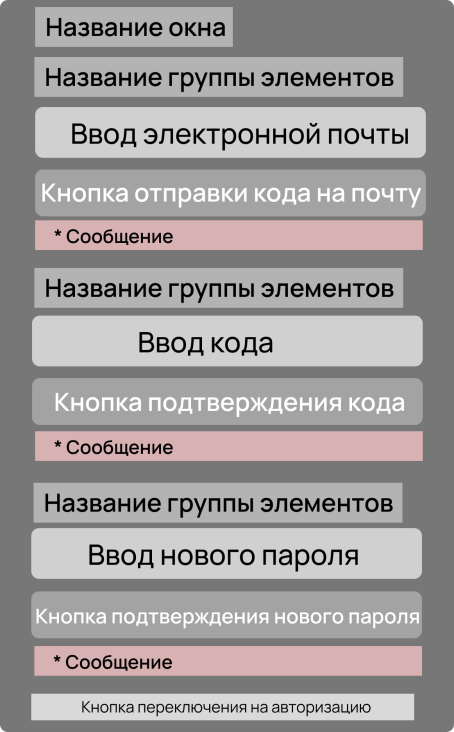


Рисунок 2.3

В окне восстановления пароля, макет которого представлен на рисунке 2.3, будет представлена возможность изменения пароля авторизованному пользователю путем отправки кода на электронную почту.

Макет окна регистрации участника олимпиады, на которое можно будет перейти путем нажатия кнопки переключения на регистрацию участника, представлен на рисунке 2.4.



Рисунок 2.4

После успешной авторизации в окне авторизации, макет которого представлен на рисунке 2.2, и нажатия кнопки входа будет доступно главное окно программы, макет которого представлен на рисунке 2.5.

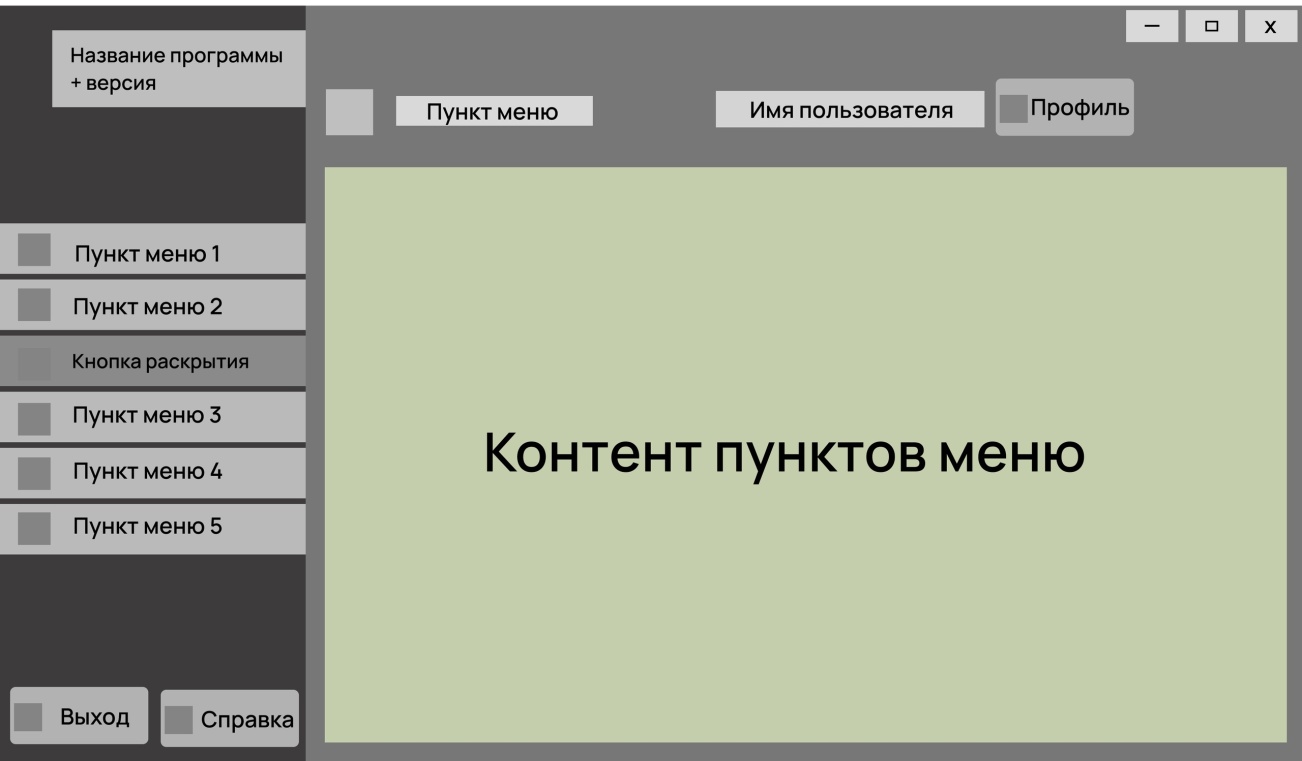


Рисунок 2.5

В главном окне программы участнику необходимо выбрать в меню раздел «Главная» для начала прохождения викторины. В результате будет отображено окно начала прохождения викторины, макет которого представлен на рисунке 2.6.

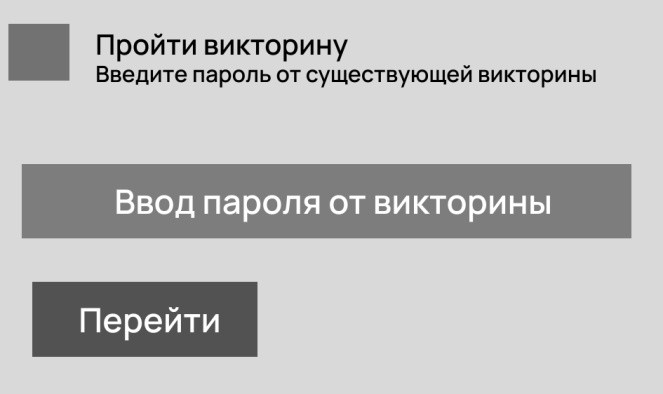


Рисунок 2.6

В окне начала прохождения викторины необходимо будет ввести пароль от викторины, а затем нажать на кнопку «Перейти». В результате будет отображено окно начала викторины, макет которого представлен на рисунке 2.7.

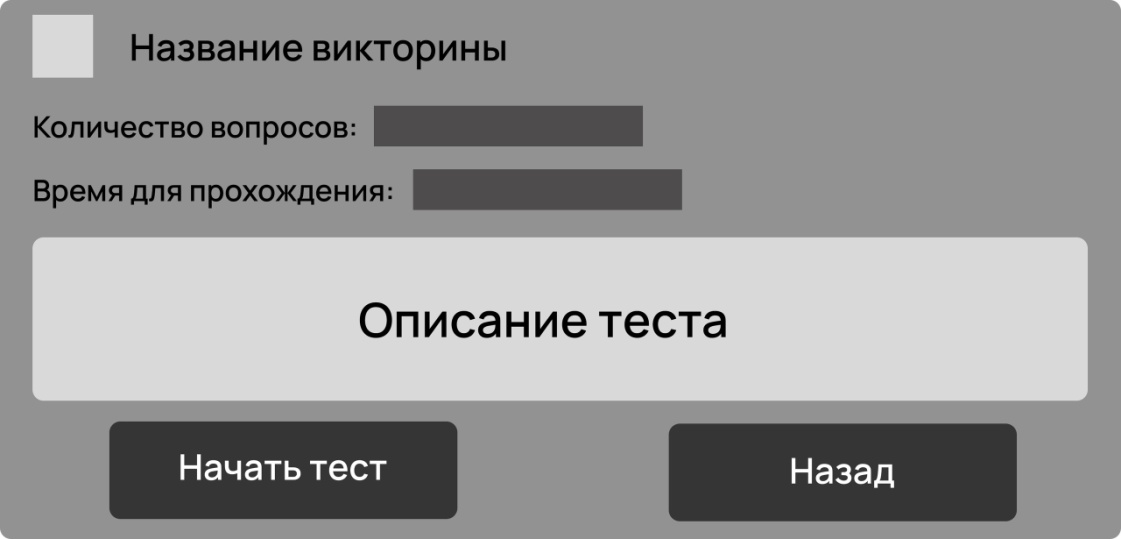


Рисунок 2.7

После нажатия кнопки «Начать тест» в окне начала викторины, макет которого представлен на рисунке 2.7, откроется окно прохождения теста, макет которого представлен на рисунке 2.8.



Рисунок 2.8

После прохождения всех вопросов и нажатия кнопки «Окончить викторину» в окне прохождения теста, макет которого представлен на рисунке 2.8, будет отображено окно заверщения теста с личным результатом и временем прохождения, макет которого представлен на рисунке 2.9.

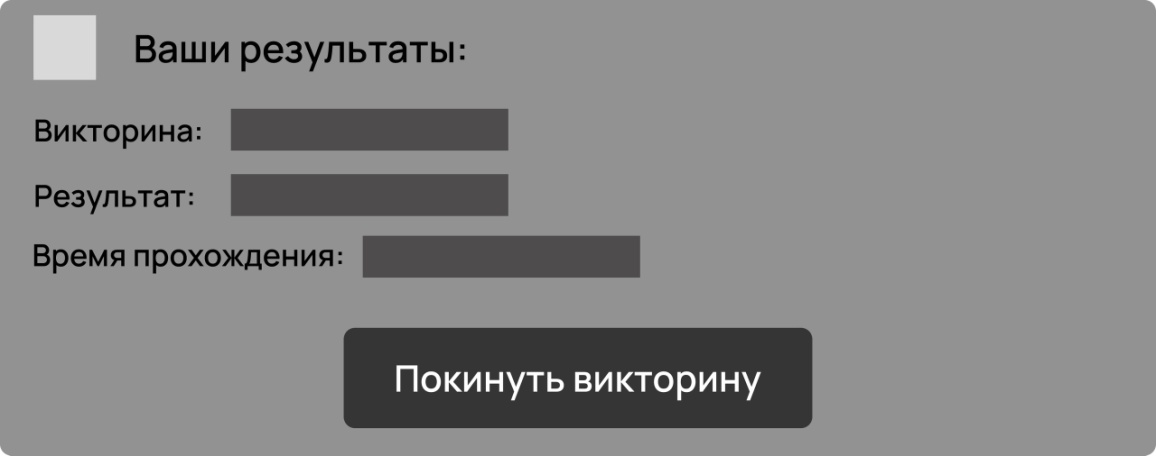


Рисунок 2.9

Для просмотра личных результатов тестирования необходимо будет в главном окне, макет которого представлен на рисунке 2.5, нажать пункт меню «Мои результаты». В результате будет отображено окно личных результатов, макет которого представлен на рисунке 2.10.

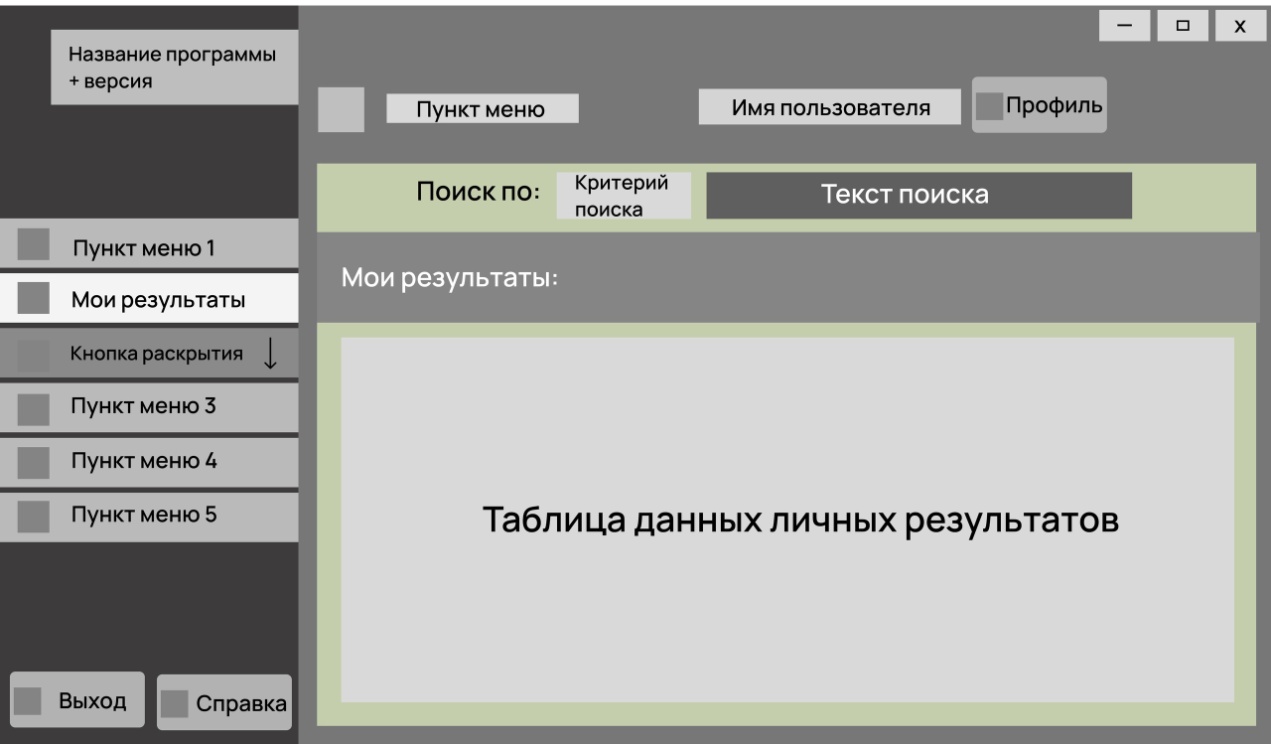


Рисунок 2.10

В программе на окнах будут использованы следующие основные компоненты:

* сетка «Grid» для сеткоподобного расположения элементов управления во всех окнах программного средства;
* стеллажная панель «StackPanel» ­ для горизонтального расположения элементов управлении и иконок отображения, будет использоваться во всех окнах программного средства;
* элемент «Button» ­ для открытия окон, авторизации, регистрации, смены пароля, для прохождения викторин, отправки ответов на вопросы, перехода между окнами и так далее;
* элемент «TextBox» ­ для ввода информации пользователя, информации о викторине, ответов на вопросы и так далее;
* элемент «TextBlock» ­ для отображения заголовков всех окон, заголовков и подсказок элементов ввода, сообщений об ошибках и так далее;
* элемент «ComboBox» – для выбора из списка предопределенных критериев отображения участников, групп, викторин и используется на окнах «Викторины», «Группы», «Участники»;
* элемент управления содержимым «ContentControl» – для отображения окон в зависимости от выбранного раздела в навигационном меню, будет использоваться только в главном окне программного средства;
* элемент «Border» ­ для добавления границы вокруг элемента управления или группы элементов управления, будет использоваться во всех окнах программного средства;
* элемент «ScrollViewer» – для добавления прокрутки к элементам управления или для отображения длинных списков или больших объемов данных о викторинах, группах, участниках, категориях, событиях;
* элемент «CheckBox» ­ для выбора одного или нескольких вариантов из предопределенного списка или обозначения выбора какого-либо из элементов;
* элемент «RadioButton» ­ для выбора одного из разделов программы.

# Описание реализации программного средства

## Инструменты разработки и применяемые технологии

При разработке программного средства для проведения тестирования знаний участников олимпиады использовались следующие инструменты разработки и технологии:

* операционная система Windows 10;
* среда разработки Visual Studio 2022;
* язык программирования C#;
* система управления базами данных (СУБД) MySQL;
* фреймворк с открытым исходным кодом Entity Framework;
* паттерн проектирования Model-View-ViewModel (MVVM);
* системы построения клиентских приложений Windows Presentation Foundation (WPF);
* инструмент для создания диаграмм и схем Draw.io.

Windows 10 была выбрана из-за своей широкой распространенности и надежных инструментов для разработчиков. Она предоставляет стабильную и привычную среду для разработки программного обеспечения, гарантируя совместимость с различными аппаратными и программными компонентами. Благодаря популярности Windows 10, доступны ресурсы для устранения проблем и разработки [8].

Visual Studio 2022 – это интегрированная среда разработки, которая предлагает полный набор инструментов разработки для создания программных приложений с использованием языка C#. Данная технология была выбрана для использования удобного интерфейса, мощных возможностей отладки и легкой интеграцией с другими продуктами и сервисами Microsoft [9].

C# представляет собой мощный и гибкий язык программирования, который обладает высоким уровнем контроля и отличной производительностью. Благодаря широкому использованию и обширным библиотекам, он является надежным выбором для создания эффективных и стабильных программных приложений. Кроме того, использование C# согласуется с текущей системой программного обеспечения и обеспечивает совместимость с другими технологиями, используемыми в проекте [10].

Система построения клиентских приложений WPF используется при создании графического интерфейса пользователя (GUI) для приложений, написанных на C#. Была выбрана в связи с тем, что Visual Studio 2022 предоставляет для инструментов разработки WPF широкий спектр элементов графического интерфейса пользователя, которые можно легко настроить в соответствии с потребностями приложения [11].

Для того чтобы обеспечить хорошо структурированную и удобную в обслуживании кодовую базу данных был использован паттерн проектирования Model-View-ViewModel (MVVM). Данный паттерн идеально подходит для десктопных приложений с использованием WPF из-за своих удобных инструментов привязки данных. Паттерн MVVM отделяет пользовательский интерфейс (View) от данных и логики приложения (Model) путем введения промежуточного слоя, называемого ViewModel. Модель ViewModel действует как связующее звено между представлением и моделью, позволяя им общаться без непосредственного взаимодействия друг с другом. Такое разделение задач облегчает модификацию и обновление кода, не затрагивая другие части приложения [18].

Для хранения и управления данными, которые используются в программе, была использована система управления базами данных (СУБД) MySQL, которая имеет ряд преимуществ. Она бесплатна и открыта. MySQL обеспечивает высокую производительность, масштабируемость и поддерживает разнообразные функции, включая хранимые процедуры и репликацию данных, она является надежной и гибкой СУБД, широко применяемой в различных областях разработки [19].

Entity Framework – это фреймворк с открытым исходным кодом, который упрощает доступ к данным, позволяя разработчикам работать с реляционными данными, используя специфические для домена объекты. Он обеспечивает высокоуровневый уровень абстракции, который позволяет разработчикам работать с данными в виде объектов, а не писать сложные SQL-запросы напрямую.  Entity Framework предоставляет ряд возможностей, включая поддержку Language Integrated Query (LINQ) и ряд поставщиков баз данных. Он также поддерживает подходы разработки «сначала код», «сначала база данных» и «сначала модель», предоставляя разработчикам гибкость в выборе подхода, который лучше всего соответствует их потребностям. Данный фреймворк был выбран из-за его легкой интеграции с языком C# и СУБД MySQL [13].

Инструмент Draw.io был выбран для создания диаграмм UML и других визуальных представлений из-за его обширного функционала и простоты использования. Он предлагает множество шаблонов и поддерживает различные типы диаграмм, что позволяет наглядно и интуитивно представить сложные отношения и структуры [16].

## Порядок авторизации пользователей

Для обеспечения безопасного функционирования программного средства, которое работает с конфиденциальной информацией, необходимо уделить внимание аутентификации и авторизации. Роль в этом процессе играют назначение ролей пользователям и предоставление соответствующих привилегий, чтобы гарантировать доступ только к необходимой информации, связанной с их ролями.

В программном средстве предусмотрены две основные роли – «Организатор» и «Участник». Важно определить характеристики каждой роли и предоставить им соответствующие привилегии.

«Организатор» имеет возможность создавать и настраивать викторины с вопросами, создавать категории и распределять викторины по категориям, формировать группы и добавлять участников, а также назначать викторины для участников (и групп участников) и просматривать результаты. Организатор также может экспортировать необходимую информацию.

С другой стороны, «Участник» ограничен выполнением назначенных тестов и просмотром своих результатов без возможности внесения изменений в викторины или получения доступа к другим функциям, предназначенных для организаторов.

Для роли «Организатор» в качестве логина будет использоваться электронная почта «organizer@mail.ru» с паролем «123456». Для роли «Участник» в качестве логина будет использоваться электронная почта «student@mail.ru» с паролем «123456».

В целом, предложенная процедура аутентификации и авторизации гарантирует, что только авторизованные пользователи могут получить доступ к тем функциям программного средства, которые для них важны. Ограничивая доступ к определенным функциям в зависимости от роли пользователя, будет обеспечена безопасность программного средства и защита данных пользователя.

## 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 | 255 | 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 |
| 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 |

Продолжение таблицы 3.8

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| 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 |
| EventId | varchar | 255 | Уникальный идентификатор события из таблицы «events» |
| 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 группы |

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 группы из таблицы «aspnetroles» |

Физическая схема базы данных представлена на рисунке 3.1.

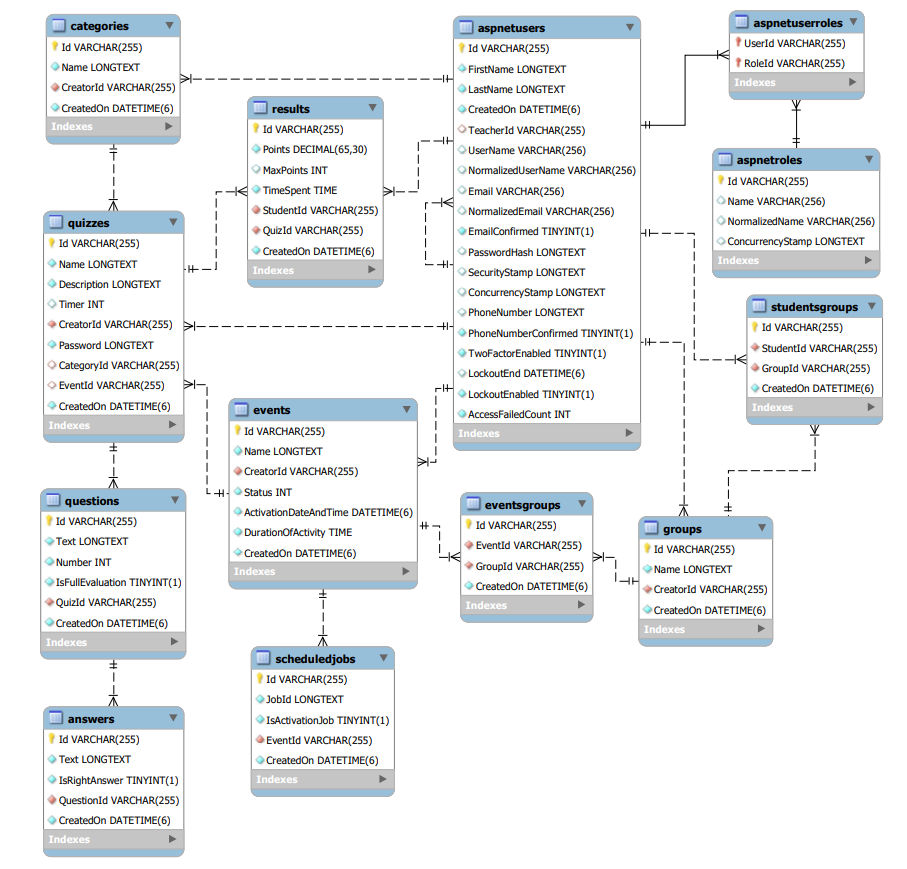


Рисунок 3.1

## Функции: логическая и физическая организация

В процессе разработки программного средства для тестирования знаний участников олимпиады было создано множество программных функций и процедур, каждая из которых выполняет определенные действия в определенный момент времени.

Для регистрации нового пользователя в программе в качестве «Участника» требуется нажать компонент button под названием «Регистрация участника» в окне авторизации. После этого появится окно регистрации участника, где необходимо заполнить соответствующие поля данными и нажать компонент button под названием «Регистрация». Затем 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;

}

Функция «LoginAsync» выполняет проверку данных, введенных пользователем при регистрации (проверка наличия пользователя с указанным адресом электронной почты и проверка пароля), а также обновляет необходимую информацию, связанную с учетной записью. Эта функция возвращает логическое значение, которое указывает на успешность или неудачу процесса входа. Код функции «LoginAsync» 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)

{

CurrentUser.IsLoggedIn = true;

CurrentUser.CurrentUser = user;

AccountStore.CurrentAdminId = user.Id;

return true;

}

return false;

}

Для перехода к окну начала прохождения викторины необходимо в главном окне ввести в компонент textBlock пароль существующей викторины и нажать на компонент button под названием «Перейти». В результате чего будет вызвана функция «GetQuizByPasswordAsync», код которой представлен ниже

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

{

return await quizRepository

.AllAsNoTracking()

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

.To<T>()

.FirstOrDefaultAsync();

}

Для вычисления результатов прохождения викторины разработана функция «CalculateResult», 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);

}

Для сохранения результата тестирования используется функция «CreateResultAsync», которая вызывается после окончании прохождения викторины при нажатии в окне окончания викторины компонента button под названием «Покинуть викторину». Данная функция сохраняет время прохождения, набранные баллы и иную информацию по результатам прохождения тестирования. Код функции «CreateResultAsync» представлен ниже

public async Task<string> CreateResultAsync(string studentId, int points, int maxPoints, string quizId)

{

var @event = await eventRepository

.All()

.Include(e => e.Quizzes)

.FirstOrDefaultAsync(e => e.Quizzes.Any(q => q.Id == quizId));

var quiz = @event.Quizzes.FirstOrDefault(q => q.Id == quizId);

var result = new Result()

{

Points = points,

StudentId = studentId,

MaxPoints = maxPoints,

EventId = @event.Id,

EventName = @event.Name,

QuizName = quiz.Name,

EventActivationDateAndTime = @event.ActivationDateAndTime,

};

await repository.AddAsync(result);

await repository.SaveChangesAsync();

@event.Results.Add(result);

await eventRepository.SaveChangesAsync();

return result.Id;

}

Полный код программы представлен в приложении А.

## Входные и выходные данные

В данном программном средстве входными данными являются данные об участниках и викторинах (вопросы и ответы). Кроме того, входными являются данные, вводимые в поля для поиска данных.

Входными данными при регистрации организатора или учащегося являются:

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

Входными данными при поиске по личным результатам являются:

* название викторины;
* название события.

Входными данными при поиске по событиям является название события.

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 прохождении викторины являются:

* пароль от викторины;
* ответ на вопрос викторины.

Выходными данными являются:

* отчет, показывающий личные результаты участника, который представлен в приложении Б на рисунке Б.1;
* отчет о событиях, назначенных участнику, который представлен в приложении Б на рисунке Б.2.

## Функциональное 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 для участника олимпиады представлено в приложении Б на рисунке Б.7.

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.3).

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.4).

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 невалидными 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 «LoginAsync» класса «UserAccountService.cs» для авторизации пользователя в роли «Участник», когда введенные данные валидные (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.5 и рисунке Б.7).

Таблица 3.16

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка авторизации валидного пользователя в роли «Участник» с валидными данными |
| Название теста | LoginAsync\_ValidCredentials\_ReturnsTrue |
| Входные данные | Экземпляр пользователя с ролью «Участник» с валидными данными |
| Действие | Вызов метода «LoginAsync» с входными данными |
| Ожидаемый результат | true |
| Фактический результат | true |
| Статус | Passed (Пройден) |

Код теста «LoginAsync\_ValidCredentials\_ReturnsTrue» представлен ниже

[Fact]

public async Task LoginAsync\_ValidCredentials\_ReturnsTrue()

{

// Arrange

var email = "organizer@mail.ru";

var password = "123456";

var user = new ApplicationUser { Email = email };

mockUserManager.Setup(m => m.FindByEmailAsync(email))

.ReturnsAsync(user);

mockUserManager.Setup(m => m.CheckPasswordAsync(user, password))

.ReturnsAsync(true);

mockUserManager.Setup(m => m.GetRolesAsync(user))

.ReturnsAsync(new[] { "Student" });

// Act

var result = await userAccountService.LoginAsync(email, password);

// Assert

Assert.True(result);

}

В таблице 3.17 представлено описание теста метода «LoginAsync» класса «UserAccountService.cs» для авторизации пользователя в роли «Участник», когда введенные данные невалидные (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.6).

Таблица 3.17

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка авторизации валидного пользователя в роли «Участник» с невалидными данными |
| Название теста | LoginAsync\_InvalidCredentials\_ReturnsFalse |
| Входные данные | Экземпляр пользователя с ролью «Участник» с невалидными данными |
| Действие | Вызов метода «LoginAsync» с входными данными |
| Ожидаемый результат | true |
| Фактический результат | true |
| Статус | Passed (Пройден) |

Код теста «LoginAsync\_InvalidCredentials\_ReturnsFalse» представлен ниже

[Fact]

public async Task LoginAsync\_InvalidCredentials\_ReturnsFalse()

{

// Arrange

var email = "organizer@mail.ru";

var password = "1";

mockUserManager.Setup(x => x.FindByEmailAsync(email))

.ReturnsAsync((ApplicationUser)null);

// Act

var result = await userAccountService.LoginAsync(email, password);

// Assert

Assert.False(result); }

В таблице 3.18 представлено описание теста метода «CreateQuestionAsync» класса «QuestionsService.cs» для создания нового вопроса в викторине с неполным совпадением ответа (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.8).

Таблица 3.18

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка создания нового вопроса с указанными данными |
| Название теста | CreateQuestionAsync\_ShouldCreateNewQuestionInDb |
| Входные данные | * идентификатор викторины: случайно сгенерированный уникальный идентификатор; * проверка на 100 % совпадение ответа: false; * описание вопроса: «Текст первого вопроса». |
| Действие | Вызов метода «CreateQuestionAsync» с входными данными |
| Ожидаемый результат | * количество вопросов в базе данных равно одному; * в коллекции «Questions» существует вопрос с идентификатором «newQuestionId». |
| Фактический результат | * количество вопросов в базе данных равно одному; * в коллекции «Questions» существует вопрос с идентификатором «newQuestionId». |
| Статус | Passed (Пройден) |

Код теста «CreateQuestionAsync\_ShouldCreateNewQuestionInDb» представлен ниже

[Fact]

public async Task CreateQuestionAsync\_ShouldCreateNewQuestionInDb()

{

//Arrange

var quizId = await CreateQuizAsync();

//Act

var newQuestionId = await Service.CreateQuestionAsync(quizId, false, "Текст первого вопроса");

var questionsCount = DbContext.Questions.ToArray().Count();

//Assert

Assert.Equal(1, questionsCount);

Assert.NotNull(await DbContext.Questions.FirstOrDefaultAsync(x => x.Id == newQuestionId));

}

В таблице 3.19 представлено описание теста метода «CreateQuestionAsync» класса «QuestionsService.cs» для создания нового вопроса в викторине с полным совпадением ответа (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.9).

Таблица 3.19

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка создания нового вопроса с указанными данными |
| Название теста | CreateQuestionAsync\_ShouldCreateNewQuestionInDb |
| Входные данные | * идентификатор викторины: случайно сгенерированный уникальный идентификатор; * проверка на 100% совпадение ответа: true; * описание вопроса: «Текст второго вопроса». |
| Действие | Вызов метода «CreateQuestionAsync» с входными данными |
| Ожидаемый результат | * количество вопросов в базе данных равно двум; * в коллекции «Questions» существует вопрос с идентификатором «newQuestionId». |
| Фактический результат | * количество вопросов в базе данных равно двум; * в коллекции «Questions» существует вопрос с идентификатором «newQuestionId». |
| Статус | Passed (Пройден) |

Код теста «CreateQuestionAsync\_ShouldCreateNewQuestionInDb» представлен ниже

[Fact]

public async Task CreateQuestionAsync\_ShouldCreateNewQuestionInDb()

{

//Arrange

var quizId = await CreateQuizAsync();

//Act

var newQuestionId = await Service.CreateQuestionAsync(quizId, true, "Текст второго вопроса");

var questionsCount = DbContext.Questions.ToArray().Count();

//Assert

Assert.Equal(2, questionsCount);

Assert.NotNull(await DbContext.Questions.FirstOrDefaultAsync(x => x.Id == newQuestionId));

}

В таблице 3.20 представлено описание теста метода «CreateAnswerAsync» класса «AnswersService.cs» для создания неправильного ответа к вопросу (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.10).

Таблица 3.20

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка создания нового неправильного ответа с указанными данными |
| Название теста | CreateAnswerAsync\_ShouldCreateNewAnswer |
| Входные данные | * идентификатор вопроса: случайно сгенерированный уникальный идентификатор; * индикатор правильности вопроса: false; * описание ответа: «Неверный ответ». |
| Действие | Вызов метода «CreateAnswerAsync» с входными данными |
| Ожидаемый результат | * количество ответов в базе данных равно одному; * свойство «Text» ответа равно «Неверный ответ»; * свойство «QuestionId» ответа совпадает с questionId; * свойство «IsRightAnswer» ответа равно false. |
| Фактический результат | * количество ответов в базе данных равно одному; * свойство «Text» ответа равно «Неверный ответ»; * свойство «QuestionId» ответа совпадает с questionId; * свойство «IsRightAnswer» ответа равно false. |
| Статус | Passed (Пройден) |

Код теста «CreateAnswerAsync\_ShouldCreateNewAnswer» представлен ниже

[Fact]

public async Task CreateAnswerAsync\_ShouldCreateNewAnswer()

{

//Arrange

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

//Act

await Service.CreateAnswerAsync("Неверный ответ", false, questionId);

//Assert

var answer = await DbContext.Answers.FirstOrDefaultAsync();

var answersCount = DbContext.Answers.ToArray().Count();

Assert.Equal(1, answersCount);

Assert.Equal("Неверный ответ", answer.Text);

Assert.Equal(questionId, answer.QuestionId);

Assert.False(answer.IsRightAnswer);

}

В таблице 3.21 представлено описание теста метода «CreateAnswerAsync» класса «AnswersService.cs» для создания правильного ответа к вопросу (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.11).

Таблица 3.21

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка создания нового правильного ответа с указанными данными |
| Название теста | CreateAnswerAsync\_ShouldCreateNewAnswer |
| Входные данные | * идентификатор вопроса: случайно сгенерированный уникальный идентификатор; * индикатор правильности вопроса: true; * описание ответа: «Верный ответ». |
| Действие | Вызов метода «CreateAnswerAsync» с входными данными |
| Ожидаемый результат | * количество ответов в базе данных равно одному; * свойство «Text» ответа равно «Верный ответ»; * свойство «QuestionId» ответа совпадает с questionId; * свойство «IsRightAnswer» ответа равно true. |
| Фактический результат | * количество ответов в базе данных равно одному; * свойство «Text» ответа равно «Верный ответ»; * свойство «QuestionId» ответа совпадает с questionId; * свойство «IsRightAnswer» ответа равно true. |
| Статус | Passed (Пройден) |

Код теста «CreateAnswerAsync\_ShouldCreateNewAnswer» представлен ниже

[Fact]

public async Task CreateAnswerAsync\_ShouldCreateNewAnswer()

{

//Arrange

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

//Act

await Service.CreateAnswerAsync("Верный ответ", true, questionId);

//Assert

var answer = await DbContext.Answers.FirstOrDefaultAsync();

var answersCount = DbContext.Answers.ToArray().Count();

Assert.Equal(1, answersCount);

Assert.Equal("Верный ответ", answer.Text);

Assert.Equal(questionId, answer.QuestionId);

Assert.True(answer.IsRightAnswer);

}

В таблице 3.22 представлено описание теста метода «AddToQuizQuestionAsync» класса «QuestionsService.cs» для добавления вопроса к викторине (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке  Б.12).

Таблица 3.22

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка добавления нового вопроса с указанными данными к викторине |
| Название теста | CreateAndAddQuestionToQuiz |
| Входные данные | * идентификатор викторины: случайно сгенерированный уникальный идентификатор; * номер вопроса в викторине: единица; * описание вопроса: «Текст первого вопроса». |
| Действие | Вызов метода «AddQuestionToQuizAsync» с входными данными |
| Ожидаемый результат | * количество ответов в викторине равно одному; * свойство «Text» вопроса равно «Текст первого вопроса»; * свойство «quizId» ответа совпадает с quizId. |
| Фактический результат | * количество ответов в викторине равно одному; * свойство «Text» вопроса равно «Текст первого вопроса»; * свойство «quizId» ответа совпадает с quizId. |
| Статус | Passed (Пройден) |

Код теста «CreateAndAddQuestionToQuiz» представлен ниже

private async Task<string> CreateAndAddQuestionToQuiz(string quizId, int questionNumber, string text)

{

var question = new Question

{

Number = questionNumber,

Text = text,

QuizId = quizId,

};

var quiz = await DbContext.Quizzes.FirstOrDefaultAsync(x => x.Id == quizId);

quiz.Questions.Add(question);

DbContext.Update(quiz);

await DbContext.SaveChangesAsync();

DbContext.Entry(question).State = EntityState.Detached;

return question.Id;

}

В таблице 3.23 представлено описание теста метода «CreateResultAsync» класса «ResultsService.cs» для прохождения тестирования (визуальный интерфейс программы для работы с указанным классом и его методами представлен в приложении Б на рисунке Б.13 и рисунке Б.14).

Таблица 3.23

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка прохождения тестирования |
| Название теста | CreateResultAsync\_ShouldCreateNewResult |
| Входные данные | * идентификатор учащегося: случайно сгенерированный уникальный идентификатор * идентификатор викторины: случайно сгенерированный уникальный идентификатор |
| Действие | Вызов метода «CreateResultAsync» с входными данными |
| Ожидаемый результат | * количество результатов в базе данных равно 1; * результат с уникальным идентификатором newResultId существует в коллекции «Results». |
| Фактический результат | * количество результатов в базе данных равно 1; * результат с уникальным идентификатором newResultId существует в коллекции «Results». |
| Статус | Passed(Пройден) |

Код теста «CreateResultAsync\_ShouldCreateNewResult» представлен ниже

[Fact]

public async Task CreateResultAsync\_ShouldCreateNewResult()

{

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

var eventInfo = await CreateEventAsync("event", DateTime.UtcNow);

var quizId = eventInfo[1];

var newResultId = await Service.CreateResultAsync(studentId, quizId);

var resultsCount = DbContext.Results.ToArray().Count();

Assert.Equal(1, resultsCount);

Assert.NotNull(await DbContext.Results.FirstOrDefaultAsync(x => x.Id == newResultId));

}

В таблице 3.24 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.15).

Таблица 3.24

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| Описание теста | Проверка вычисления результата прохождения тестирования, когда все ответы верны и включена полная оценка |

Продолжение таблицы 3.24

|  |  |
| --- | --- |
| Характеристика теста | Описание характеристики |
| 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]

publicvoid CalculateResult\_FullEvaluation\_AllAnswersCorrect\_Returns1()

{

// Arrange

varoriginalQuizQuestions = new List<QuestionViewModel>

{

newQuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

newAnswerViewModel{ Id = "1", IsRightAnswer = true },

newAnswerViewModel{ Id = "2", IsRightAnswer = true }

}

}

};

varattemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

newAttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = true,

Answers = new List<AttemptedQuizAnswerViewModel>

{

newAttemptedQuizAnswerViewModel{ Id = "1", IsRightAnswerAssumption = true },

newAttemptedQuizAnswerViewModel{ Id = "2", IsRightAnswerAssumption = true }

}

}

};

// Act

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

// Assert

Assert.Equal(1, result);

}

В таблице 3.25 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 (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.15).

Таблица 3.25

|  |  |
| --- | --- |
| 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\_AllAnswersCorrect\_Returns0» представлен ниже

[Fact]

publicvoid CalculateResult\_FullEvaluation\_NotAllAnswersCorrect\_Returns0()

{

// Arrange

varoriginalQuizQuestions = new List<QuestionViewModel>

{

newQuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

newAnswerViewModel{ Id = "1", IsRightAnswer = true },

newAnswerViewModel{ Id = "2", IsRightAnswer = true }

}

}

};

varattemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

newAttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = true,

Answers = new List<AttemptedQuizAnswerViewModel>

{

newAttemptedQuizAnswerViewModel{ Id = "1", IsRightAnswerAssumption = true },

newAttemptedQuizAnswerViewModel{ Id = "2", IsRightAnswerAssumption = false }

}

}

};

// Act

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

// Assert

Assert.Equal(0, result);

}

В таблице 3.26 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.15).

Таблица 3.26

|  |  |
| --- | --- |
| 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 |
| Статус | Passed (Пройден) |

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

[Fact]

publicvoid CalculateResult\_PartialEvaluation\_AllAnswersCorrect\_Returns1()

{

// Arrange

varoriginalQuizQuestions = new List<QuestionViewModel>

{

newQuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

newAnswerViewModel{ Id = "1", IsRightAnswer = true },

newAnswerViewModel{ Id = "2", IsRightAnswer = true }

}

}

};

varattemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

newAttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = false,

Answers = new List<AttemptedQuizAnswerViewModel>

{

newAttemptedQuizAnswerViewModel{ Id = "1", IsRightAnswerAssumption = true },

newAttemptedQuizAnswerViewModel{ Id = "2", IsRightAnswerAssumption = true }

}

}

};

// Act

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

// Assert

Assert.Equal(1, result);

}

В таблице 3.27 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.15).

Таблица 3.27

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

publicvoid CalculateResult\_PartialEvaluation\_SomeAnswersCorrect\_ReturnsPartialPoints()

{

// Arrange

varoriginalQuizQuestions = new List<QuestionViewModel>

{

newQuestionViewModel

{

Id = "1",

Answers = new List<AnswerViewModel>

{

newAnswerViewModel{ Id = "1", IsRightAnswer = true },

newAnswerViewModel{ Id = "2", IsRightAnswer = true }

}

}

};

varattemptedQuizQuestions = new List<AttemptedQuizQuestionViewModel>

{

newAttemptedQuizQuestionViewModel

{

Id = "1",

IsFullEvaluation = false,

Answers = new List<AttemptedQuizAnswerViewModel>

{

newAttemptedQuizAnswerViewModel{ Id = "1", IsRightAnswerAssumption = true },

newAttemptedQuizAnswerViewModel{ 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.

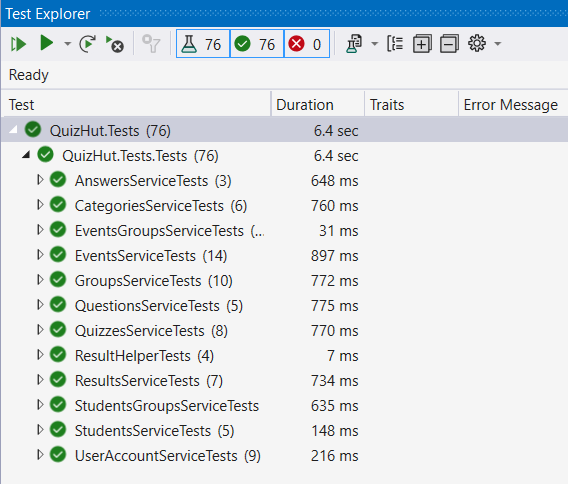


Рисунок 3.2

## 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 [15].

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.

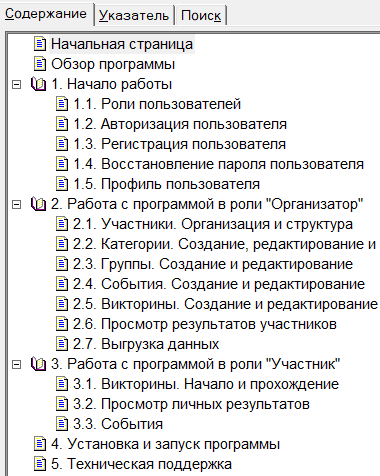


Рисунок 3.3

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

# Применение

## Назначение программного средства

Программа «QuizHut.exe» разработана специально для частного учреждения образования «Колледж бизнеса и права» с целью автоматизации следующих функций:

* регистрация информации об участниках олимпиады;
* создание и редактирование тестовых заданий и их публикация по таймеру;
* получение решений от участников в заданные сроки;
* задание критериев оценки работ участников олимпиады;
* выгрузка данных об итогах проведения олимпиады;
* получение результатов по разным событиям;
* создание и редактирование структурных единиц программы: категорий, групп;
* просмотр и изменение данных пользователя.

Разработанное программное средство автоматизирует комплекс процессов проведения олимпиад и делает возможным проведение заочных туров. Использование программы «QuizHut.exe» позволяет многократно снизить затрат на проведение олимпиады и исключить необходимость выполнения однотипной работы. После завершения олимпиады сформированные документы могут быть использованы для включения в отчет для контролирующих органов.

## 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», чтобы программа могла отправлять электронные письма;
* запустить «setup.exe» файл и следовать указаниям установщика.

# Охрана окружающей среды

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

Компания ООО «Модсен» имеет системный менеджмент управления охраной труда (СУОТ), разработанный в соответствии с требованиями нормативных правовых актов Республики Беларусь и СТБ ISO 45001-2020 «Системы менеджмента здоровья и безопасности труда при профессиональной деятельности. Требования и руководство по применению». Эта система направлена на обеспечение безопасных условий работы и соблюдение законодательства в области охраны труда, способствуя сохранению жизни и здоровья сотрудников [6].

Для организации и контроля по вопросам охраны труда в компании создана служба охраны труда под руководством инженера по охране труда.

Главные задачи службы охраны труда включают:

* координацию деятельности отделов и подразделений компании для обеспечения безопасных условий труда;
* совершенствование СУОТ;
* внедрение передового опыта и научных разработок в области безопасности и охраны труда, а также распространение знаний охраны труда;
* информирование и консультирование сотрудников компании, включая руководство, по вопросам охраны труда;
* контроль за соблюдением требований законодательства, технических нормативных актов и локальных правовых актов, связанных с охраной труда.

Главный инженер компании отвечает за проведение мероприятий по устранению выявленных недостатков и нарушений в области охраны труда, выявленных в результате ежемесячного контроля. Если нарушения не могут быть устранены отделом или подразделением, они сообщаются генеральному директору компании.

Генеральный директор компании ООО «Модсен» осуществляет организацию, оперативное руководство и координацию работы по охране труда, а также общее руководство управлением охраной труда. Главный инженер и заместитель генерального директора являются ответственными за реализацию функций СУОТ.

Функции инженера по охране труда на ООО «Модсен» включают:

* анализ причин производственного травматизма, профессиональной заболеваемости в компании и разработку мер по их профилактике, контроль их выполнения и эффективности;
* участие в организации разработки и реализации программ улучшения условий и охраны труда на предприятии;
* согласование программ с заинтересованными структурными подразделениями и органами государственного надзора;
* организацию обучения, повышения квалификации и проверки знаний руководителей и специалистов компании;
* проведение инструктажей, лекций, тренингов, консультаций по актуальным темам охраны труда;
* подготовку статистической отчетности по охране труда и анализ показателей по охране труда в компании, выявление проблемных зон и направлений улучшения;
* пропаганду и распространение передового опыта в области охраны труда в компании;
* участие в расследовании несчастных случаев на производстве;
* информационное обеспечение работников компании по вопросам охраны труда;
* мониторинг нормативно-правовой базы по охране труда;
* консультации и подготовку ответов на запросы организаций и обращений граждан.

Работодатель в лице компании ООО «Модсен» обязан в соответствии с Законом Республики Беларусь от 23.06.2008 № 356-З «Об охране труда» и Трудовым кодексом Республики Беларусь осуществлять обучение, стажировку, инструктаж и проверку знаний работников по вопросам охраны труда [6].

Обучение работников компании по охране труда соответствует государственным стандартам, межотраслевым положениям, типовым инструкциям и программам.

Руководители и специалисты проходят обучение по учебным планам и программам на основе типового перечня вопросов. Руководитель предприятия определяет список должностей для стажировки и ее длительность в соответствии с законодательством. Руководители и специалисты проходят проверку знаний в комиссиях не реже одного раза в три года.

Ответственность за организацию обучения, инструктажа, стажировки и проверки знаний лежит на генеральном директоре и руководителях структурных подразделений. Инженер по охране труда проводит вводный инструктаж с недавно принятыми работниками, а главные инженеры цехов, начальники отделов и подразделений проводят первичный, повторный, целевой и внеплановый инструктажи.

Расследование несчастных случаев и профессиональных заболеваний на производстве осуществляется в соответствии с правилами расследования и учета несчастных случаев на производстве и профессиональных заболеваний, утвержденными Постановлением Совета Министров Республики Беларусь от 10.01.2004 года №30.

Расследование несчастных случаев и профессиональных заболеваний на производстве имеет следующие цели и меры:

* определение причин несчастного случая и принятие экстренных мер для устранения их последствий;
* выявление лиц, ответственных за нарушения, приведшие к несчастному случаю, и определение нарушенных норм и правил;
* привлечение виновных лиц к ответственности в соответствии с законодательством и нормативными актами.

Для расследования и анализа инцидентов на предприятии формируется комиссия, включающая представителей администрации, инженера по охране труда и профсоюзного комитета. Комиссия анализирует происшествия, выявляет причины и разрабатывает рекомендации для предотвращения подобных случаев в будущем.

Для предотвращения производственных травм и профессиональных заболеваний на предприятии проводятся следующие мероприятия:

* обучение и инструктаж персонала по правилам охраны труда, технике безопасности и пожарной безопасности;
* контроль соблюдения правил охраны труда и техники безопасности на рабочих местах;
* аудиты и инспекции рабочих мест для выявления и устранения нарушений правил охраны труда и техники безопасности;
* профилактические мероприятия по предотвращению профессиональных заболеваний, включая регулярные медицинские осмотры и обучение здоровому образу жизни;
* обеспечение регламентированных перерывов в работе для отдыха и восстановления работоспособности персонала.

Эффективная реализация мер по охране труда и технике безопасности в компании снижает риск несчастных случаев и профессиональных заболеваний, обеспечивая безопасные условия труда для всех работников предприятия [6].

Основные виды контроля за соблюдением правил охраны труда в компании ООО «Модсен» включают:

* контроль соблюдения законодательства об охране труда, который осуществляют руководители и специалисты компании в соответствии с их должностными обязанностями;
* контроль охраны труда, который осуществляет служба охраны труда предприятия в соответствии с типовым положением о службе охраны труда организации, утвержденным постановлением Министерства труда и социальной защиты Республики Беларусь от 24 мая 2002 года N82;
* периодический контроль за соблюдением законодательства об охране труда, который проводят представители компании с участием общественных инспекторов профсоюзов по охране труда.

Периодический контроль проводится в следующем порядке:

* ежедневно: контроль состояния охраны труда осуществляется руководителем структурного подразделения с участием общественного инспектора профсоюза по охране труда на рабочих участках и в производственных отделах;
* ежемесячно: контроль проводится главным инженером компании с участием общественного инспектора профсоюза по охране труда и ведущим инженером по охране труда на различных рабочих участках;
* ежеквартально: контроль осуществляется генеральным директором компании с участием руководителей структурных подразделений и общественных инспекторов по ОТ.

Эти меры контроля помогают обеспечить соблюдение требований охраны труда и техники безопасности в компании ООО «Модсен» и создать безопасные условия труда для всех работников.

## Разработка мер по нормализации и защите от источников ионизирующих излучений при эксплуатации ПЭВМ

В данном подразделе изучается нормализация и защита от ионизирующих излучений при эксплуатации ПЭВМ.

Ионизирующее излучение может возникать при радиоактивном распаде, ядерных превращениях или торможении заряженных частиц в веществе. Оно классифицируется на корпускулярное и фотонное излучение.

Корпускулярное излучение состоит из потока элементарных частиц, таких как а-частицы и р-частицы, нейтроны, протоны, дейтроны и тяжелые ионы.

Фотонное излучение, в свою очередь, включает в себя рентгеновские лучи, у-лучи и тормозное излучение. Ионизирующая способность излучения определяется удельной ионизацией, а проникающая способность – величиной пробега.

Фотонное излучение обладает наименьшей ионизирующей и наибольшей проникающей способностью. Взаимодействуя со средой, электромагнитное излучение преобразуется в кинетическую энергию вторичных электронов, вызывая ионизацию.

Ионизирующие излучения оказывают биологический эффект, вызывая сложные физические и биохимические процессы в тканях организма. Разрыв молекулярных связей и изменение химической структуры соединений приводят к нарушениям в работе клеток и могут быть обратимыми или необратимыми. Необратимые нарушения могут привести к поражению органов или развитию лучевой болезни [12].

Радиационная безопасность обеспечивается выполнением Закона Республики Беларусь «О радиационной безопасности населения», «Нормами радиационной безопасности» (НРБ-2000) и «Основными санитарными правилами обеспечения радиационной безопасности» (ОСП-2002) [3].

Основными принципами обеспечения радиационной безопасности при практической деятельности в условиях нормальной эксплуатации ПЭВМ и других источников излучения являются:

* непревышение допустимых пределов индивидуальных доз облучения человека от всех источников излучений;
* запрещение всех видов деятельности по использованию источников излучения, при которых полученная для человека и общества польза не превышает риск возможного вреда, причиненного дополнительным облучением, превышающим естественный радиационный;
* поддержание индивидуальных доз облучения на возможно низком и достижимом уровне с учетом экономических и социальных факторов и количества облучаемых лиц при использовании любого источника излучения.

НРБ-2000 устанавливают две категории облучаемых лиц:

* персонал;
* все население, включая и персонал, который находится за пределами их производственной деятельности.

Для категорий облучаемых лиц НРБ-2000 установлены три класса нормативов:

* основные пределы доз (ПД);
* допустимые уровни монофакторного воздействия и пути его поступления;
* контрольные уровни (дозы, уровни активности, плотности потоков и другое).

Дозовые пределы облучения являются основной нормируемой величиной и измеряются в миллизивертах (мЗв).

Эффективная доза для персонала не должна превышать за весь период трудовой деятельности (50 лет) – 1000 мЗв, а для населения за период жизни (70 лет) – 70 мЗв.

Допустимые уровни являются производными дозовых пределов.

Количественными значениями допустимых уровней являются:

* предел годового поступления (ПГП);
* допустимая среднегодовая объемная активность (ДОА) и допустимая удельная активность (ДУА).

Предел годового поступления – допустимый уровень поступления данного радионуклида в организм в течение года, который при монофакторном воздействии приводит к облучению условного человека ожидаемой дозой, равной соответствующему пределу дозы [12].

При одновременном воздействии на человека источников внешнего и внутреннего излучения годовая эффективная доза не должна превышать установленных пределов доз.

Основные пределы доз излучения представлены в таблице 5.1.

Таблица 5.1

|  |  |  |
| --- | --- | --- |
| Нормируемая величина | Пределы доз для персонала (мЗв) | Пределы доз для населения (мЗв) |
| Эффективная доза | 20 мЗв в год в среднем за любые последовательные 5 лет, но не более 50 мЗв в год | 1 мЗв в год в среднем за любые последовательные 5 лет, но не более 5 мЗв в год |
| Эквивалентная доза за год в хрусталике глаза | 150 | 15 |
| Эквивалентная доза за год в коже | 500 | 50 |
| Эквивалентная доза за год в кистях и стопах. | 500 | 50 |

Для оценки состояния радиационной безопасности используется показатель радиационного риска, который характеризуется суммарной накопленной эффективной дозой от всех источников излучения.

Радиационная безопасность на объекте и вокруг него обеспечивается за счет:

* качества проектирования;
* обоснованного выбора района и площадки для размещения;
* физической защиты источников излучения;
* зонирования территории объекта; условий эксплуатации технологических систем;
* разрешений (лицензии) уполномоченных государственных органов на практическую деятельность объекта;
* государственной санитарно-гигиенической экспертизы изделий и технологий по радиационному фактору;
* наличия системы радиационного контроля;
* планирования и проведения мероприятий по обеспечению радиационной безопасности персонала и населения;
* радиационно-гигиенической грамотности персонала и населения.

Радиационная безопасность персонала в соответствии с ОСП-2002 обеспечивается:

* ограничением допуска к работе с источниками излучения по возрасту, полу, состоянию здоровья, уровню предыдущего облучения;
* знанием и соблюдением правил работы с источниками излучения;
* достаточностью коллективных средств защиты, экранов и расстояния от источников излучения, а также ограничением времени работы с источниками излучения [12];
* созданием условий труда, отвечающих требованиям НРБ-2000 и ОСП-2002;
* применением индивидуальных средств защиты;
* соблюдением установленных контрольных уровней;
* организацией радиационного контроля;
* информированием о радиационной обстановке;
* проведением эффективных мероприятий по защите персонала в случае угрозы и возникновении аварии.

Проектирование защиты от внешнего ионизирующего излучения должно выполняться с учетом назначения помещений, категории облучаемых лиц и длительности облучения.

При расчете защиты с коэффициентом запаса, равным двум, проектная мощность эквивалентной дозы излучения (Н) на поверхности защиты определяется по формуле (5.1).

(5.1)

где Д – предел дозы для персонала или населения, мЗв/год;

t – продолжительность облучения, часов в год.

К работе с источниками излучения допускаются лица не моложе 18 лет, не имеющие медицинских противопоказаний. Перед допуском к работе персонал должен пройти обучение, инструктаж по охране труда и проверку знаний правил безопасности ведения работ и соответствующих инструкций.

Проверка знаний проводится комиссией до начала работ и периодически, не реже одного раза в год, руководящего состава – не реже одного раза в три года.

Защита от внешнего облучения, так же как и электромагнитного, достигается временем, расстоянием и экранированием.

Продолжительность пребывания работника в опасной зоне должна ограничиваться временем, в течение которого он получает дозу, не превышающую допустимую.

Безопасность работ с ионизирующими излучениями обеспечивается также защитными экранами, толщина которых рассчитывается на основе законов ослабления излучений веществом экрана. Стационарными защитными экранами являются стены, перекрытия пола и потолка, двери, смотровые окна и тому подобное. К передвижным защитным устройствам относятся ширмы и экраны, изготовленные из специальных материалов, тубусы и диафрагмы, ограничивающие поток ионизирующего излучения, контейнеры для транспортировки и хранения источников излучения и так далее [12].

Защита работающих от внутреннего облучения заключается в исключении контакта человека с радиоактивными веществами в открытом виде, попадания их внутрь организма через воздух рабочей зоны, зараженную воду, пищу и тому подобное, предотвращении загрязнения радиоактивными веществами рук, одежды, поверхностей оборудования и помещения.

В заключение следует отметить, что защита сотрудников и окружающей среды от вредного воздействия ионизирующего излучения является важнейшей задачей при эксплуатации ПЭВМ и других электронных устройств. Важно разработать соответствующие меры, чтобы не превысить пределы дозы и обеспечить соблюдение стандартов безопасности. Этого можно достичь путем внедрения соответствующего экранирования, использования оборудования с более низким уровнем излучения и проведения регулярного обучения сотрудников, чтобы они знали о потенциальных рисках и способах защиты.

## Пожарная безопасность

По классификации правил устройства электроустановок (ПУЭ) помещения компании ООО «Модсен» относятся к категории B-Ia, так как при нормальной эксплуатации взрывоопасные смеси горючих газов или паров легковоспламеняющихся жидкостей с воздухом не образуются, а возможны только в результате аварий или неисправностей [3].

Класс пожароопасных зон П-I, так как в них обращаются горючие жидкости с температурой вспышки выше 61°С.

Здание ООО «Модсен» согласно паспорту и конструктивным характеристикам, по степени огнестойкости относится к категории IV:

* несущие элементы здания – R 60-К0 – предел огнестойкости 60 минут – по потере несущей способности; по пожарной опасности, не пожароопасные;
* самонесущие стены – RE 45-К0 – предел огнестойкости 45 минут – по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее; по пожарной опасности, не пожароопасные;
* наружные ненесущие стены – Е30-К1 – предел огнестойкости 30 минут – по потере теплоизолирующей способности; по пожарной опасности, небольшая пожароопасность;
* перекрытия междуэтажные – REI 45-К0 – предел огнестойкости 45 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое из трех предельных состояний наступит ранее; по пожарной опасности, не пожароопасные;
* настилы в том числе с утеплителями – RE 15-К1 – предел огнестойкости 15 минут – по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее; по пожарной опасности, мало пожароопасные;
* фермы, балки, прогоны – R 15-К1 – предел огнестойкости 15 минут – по потере несущей способности; по пожарной опасности, мало пожароопасные;
* внутренние стены – REI 90-К0 – предел огнестойкости 90 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое из трех предельных состояний наступит ранее; по пожарной опасности, не пожароопасные;
* марши и площадки лестниц – R 45-К0 – предел огнестойкости 45 минут – по потере несущей способности; по пожарной опасности, не пожароопасные.

Средством сигнализации является извещатель пожарный СО-3, который необходим для обнаружения возгорания, сопровождающегося появлением дыма, и выдачи громкого звукового и светового сигнала «Тревога», а также для работы в сети с другими автономными извещателями и внешними светозвуковыми оповещателями.

Первичными средствами пожаротушения являются углекислотные огнетушители ОУ-2, в каждом кабинете находится по одному углекислотному огнетушителю ОУ-2 размещенном в специальном шкафу, недалеко от входных дверей с внутренней стороны кабинета.

Во всех зданиях расположены системы водяного пожаротушения. Они представляют собой напорный трубопровод и оросители, однако последние не имеют теплового элемента. Сама по себе система не реагирует на пожар, поэтому «пусковым механизмом» служит система пожарной сигнализации, которая также расположена по всему зданию, в непосредственной близости от каждого оросителя [3].

В соответствии с пунктом шесть статьи 17 Закона Республики Беларусь от 15.06.1993 «О пожарной безопасности» и положением для проведения профилактических мероприятий по предупреждению и тушению пожаров в организациях создается пожарная дружина из числа рабочих и специалистов организации:

* с численностью работающих в здании, сооружении и помещениях организации от 25 до 100 человек – в количестве не менее пяти человек;
* с численностью работающих свыше 100 человек – в количестве не менее пяти человек дополнительно на каждые 100 работающих;
* в организации с численностью работающих в здании, сооружении и помещениях менее 25 человек пожарная дружина может не создаваться, а обязанности на случай возникновения пожара должны распределяться между работниками такой организации решением ее руководителя.

Пожарная дружина также может не создаваться при наличии в организации пожарной команды. В компании ООО «Модсен» нет пожарной команды, но есть пожарная дружина из числа работников и специалистов, основными задачами которой являются:

* контроль за соблюдением противопожарного режима;
* проведение разъяснительной работы среди рабочих, служащих, инженерно-технических работников по соблюдению противопожарного режима на рабочем месте 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 пожаротушения;

Для безопасной эвакуации клиентов и работников в случае чрезвычайных ситуаций разработан эффективный план эвакуации, копии которого, расположены на каждом этаже, возле входа.

## Охрана окружающей среды

В компании ООО «Модсен» за охрану окружающей среды отвечает лицо, на которого возложены функции специалиста по охране окружающей среды, которое осуществляет постоянный мониторинг деятельности всех служб в рамках соблюдения требований инструкции по охране окружающей среды. Ответственность предприятия, направленная прежде всего на рациональное использование природных ресурсов, способствует снижению негативного воздействия производства на окружающую среду и улучшению экологической обстановки в зоне влияния предприятия [6].

Основные направления разработки мероприятий по охране окружающей среды в IT-компании:

* охрана атмосферного воздуха, защита озонового слоя: создание автоматизированных систем контроля выбросов вредных веществ в атмосферу, приобретение необходимого оборудования, приборов и программного обеспечения; проведение регулярных осмотров и обслуживание систем вентиляции;
* рациональное использование и охрана водных ресурсов: внедрение передовых технологий для предотвращения загрязнения сточных вод и сокращения потребления водных ресурсов; строительство, реконструкция и модернизация сетей и сооружений для сбора, транспортировки, очистки и отведения сточных вод; приобретение оборудования, необходимого для этих целей;
* мероприятия по предотвращению и снижению негативного воздействия отходов: внедрение современных технологий использования и обезвреживания отходов; приобретение контейнеров для сбора отходов и вторичных материальных ресурсов;
* снижение шума и вибрации: применение шумоизолирующих материалов и конструкций в рабочих помещениях; регулярное техническое обслуживание оборудования для предотвращения повышенного уровня вибрации и шума; разработка и внедрение инженерных решений для снижения вибрации и шума;
* осуществление мониторинга окружающей среды: регулярные измерения уровней загрязнения атмосферного воздуха, почвы и водных ресурсов; анализ полученных данных и принятие мер для устранения источников загрязнения;
* сотрудничество с органами государственного контроля и общественными организациями: обеспечение информационной прозрачности деятельности компании в области охраны окружающей среды; активное взаимодействие с органами государственного контроля по экологической безопасности и соблюдению законодательства; участие в экологических проектах и инициативах, организация мероприятий по информированию и просвещению населения о проблемах охраны окружающей среды и методах их решения [6];
* внедрение энергосберегающих технологий: оптимизация использования энергоресурсов путем внедрения современных, энергоэффективных технологий; проведение регулярного аудита энергопотребления для выявления возможных мероприятий по снижению энергозатрат.

В заключение, компанией ООО «Модсен» осуществляется систематическая работа по охране окружающей среды. Она включает создание автоматизированных систем контроля выбросов вредных веществ, рациональное использование и охрану водных ресурсов, предотвращение и снижение негативного воздействия отходов, снижение шума и вибрации, мониторинг окружающей среды, сотрудничество с государственными органами и общественными организациями по охране окружающей среды, а также внедрение энергосберегающих технологий. Компания стремится создать безопасные и экологически устойчивые условия работы, осознавая важность сохранения окружающей среды.

# Экономический раздел

## Технико-экономическое обоснование разработки программного средства

При рассмотрении вопроса о разработке программного средства для проведения тестирования знаний участников олимпиады необходимо оценить экономическую эффективность проекта. Этого можно достичь с помощью комплексного технико-экономического обоснования, которое обеспечит всесторонний анализ потенциальной жизнеспособности проекта [2].

В технико-экономическом обосновании будут рассмотрены три ключевые области:

* составление плана разработки программного средства, в котором будут указаны основные этапы проекта и необходимые ресурсы;
* расчёт стоимости разработки с учетом всех затрат, связанных с проектом, начиная с получения первой версии технического задания и заканчивая оформлением документации и сдачей разработки;
* оценка экономической эффективности программного средства, принимая во внимание выгоды, которые, как ожидается, это средство принесет организации и ее клиентам.

Чтобы рассчитать экономическую эффективность программного средства, важно использовать комплексный и систематический подход. Технико-экономическое обоснование будет состоять из трех частей. Первая часть будет посвящена расчету периода разработки программного средства с учетом всех факторов, которые могут повлиять на сроки реализации проекта. Вторая часть будет посвящена расчету стоимости разработки с учетом всех прямых и косвенных затрат, связанных с проектом.

Наконец, в третьей части будет проведена оценка экономической эффективности программного средства с учетом как затрат на разработку, так и ожидаемых выгод от использования инструмента.

## Составление плана по разработке программного средства

План разработки программного средства представлен в таблице 6.1.

Таблица 6.1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов и видов работ | Исполнитель | Количество исполнителей | Трудоемкость, человеко-дни |
| Инициализация | техник-программист | 1 | 3 |
| Планирование | техник-программист | 1 | 6 |
| Продолжение таблицы 6.1 | | | |
| Наименование этапов и видов работ | Исполнитель | Количество исполнителей | Трудоемкость, человеко-дни |
| Разработка | техник-программист | 1 | 20 |
| Реализация и тестирование | техник-программист | 1 | 7 |
| Мониторинг и завершение проекта | техник-программист | 1 | 4 |

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

Рисунок 6.1

## Определение цены программного средства

В условиях рыночной экономики научно-техническая продукция рассматривается как товар, поэтому ключевым фактором при проведении технико-экономического обоснования является определение цены основного результата дипломного проекта.

При установлении цены на эту научно-техническую продукцию важно учитывать отраслевые рекомендации, которые следуют варианту механической аналогии с материальным производством, принимая во внимание понесенные или предполагаемые затраты, а также минимальный уровень рентабельности [2].

Чтобы оценить стоимость программного средства и оценить его экономическое воздействие, необходимо рассмотреть ряд элементов в определенном порядке [2]. Этими элементами являются:

* материальные затраты – стоимость любых материалов, необходимых для создания программного средства;
* затраты на оплату труда – заработная плата разработчиков, дизайнеров и другого персонала, задействованного в проекте;
* отчисления на социальное обеспечение – любые необходимые налоги или взносы в программы социального обеспечения;
* амортизация основных средств и нематериальных активов – амортизация любого оборудования или нематериальных активов, использованных при разработке программного средства;
* прочие расходы – любые затраты, не учтенные вышеперечисленными элементами, например, расходы на маркетинг и рекламу.

Расчет цены основного результата дипломного проекта представляет собой систематический процесс. Он начинается с определения материальных затрат, связанных с проектом, включая стоимость любых покупных компонентов или полуфабрикатов, которые могут понадобиться для изготовления макетов и прототипов.

Расчет материальных затрат осуществляется по формуле (6.1)

 (6.1)

где – коэффициент, учитывающий транспортно-заготовительные расходы (в работе принимаем от 1,05 до 1,10);

– норма расхода i-го вида материалов;

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

n – количество применяемых видов материалов.

Расчет целесообразно представить в таблице 6.2.

Таблица 6.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование материалов | Единица измерения | Количество | Цена приобретения, руб. | Сумма, руб. |
| Бумага | пачка | 1 | 13,90 | 16,68 |
| Ручка шариковая | штука | 1 | 2,04 | 2,45 |
| Папка-скоросшиватель | штука | 1 | 0,90 | 1,08 |
| Диск | штука | 1 | 2,50 | 3,00 |
| Всего расходов | | | | 19,34 |
| Всего с транспортно-заготовительными расходами | | | | 21,27 |

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 формуле (6.2)

, (6.2)

где– стоимость одного кВт/ч, руб.;

– количество кВт/ч.

Тарифы на электроэнергию применяются согласно приложения к Декларации *«*Об уровне тарифов на электроэнергию, отпускаемую 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

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 (то есть на 168 часов).

Тарифная ставка первого разряда на предприятии составляет 235 руб. Трудоемкость определяется исходя из данных, представленных в таблице 6.1 (значение столбца «Трудоемкость» умноженное 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, дни или месяцы;

– коэффициент премий по премиальным системам, от 1,10 до 1,30 (1,25).

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, дни или месяцы | Сумма, руб. |
| Техник-программист | 1 | 477,05 | 2 | 921,62 |
| Всего | | | | 954,10 |
| Всего с коэффициентом премий | | | | 1192,63 |

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 %.

Рдз = 1192,63= 119,26 руб.

Рассчитываются отчисления органам социальной защиты по формуле (6.5)

, (6.5)

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

Рос = (1192,63+119,26)= 446,04 руб.

Также рассчитываются отчисления на страхование от несчастных случаев на производстве и профессиональных заболеваний (Остр) 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)

Рстр= (1192,63+119,26)= 7,87 руб.

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 компьютерного оборудования составляют пять лет.

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

, (6.7)

гдеТс – срок службы оборудования, лет.

= 20%.

Для линейного способа начисления амортизационные отчисления равномерно распределены на весь период службы оборудования и вычисляются на один год. Так как разработка программного средства длилась 40 дней, то сумма амортизационных отчислений (АО) за этот период высчитывается по формуле (6.8)

, (6.8)

РАО == 66,67 руб.

Определяются прочие прямые расходы, связанные с амортизационными отчислениями на полное восстановление основных производственных фондов, арендная плата и лизинговые платежи, компенсация за износ (амортизацию) использованного в процессе создания научно-технической продукции оборудования по договоренности. Рпр рассчитывается по формуле (6.9)

, (6.9)

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

Рпр = 1192,63= 238,53 руб.

На основании полученных данных по отдельным статьям затрат рассчитывается общая сумма затрат на разработку программного средства. Общая сумма затрат по элементам на разработку программного средства рассчитывается по формуле (6.10)

, (6.10)

З=21,27+62,79+1192,63+119,26+446,04+7,87+66,67+238,53=2155,06 руб.

Все приведенные выше расчеты целесообразно объединить в сводную таблицу 6.4.

Таблица 6.4 ­ Расчет затрат на разработку программного средства

|  |  |
| --- | --- |
| Элемент затрат | Затраты, руб. |
| Материальные затраты () | 21,27 |
| Электроэнергия () | 62,79 |
| Затраты на оплату труда () | 1192,63 |
| Дополнительная заработная плата () | 119,26 |
| Отчисления органам социальной защиты (); | 446,04 |
| Отчисления на страхования от несчастных случаев () | 7,87 |
| Амортизация () | 66,67 |
| Прочие затраты () | 238,53 |
| Общая сумма затрат ( ) | 2155,06 |

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

Рисунок 6.2

На основании данных, представленных на круговой диаграмме, можно сделать вывод, что себестоимость програмного средства состоит из множества компонентов, каждый из которых выражен в рублях. Самым крупным компонентом затрат являются затраты на оплату труда, общая сумма которых вместе с премией составляет 1192,63. Другие элементы затрат включают материальные затраты, электроэнергию, дополнительный фонд оплаты труда, отчисления в органы социальной защиты, отчисления на социальное страхование, амортизацию, прочие затраты. Общая сумма затрат на разработку про программного средства составила 2155,06. Несмотря на высокую стоимость оплаты труда, другие компоненты, такие как расходы на электроэнергию и отчисления на социальное страхование, также вносят значительный вклад в общую стоимость.

В заключение следует отметить, что для точного определения себестоимости программного средства важно учитывать все эти компоненты и соответствующие им затраты.

## Расчет экономического эффекта у разработчика и пользователя программного средства

### Экономический эффект у разработчика программного средства

Процесс оплаты проекта по разработке программного обеспечения включает в себя оплату клиентом полной стоимости проекта, включая сумму прибыли разработчика. После уплаты необходимых налогов с полученной прибыли заказчику остается чистая прибыль от проекта.

Поскольку программное обеспечение предназначено для конкретной цели или объекта, эту чистую прибыль можно рассматривать как экономическую выгоду разработчика от продажи программного продукта.

Важно отметить, что цена продажи программного средства в данном дипломном проекте определяется не в расчете на единицу продукции, а исходя из стоимости всего проекта и потенциальной выгоды, которую можно получить от его продажи.

Эта цена реализации формируется путем расчета, в котором учитываются различные факторы, включая планируемую себестоимость продукции, установленные налоги и прибыль, качество и потребительские свойства продукта, а также текущие рыночные условия [2].

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 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 на продукцию с учетом НДС рассчитывается по формуле (6.14)

, (6.14)

Таким образом, разработчик программного средства может продать заказчику программное срество, что покроет затраты и обеспечит прибыль за разработку проекта.

### Экономический эффект у пользователя программного средства

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

* повышения 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;
* сокращения 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 [2].

Программа для проведения тестирования знаний участников олимпиады предназначен для использования в частном учреждении образования «Колледж бизнеса и права». Срок разработки составил 320 часов, а стоимость программного средства составила 3103,28 руб.

Автоматизация проведения тестирования знаний участников олимпиады позволит организаторам сосредоточиться на основных обязанностях, повышая производительность и качество олимпиады. Более того, способность программного обеспечения предоставлять персонализированные возможности обучения и немедленную обратную связь может помочь участникам лучше понять материал олимпиады и более рационально подойти к тестированию своих знаний. Разработка программного средства для проведения олимпиад способствует успеху и увеличению числа участников, желающих принять участие в олимпиаде.

Все это делает программное средство ценным для экономики и образования, повышая производительность, улучшая результаты олимпиады и способствуя развитию отрасли.

**Заключение**

В рамках дипломного проекта на тему «Создание программного средства для проведения тестирования знаний участников олимпиады» было разработано программного средство «QuizHut.exe» со следующими возможностями:

* регистрация информации об участниках олимпиады;
* создание и редактирование тестовых заданий и их публикация по таймеру;
* получение решений от участников в заданные сроки;
* задание критериев оценки работ участников олимпиады;
* выгрузка данных об итогах проведения олимпиады;
* получение результатов по разным событиям;
* создание и редактирование структурных единиц программы: категорий, групп;
* просмотр и изменение данных пользователя.

Простота и высокое качество разработки было обеспечено применением технологии .NET, WPF и библиотеки Entity Framwork с использованием среды разработки Microsoft Visual Studio 2022 и языка программирования C#. База данных разработана 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 защите от источников ионизирующего излучений при эксплуатации ПЭВМ.
* пожарная безопасность;
* охрана окружающей среды.

В экономическом разделе была рассчитана стоимость разработки программного средства, которая составила 3103,28 рубля.

Программное средство не имеет недостатков, однако обладает рядом преимуществ:

* использование одного сервера несколькими участниками позволяет им одновременно взаимодействовать с информацией и дает возможность соревноваться в режиме реального времени;
* программа значительно экономит время и сокращает объем работы для участников, освобождая их от необходимости заниматься этими задачами вручную.

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

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. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
5. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
6. 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.
7. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
8. Документация по Windows | Microsoft Learn [Электронный ресурс]. – Microsoft, 2023. – Режим доступа : <https://learn.microsoft.com/en-us/windows/>. –Дата доступа :05.05.2023.
9. 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.
10. 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.
11. Документация по Windows Presentation Foundation для .NET 7 | Microsoft Learn [Электронный ресурс]. – Microsoft, 2023. – Режим доступа : https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netdesktop.7.0. – Дата доступа : 18.05.2023.
12. Защита от ионизирующего излучения. [Электронный ресурс] / Охрана труда. – Режим доступа : https://www.protrud.com/обучение/учебный-курс/защита-от-ионизирующего-излучения/. – Дата доступа : 25.04.2023.
13. 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.
14. draw.io Документация [Электронный ресурс]. –JGraph Ltd, 2023. – Режим доступа :  https://www.drawio.com/doc/. – Дата доступа :18.04.2023.
15. 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 rightsreserved,  2004 - 2023. – Режим доступа : https://www.drexplain.ru/. ­ Дата доступа :05.05.2023.
16. draw.io Документация [Электронный ресурс]. –JGraph Ltd, 2023. – Режим доступа :  https://www.drawio.com/doc/. – Дата доступа :18.04.2023.
17. MicrosoftOfficeExcel­ [Электронный ресурс]. – ОфисГуруНьюс, 2023. –Режим доступа : <https://office-guru.ru/excel/microsoft-office-excel-chto-eto-59.html>.– Дата доступа :05.05.2023.
18. Model-View-ViewModel | Microsoft Learn Документация [Электронный ресурс]. – Microsoft, 2023. – Режим доступа :  https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm. –Дата доступа :22.04.2023.
19. 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**

class MainViewModel : DialogViewModel

{

//Декларация переменной сервиса аккаунта пользователей

private readonly IUserAccountService userAccountService;

//Декларация переменной сервиса навигации

private readonly INavigationService navigationService;

//Декларация переменной сервиса данных аккаунта

private readonly IAccountStore accountStore;

//Декларация переменной сервиса хранилища данных

private readonly ISharedDataStore sharedDataStore;

//Конструктор класса MainViewModel

public MainViewModel(

IUserAccountService userAccountService,

INavigationService navigationService,

IAccountStore accountStore,

ISharedDataStore sharedDataStore,

IViewModelFactory traderViewModelFactory)

{

//Инициализация переменных конструктора

this.userAccountService = userAccountService;

this.navigationService = navigationService;

this.accountStore = accountStore;

this.sharedDataStore = sharedDataStore;

//Установка методов изменения состояния сервисов

navigationService.StateChanged += NavigationService\_StateChanged;

accountStore.StateChanged += UserAccountService\_StateChanged;

//Инициализация команды навигации к профилю

NavigateProfileCommand = new ActionCommand(p => ProfileNavigate());

//Инициализация общей команды навигации

NavigationCommand = new NavigationCommand(navigationService, traderViewModelFactory);

NavigationCommand.Execute(ViewType.Authorization);//Выполнение навигации на страницу авторизации

//Инициализация команды выхода из аккаунта

LogoutCommand = new ActionCommand(OnLogoutCommandExecuted);

//Проверка роли пользователя для установки активной кнопки

if(CurrentUserRole == UserRole.Student)

{

SelectedOption = "StudentHome";

}

}

//Переменные и свойства класса

#region Fields and properties

//Свойство текущего окна, инициализируемое сервисом навигации

public ViewModel CurrentView => navigationService.CurrentView;

//Свойство роли пользователя, инициализируемое сервисом данных аккаунта

public UserRole CurrentUserRole => accountStore.CurrentUserRole;

//Свойство текущего пользователя

public ApplicationUser CurrentUser

{

get

{

if (accountStore.CurrentUser != null)

{

sharedDataStore.CurrentUser = accountStore.CurrentUser;

return accountStore.CurrentUser;

}

return null;

}

}

//Переменная и свойство заголовка

private string title = HomeViewModel.Title;

public string Title

{

get => title;

set => Set(ref title, value);

}

//Переменная и свойство иконки заголовка

private IconChar iconChar = HomeViewModel.IconChar;

public IconChar IconChar

{

get => iconChar;

set => Set(ref iconChar, value);

}

//Переменная и свойство выбранной кнопки меню

private string selectedOption = "Home";

public string? SelectedOption

{

get => selectedOption;

set => Set(ref selectedOption, value);

}

#endregion

//Команды навигации

#region NavigationCommands

//Общая команда навигации

public NavigationCommand NavigationCommand { get; }

//Команда навигации к странице профиля

public ICommand NavigateProfileCommand { get; }

#endregion

//Команда выхода из профиля

#region LogoutCommand

//Декларация команды

public ICommand LogoutCommand { get; }

//Метод, вызываемый при срабатывании команды

private void OnLogoutCommandExecuted(object p)

{

userAccountService.Logout();

NavigationCommand.Execute(ViewType.Authorization);

}

#endregion

//Метод для установки заголовка и иконки заголовка

private void ShowingContent()

{

if(CurrentView is IMenuView menuView)

{

Title = menuView.GetType().GetProperty("Title").GetValue(null).ToString();

IconChar = (IconChar)menuView.GetType().GetProperty("IconChar").GetValue(null);

}

}

//Метод, вызываемый при изменении состояния UserAccountService

private void UserAccountService\_StateChanged()

{

OnPropertyChanged(nameof(CurrentUserRole));

OnPropertyChanged(nameof(CurrentUser));

}

//Метод, вызываемый при изменении состояния NavigationService

private void NavigationService\_StateChanged()

{

OnPropertyChanged(nameof(CurrentView));

ShowingContent();

}

//Метод, вызываемый при срабатывании команды NavigateProfileCommand

private void ProfileNavigate()

{

SelectedOption = null;

NavigationCommand.Execute(ViewType.UserProfile);

}

//Переопределённый метод очистки некоторых свойств класса

public override void Dispose()

{

navigationService.StateChanged -= NavigationService\_StateChanged;

accountStore.StateChanged -= UserAccountService\_StateChanged;

base.Dispose();

}

}

}

//MainView.xaml

<Window x:Class="QuizHut.Views.Windows.MainView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:QuizHut.Views.Windows" mc:Ignorable="d"

xmlns:fa="http://schemas.awesome.incremented/wpf/xaml/fontawesome.sharp"

xmlns:controls="clr-namespace:QuizHut.Infrastructure.UserControls"

xmlns:nav="clr-namespace:QuizHut.Infrastructure.Services.Contracts"

Title="QuizzesHut" Height="750" Width="1300"

WindowStartupLocation="CenterScreen"

WindowStyle="None"

Background="Transparent"

AllowsTransparency="True"

Icon="\Images\main\_icon.ico">

<Border BorderThickness="2"

CornerRadius="10">

<Border.BorderBrush>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="#052661" Offset="0"/>

<GradientStop Color="#691257" Offset="0.65"/>

<GradientStop Color="#CF1A76" Offset="1"/>

</LinearGradientBrush>

</Border.BorderBrush>

<Grid>

<Grid.ColumnDefinitions>

<!-- Nagivation menu -->

<ColumnDefinition Width="250"/>

<!-- Content section -->

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!-- Navigation menu -->

<Border CornerRadius="10,0,0,10"

Grid.Column="0"

Background="{StaticResource SecondaryMainViewBackround}">

<Grid Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="70"/>

<RowDefinition Height="5"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="90"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0">

<!-- Title -->

<TextBlock Height="40"

Opacity="0.9"

Margin="20,10,0,0"

Text="QUIZZES HUT"

FontSize="28"

FontFamily="{StaticResource Manrope}"

FontWeight="Medium"

Foreground="{StaticResource SecondaryFontColor}"/>

<TextBlock Height="20"

Margin="20,0,0,0"

Text="Версия 1.0"

FontSize="12"

FontFamily="{StaticResource Manrope}"

FontWeight="Medium"

Foreground="{StaticResource SecondaryFontColor}"/>

</StackPanel>

<Separator Grid.Row="1" Background="{StaticResource PrimaryActiveColor}"/>

<!-- Unauthorized user block -->

<StackPanel Margin="0,150,0,0" Grid.Row="2">

<Image Source="\Images\LoginViewImages\MainLogo.png"

Height="90" Width="150"/>

<TextBlock Text="Добро пожаловать!

Для начала авторизуйтесь в системе, либо создайте аккаунт если он отсуствует.

Если возникли вопросы либо проблемы воспользуйтесь справкой."

TextWrapping="Wrap" TextAlignment="Center"

FontSize="16" FontFamily="{StaticResource Manrope}" FontWeight="SemiBold" Foreground="{StaticResource SecondaryFontColor}"

Margin="10"/>

<StackPanel.Style>

<Style TargetType="StackPanel">

<Style.Triggers>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Student">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Teacher">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

</Style.Triggers>

</Style>

</StackPanel.Style>

</StackPanel>

<!-- Teacher navigation menu -->

<controls:TeacherNavigationMenu Grid.Row="2" Margin="0,40">

<controls:TeacherNavigationMenu.Style>

<Style TargetType="controls:TeacherNavigationMenu">

<Style.Triggers>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Student">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Unauthorised">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

</Style.Triggers>

</Style>

</controls:TeacherNavigationMenu.Style>

</controls:TeacherNavigationMenu>

<!-- Student navigation menu -->

<controls:StudentNavigationMenu Grid.Row="2" Margin="0,40">

<controls:StudentNavigationMenu.Style>

<Style TargetType="controls:StudentNavigationMenu">

<Style.Triggers>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Teacher">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Unauthorised">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

</Style.Triggers>

</Style>

</controls:StudentNavigationMenu.Style>

</controls:StudentNavigationMenu>

<!-- Logout button -->

<Button x:Name="MVLogoutBtn"

HorizontalAlignment="Left"

Margin="20"

Grid.Row="3"

Tag="{StaticResource Color8}"

Command="{Binding LogoutCommand}">

<StackPanel Orientation="Horizontal">

<TextBlock Text="Выход" Style="{StaticResource UserOptionButtonText}"/>

<fa:IconImage Icon="RightFromBracket"

Style="{StaticResource UserOptionButtonIcon}"

Margin="3,0,0,0"/>

</StackPanel>

<Button.Style>

<Style BasedOn="{StaticResource UserOptionButton}" TargetType="Button">

<Style.Triggers>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Unauthorised">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

</Style.Triggers>

</Style>

</Button.Style>

</Button>

<!-- Help button -->

<Button x:Name="MVHelpBtn"

HorizontalAlignment="Right"

Margin="20"

Grid.Row="3"

Style="{StaticResource UserOptionButton}"

Tag="{StaticResource Color6}"

Command="{Binding ShowHelpCommand}">

<StackPanel Orientation="Horizontal">

<TextBlock Text="Справка" Style="{StaticResource UserOptionButtonText}"/>

<fa:IconImage Icon="CircleInfo"

Style="{StaticResource UserOptionButtonIcon}"

Margin="3,0,0,0"/>

</StackPanel>

</Button>

</Grid>

</Border>

<!-- Main Section -->

<Border CornerRadius="0,10,10,0"

Grid.Column="1"

Background="{StaticResource PrimaryMainViewBackround}">

<Grid>

<Grid.RowDefinitions>

<!-- Control bar -->

<RowDefinition Height="30"/>

<!-- Header / Caption -->

<RowDefinition Height="40"/>

<RowDefinition Height="5"/>

<!-- Content section / Child views -->

<RowDefinition/>

</Grid.RowDefinitions>

<!-- Control bar -->

<StackPanel x:Name="MVControlBar"

Grid.Row="0"

Orientation="Horizontal"

FlowDirection="RightToLeft"

Background="Transparent"

Margin="0,0,5,0"

MouseLeftButtonDown="MVControlBar\_MouseLeftButtonDown">

<Button x:Name="MVCLoseBtn"

Style="{StaticResource ControlButton}"

Tag="{StaticResource Color3}"

Command="{StaticResource CloseApplicationCommand}">

<fa:IconImage Icon="Xmark" Style="{StaticResource MainControlButtonIcon}"/>

</Button>

<Button x:Name="MVMaximizeBtn"

Style="{StaticResource ControlButton}"

Tag="{StaticResource Color4}"

Command="{StaticResource MaximizeApplicationCommand}">

<fa:IconImage Icon="WindowMaximize" Style="{StaticResource MainControlButtonIcon}"/>

</Button>

<Button x:Name="MVMinimizeBtn"

Style="{StaticResource ControlButton}"

Tag="{StaticResource Color5}"

Command="{StaticResource MinimizeApplicationCommand}">

<fa:IconImage Icon="WindowMinimize" Width="12"

Style="{StaticResource MainControlButtonIcon}"/>

</Button>

</StackPanel>

<!-- Caption -->

<StackPanel Grid.Row="1"

Orientation="Horizontal"

VerticalAlignment="Center">

<fa:IconImage Icon="{Binding IconChar, UpdateSourceTrigger=PropertyChanged}"

Height="25" Width="25"

Foreground="{StaticResource PrimaryFontColor}"

Margin="15,0,10,0"/>

<TextBlock Text="{Binding Title, UpdateSourceTrigger=PropertyChanged}"

Foreground="{StaticResource PrimaryFontColor}"

FontSize="24" FontWeight="Medium" FontFamily="{StaticResource Manrope}"

VerticalAlignment="Center"/>

<StackPanel.Style>

<Style TargetType="StackPanel">

<Style.Triggers>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Unauthorised">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

</Style.Triggers>

</Style>

</StackPanel.Style>

</StackPanel>

<!-- User options -->

<StackPanel Grid.Row="1"

Orientation="Horizontal"

FlowDirection="RightToLeft"

VerticalAlignment="Center"

Margin="0,0,10,0">

<!-- Profile button -->

<Button x:Name="MVProfileBtn"

Tag="{StaticResource Color9}"

Margin="10,0,0,0"

Command="{Binding NavigateProfileCommand}">

<StackPanel Orientation="Horizontal">

<TextBlock Text="Профиль" Style="{StaticResource UserOptionButtonText}"/>

<fa:IconImage Icon="AddressCard"

Style="{StaticResource UserOptionButtonIcon}"

Margin="3,0,0,0"/>

</StackPanel>

<Button.Style>

<Style TargetType="Button" BasedOn="{StaticResource UserOptionButton}">

<Style.Triggers>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Unauthorised">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

</Style.Triggers>

</Style>

</Button.Style>

</Button>

<!-- User name and role -->

<TextBlock

FontSize="16" FontWeight="Medium" FontFamily="{StaticResource Manrope}"

Foreground="{StaticResource PrimaryFontColor}"

VerticalAlignment="Center"

Margin="10,0,0,0">

<Run Text="{Binding CurrentUser.FirstName, StringFormat={}{0} }"/>

<Run Text="{Binding CurrentUser.LastName}"/>

</TextBlock>

<StackPanel.Style>

<Style TargetType="StackPanel">

<Style.Triggers>

<DataTrigger Binding="{Binding CurrentUserRole}" Value="Unauthorised">

<Setter Property="Visibility" Value="Collapsed"/>

</DataTrigger>

</Style.Triggers>

</Style>

</StackPanel.Style>

</StackPanel>

<Separator Grid.Row="2" Background="{StaticResource menuOverColor}"/>

<!-- Main content -->

<ContentControl Content="{Binding CurrentView}"

Grid.Row="3"

Margin="10"

VerticalContentAlignment="Center" HorizontalContentAlignment="Center"/>

</Grid>

</Border>

</Grid>

</Border>

</Window>

//MainViewModel.cs

using System.Windows.Input;

using FontAwesome.Sharp;

using QuizHut.BLL.Helpers.Contracts;

using QuizHut.BLL.Services.Contracts;

using QuizHut.DLL.Entities;

using QuizHut.Infrastructure.Commands;

using QuizHut.Infrastructure.Services.Contracts;

using QuizHut.ViewModels.Base;

using QuizHut.ViewModels.Contracts;

using QuizHut.ViewModels.Factory;

using QuizHut.ViewModels.MainViewModels.TeacherPartViewModels;

class MainViewModel : ViewModel

{

private readonly IUserAccountService userAccountService;

private readonly INavigationService navigationService;

private readonly IAccountStore accountStore;

private readonly ISharedDataStore sharedDataStore;

public MainViewModel(

IUserAccountService userAccountService,

INavigationService navigationService,

IAccountStore accountStore,

ISharedDataStore sharedDataStore,

IViewModelFactory traderViewModelFactory)

{

this.userAccountService = userAccountService;

this.navigationService = navigationService;

this.accountStore = accountStore;

this.sharedDataStore = sharedDataStore;

navigationService.StateChanged += NavigationService\_StateChanged;

accountStore.StateChanged += UserAccountService\_StateChanged;

NavigateProfileCommand = new ActionCommand(p => ProfileNavigate());

NavigationCommand = new NavigationCommand(navigationService, traderViewModelFactory);

NavigationCommand.Execute(ViewType.Authorization);

LogoutCommand = new ActionCommand(OnLogoutCommandExecuted);

ShowHelpCommand = new ActionCommand(OnShowHelpCommandExecuted);

}

#region Fields and properties

public ViewModel CurrentView => navigationService.CurrentView;

public UserRole CurrentUserRole => accountStore.CurrentUserRole;

public ApplicationUser CurrentUser

{

get

{

if (accountStore.CurrentUser != null)

{

sharedDataStore.CurrentUser = accountStore.CurrentUser;

return accountStore.CurrentUser;

}

return null;

}

}

private string title = HomeViewModel.Title;

public string Title

{

get => title;

set => Set(ref title, value);

}

private IconChar iconChar = HomeViewModel.IconChar;

public IconChar IconChar

{

get => iconChar;

set => Set(ref iconChar, value);

}

private string? selectedOption = "Home";

public string? SelectedOption

{

get => selectedOption;

set => Set(ref selectedOption, value);

}

#endregion

#region NavigationCommands

public NavigationCommand NavigationCommand { get; }

public ICommand NavigateProfileCommand { get; }

#endregion

#region LogoutCommand

public ICommand LogoutCommand { get; }

private void OnLogoutCommandExecuted(object p)

{

userAccountService.Logout();

NavigationCommand.Execute(ViewType.Authorization);

}

#endregion

#region ShowHelpCommand

public ICommand ShowHelpCommand { get; }

private void OnShowHelpCommandExecuted(object p)

{

System.Windows.Forms.Help.ShowHelp(null, @"QuizzesHutHelp.chm");

}

#endregion

private void ShowingContent()

{

if(CurrentView is IMenuView menuView)

{

Title = menuView.GetType().GetProperty("Title").GetValue(null).ToString();

IconChar = (IconChar)menuView.GetType().GetProperty("IconChar").GetValue(null);

}

}

private void UserAccountService\_StateChanged()

{

OnPropertyChanged(nameof(CurrentUserRole));

OnPropertyChanged(nameof(CurrentUser));

}

private void NavigationService\_StateChanged()

{

OnPropertyChanged(nameof(CurrentView));

ShowingContent();

}

private void ProfileNavigate()

{

NavigationCommand.Execute(ViewType.UserProfile);

}

public override void Dispose()

{

navigationService.StateChanged -= NavigationService\_StateChanged;

accountStore.StateChanged -= UserAccountService\_StateChanged;

base.Dispose();

}

}

//StartQuizView.xaml

<UserControl x:Class="QuizHut.Views.UserControls.MainUserControls.TeacherPartUserControls.QuizUserControls.PassingQuizUserControls.StartQuizView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:fa="http://schemas.awesome.incremented/wpf/xaml/fontawesome.sharp"

mc:Ignorable="d"

Height="400" Width="800"

HorizontalAlignment="Left" VerticalAlignment="Top"

Margin="50,50,0,0">

<UserControl.Effect>

<DropShadowEffect Color="Black" BlurRadius="20" Direction="315" Opacity="0.5"/>

</UserControl.Effect>

<Border Background="{StaticResource MainViewBackground4}"

CornerRadius="15">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="15"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="15"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="15"/>

<RowDefinition Height="40"/>

<RowDefinition Height="20"/>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

<RowDefinition Height="55"/>

<RowDefinition Height="15"/>

</Grid.RowDefinitions>

<!-- Icon and Title -->

<StackPanel Orientation="Horizontal"

Grid.Row="1" Grid.Column="1">

<fa:IconImage Grid.Row="1" Grid.Column="1"

Width="40" Height="40" HorizontalAlignment="Left" VerticalAlignment="Center"

Icon="ListOl" Foreground="{StaticResource IconColor1}"/>

<TextBlock Style="{StaticResource MainTextBlock}" Text="Викторина:"

FontSize="22" Foreground="{StaticResource FontColor4}"

VerticalAlignment="Bottom" Margin="5,0,0,0"/>

<TextBlock Style="{StaticResource MainTextBlock}" Text="{Binding CurrentQuiz.Name}"

FontSize="22" VerticalAlignment="Bottom" Margin="10,0,0,0"/>

</StackPanel>

<!-- Questions quantity -->

<StackPanel Orientation="Horizontal"

Grid.Row="3" Grid.Column="1">

<TextBlock Style="{StaticResource MainTextBlock}" Text="Количество вопросов:"

FontSize="18" Foreground="{StaticResource FontColor4}"/>

<TextBlock Style="{StaticResource MainTextBlock}" Text="{Binding CurrentQuiz.Questions.Count}"

FontSize="22" Margin="10,0,0,0"/>

</StackPanel>

<!-- Timer -->

<StackPanel Orientation="Horizontal"

Grid.Row="4" Grid.Column="1">

<TextBlock Style="{StaticResource MainTextBlock}" Text="Время для прохождения:"

FontSize="18" Foreground="{StaticResource FontColor4}"/>

<TextBlock Style="{StaticResource MainTextBlock}" Text="{Binding CurrentQuiz.Timer}"

FontSize="22" Margin="10,0,0,0"/>

</StackPanel>

<!-- Question description -->

<TextBox x:Name="SQVQuestionDescriptionTB"

Text="{Binding CurrentQuiz.Description}"

Grid.Row="5" Grid.Column="1"

Style="{StaticResource textBoxStyle}"

Height="150" FontSize="15" MaxLength="300" TextWrapping="Wrap" TextAlignment="Justify"

VerticalContentAlignment="Top" IsEnabled="False"/>

<!-- Info message -->

<TextBlock Grid.Row="6" Grid.Column="1"

Style="{StaticResource ErrorTextBlock1}" FontSize="16"

Text="\* помните что вы не сможете вернутся если вы случайно закроете викторину"/>

<!-- Start/Cancel button -->

<Button x:Name="SQVStartQuizBtn" Grid.Row="7" Grid.Column="1"

Template="{StaticResource MainButtonTemplate}"

Background="{StaticResource PrimaryActiveColor}"

HorizontalAlignment="Left" VerticalAlignment="Center"

Width="320" Height="50"

Content="Начать викторину"

Foreground="{StaticResource SecondaryFontColor}" FontSize="22" FontWeight="SemiBold"

Command="{Binding NavigateTakingQuizAsyncCommand}"/>

<Button x:Name="SQVCancelBtn" Grid.Row="7" Grid.Column="1"

Template="{StaticResource MainButtonTemplate}"

Background="{StaticResource SecondaryActiveColor}"

HorizontalAlignment="Right" VerticalAlignment="Center"

Width="320" Height="50"

Content="Назад"

Foreground="{StaticResource SecondaryFontColor}" FontSize="22" FontWeight="SemiBold"

Command="{Binding NavigateHomeCommand}"/>

</Grid>

</Border>

</UserControl>

//StartQuizViewModel.cs

using System;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Input;

using QuizHut.BLL.Helpers.Contracts;

using QuizHut.BLL.Services.Contracts;

using QuizHut.Infrastructure.Commands;

using QuizHut.Infrastructure.Commands.Base;

using QuizHut.Infrastructure.Commands.Base.Contracts;

using QuizHut.Infrastructure.EntityViewModels.Quizzes;

using QuizHut.Infrastructure.Services.Contracts;

using QuizHut.ViewModels.Base;

class StartQuizViewModel : ViewModel

{

private readonly IResultsService resultsService;

private readonly IShuffler shuffler;

private readonly ISharedDataStore sharedDataStore;

private readonly IRenavigator takingQuizRenavigator;

private readonly IAccountStore accountStore;

private readonly IRenavigator homeRenavigator;

private readonly IRenavigator studentHomeRenavigator;

public StartQuizViewModel(

IResultsService resultsService,

IShuffler shuffler,

ISharedDataStore sharedDataStore,

IRenavigator takingQuizRenavigator,

IAccountStore accountStore,

IRenavigator homeRenavigator,

IRenavigator studentHomeRenavigator)

{

this.resultsService = resultsService;

this.shuffler = shuffler;

this.sharedDataStore = sharedDataStore;

this.accountStore = accountStore;

this.takingQuizRenavigator = takingQuizRenavigator;

this.homeRenavigator = homeRenavigator;

this.studentHomeRenavigator = studentHomeRenavigator;

NavigateTakingQuizAsyncCommand = new ActionCommandAsync(OnNavigateTakingQuizAsyncCommandExecute);

NavigateHomeCommand = new ActionCommand(OnNavigateHomeCommandExecuted);

}

#region Fields and properties

private AttemptedQuizViewModel currentQuiz;

public AttemptedQuizViewModel CurrentQuiz

{

get

{

currentQuiz = sharedDataStore.QuizToPass;

return currentQuiz;

}

set => Set(ref currentQuiz, value);

}

#endregion

#region NavigationCommands

public ICommandAsync NavigateTakingQuizAsyncCommand { get; }

private async Task OnNavigateTakingQuizAsyncCommandExecute(object p)

{

sharedDataStore.QuizToPass.Questions = sharedDataStore.QuizToPass.Questions.OrderBy(q => q.Number).ToList();

foreach (var question in sharedDataStore.QuizToPass.Questions)

{

question.Answers = shuffler.Shuffle(question.Answers);

}

sharedDataStore.RemainingTime = TimeSpan.FromMinutes(CurrentQuiz.Timer);

if (sharedDataStore.CurrentUserRole == UserRole.Student)

{

sharedDataStore.CurrentResultId = await resultsService.CreateResultAsync(sharedDataStore.CurrentUser.Id, sharedDataStore.QuizToPass.Id);

}

takingQuizRenavigator.Renavigate();

}

public ICommand NavigateHomeCommand { get; }

private void OnNavigateHomeCommandExecuted(object p)

{

if (accountStore.CurrentUserRole == UserRole.Student)

{

studentHomeRenavigator.Renavigate();

}

else

{

homeRenavigator.Renavigate();

}

}

#endregion

}

//TakingQuizView.xaml

<UserControl x:Class="QuizHut.Views.UserControls.MainUserControls.TeacherPartUserControls.QuizUserControls.PassingQuizUserControls.TakingQuizView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:fa="http://schemas.awesome.incremented/wpf/xaml/fontawesome.sharp"

mc:Ignorable="d"

Height="610" Width="900"

HorizontalAlignment="Left" VerticalAlignment="Top"

Margin="50,00,0,0">

<UserControl.Effect>

<DropShadowEffect Color="Black" BlurRadius="20" Direction="315" Opacity="0.5"/>

</UserControl.Effect>

<StackPanel>

<TextBlock Style="{StaticResource MainTextBlock}"

Text="{Binding TimeRemainingText}" FontSize="40"

Height="50"

VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border Background="{StaticResource MainViewBackground4}"

CornerRadius="15"

Height="550" Width="900"

Margin="0,10,0,0">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="15"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="15"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="15"/>

<RowDefinition Height="30"/>

<RowDefinition Height="10"/>

<RowDefinition Height="auto" MinHeight="50" MaxHeight="110"/>

<RowDefinition Height="5"/>

<RowDefinition Height="auto" MinHeight="150" MaxHeight="330"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="55"/>

<RowDefinition Height="35"/>

<RowDefinition Height="15"/>

</Grid.RowDefinitions>

<!-- Icon and Title -->

<StackPanel Orientation="Horizontal"

Grid.Row="1" Grid.Column="1">

<fa:IconImage Grid.Row="1" Grid.Column="1"

Width="30" Height="30" HorizontalAlignment="Left" VerticalAlignment="Center"

Icon="CircleQuestion" Foreground="{StaticResource IconColor1}"/>

<TextBlock Style="{StaticResource MainTextBlock}"

FontSize="22" Foreground="{StaticResource FontColor4}"

VerticalAlignment="Bottom" Margin="5,0,0,0">

<Run Text="{Binding CurrentQuestion.Number}"/>

<Run Text="/"/>

<Run Text="{Binding Questions.Count, Mode=OneWay}"/>

</TextBlock>

</StackPanel>

<!-- Question -->

<TextBlock Grid.Row="3" Grid.Column="1"

Style="{StaticResource MainTextBlock}"

Text="{Binding CurrentQuestion.Text}"

FontSize="20" Foreground="{StaticResource PrimaryFontColor}"

TextAlignment="Justify" TextWrapping="Wrap" VerticalAlignment="Top"/>

<Separator Grid.Row="4" Grid.Column="1"

Background="{StaticResource IconColor2}"/>

<!-- Answers -->

<ScrollViewer x:Name="TQVAnswersSV"

Grid.Row="5" Grid.Column="1"

CanContentScroll="False"

HorizontalScrollBarVisibility="Disabled"

Background="Transparent"

Margin="0,5,0,5"

Foreground="{StaticResource PrimaryFontColor}">

<ItemsControl ItemsSource="{Binding CurrentQuestion.Answers}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<WrapPanel Orientation="Horizontal" Margin="0,5" >

<CheckBox IsChecked="{Binding IsRightAnswerAssumption}"

VerticalAlignment="Center"/>

<TextBlock Style="{StaticResource MainTextBlock}"

VerticalAlignment="Center"

TextWrapping="Wrap" TextAlignment="Left"

Text="{Binding Text}" Margin="10,0"/>

</WrapPanel>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

<!-- Previous/Next/Cancel button -->

<Button x:Name="TQVPreviousBtn" Grid.Row="7" Grid.Column="1"

Command="{Binding GoToPreviousQuestionCommand}"

Template="{StaticResource MainButtonTemplate}"

Background="{StaticResource PrimaryActiveColor}"

HorizontalAlignment="Left"

Width="350" Height="50"

VerticalAlignment="Center"

Content="Предыдущий"

Foreground="{StaticResource SecondaryFontColor}" FontSize="22" FontWeight="SemiBold"/>

<Button x:Name="TQVNextlBtn" Grid.Row="7" Grid.Column="1"

Command="{Binding GoToNextQuestionCommand}"

Template="{StaticResource MainButtonTemplate}"

Background="{StaticResource menuOverColor}"

HorizontalAlignment="Right"

Width="350" Height="50"

VerticalAlignment="Center"

Content="Следующий"

Foreground="{StaticResource SecondaryFontColor}" FontSize="22" FontWeight="SemiBold"/>

<Button x:Name="TQVFinishQuizBtn" Grid.Row="8" Grid.Column="1"

Template="{StaticResource ButtonTemplate3}"

Foreground="{StaticResource PrimaryActiveColor}" Height="35"

VerticalAlignment="Center" HorizontalAlignment="Center" Margin="0,10,0,0"

FontFamily="{StaticResource Manrope}" FontSize="22" FontWeight="SemiBold"

Command="{Binding StopTimerAndGoToEndQuizCommand}">

<Button.Content>

<TextBlock Style="{StaticResource TextBlockStyle1}">

Закончить викторину

</TextBlock>

</Button.Content>

</Button>

</Grid>

</Border>

</StackPanel>

</UserControl>

//TakingQuizViewModel.cs

using System;

using System.Collections.ObjectModel;

using System.Linq;

using System.Windows.Input;

using System.Windows.Threading;

using QuizHut.Infrastructure.Commands;

using QuizHut.Infrastructure.EntityViewModels.Questions;

using QuizHut.Infrastructure.Services.Contracts;

using QuizHut.ViewModels.Base;

class TakingQuizViewModel : ViewModel

{

private readonly ISharedDataStore sharedDataStore;

private readonly IRenavigator nextQuestionRenavigator;

private readonly IRenavigator endQuizRenavigator;

public TakingQuizViewModel(

ISharedDataStore sharedDataStore,

IRenavigator nextQuestionRenavigator,

IRenavigator endQuizRenavigator)

{

this.sharedDataStore = sharedDataStore;

this.nextQuestionRenavigator = nextQuestionRenavigator;

this.endQuizRenavigator = endQuizRenavigator;

Questions = new(sharedDataStore.QuizToPass.Questions);

CurrentQuestion = sharedDataStore.CurrentQuestion ?? Questions.First();

NavigateEndQuizCommand = new RenavigateCommand(endQuizRenavigator);

GoToNextQuestionCommand = new ActionCommand(OnGoToNextQuestionCommandExecuted, CanGoToNextQuestionCommandExecute);

GoToPreviousQuestionCommand = new ActionCommand(OnGoToPreviousQuestionCommandExecuted, CanGoToPreviousQuestionCommandExecute);

StopTimerAndGoToEndQuizCommand = new ActionCommand(OnStopTimerAndGoToEndQuizCommandExecuted);

SetupTimer();

}

#region Fields and properties

private DispatcherTimer timer;

public DispatcherTimer Timer

{

get => timer;

set => Set(ref timer, value);

}

private ObservableCollection<AttemptedQuizQuestionViewModel> questions;

public ObservableCollection<AttemptedQuizQuestionViewModel> Questions

{

get => questions;

set => Set(ref questions, value);

}

private AttemptedQuizQuestionViewModel currentQuestion;

public AttemptedQuizQuestionViewModel CurrentQuestion

{

get => currentQuestion;

set => Set(ref currentQuestion, value);

}

private TimeSpan timeRemaining;

public string TimeRemainingText => timeRemaining.ToString(@"hh\:mm\:ss");

#endregion

#region NavigationCommands

public ICommand NavigateEndQuizCommand { get; }

#endregion

#region GoToNextQuestionCommand

public ICommand GoToNextQuestionCommand { get; }

private bool CanGoToNextQuestionCommandExecute(object p)

{

return Questions.IndexOf(CurrentQuestion) < Questions.Count - 1;

}

private void OnGoToNextQuestionCommandExecuted(object p)

{

var currentIndex = Questions.IndexOf(CurrentQuestion);

sharedDataStore.CurrentQuestion = Questions[currentIndex + 1];

sharedDataStore.RemainingTime = timeRemaining;

nextQuestionRenavigator.Renavigate();

}

#endregion

#region GoToPreviousQuestionCommand

public ICommand GoToPreviousQuestionCommand { get; }

private bool CanGoToPreviousQuestionCommandExecute(object p)

{

return Questions.IndexOf(CurrentQuestion) > 0;

}

private void OnGoToPreviousQuestionCommandExecuted(object p)

{

var currentIndex = Questions.IndexOf(CurrentQuestion);

sharedDataStore.CurrentQuestion = Questions[currentIndex - 1];

sharedDataStore.RemainingTime = timeRemaining;

nextQuestionRenavigator.Renavigate();

}

#endregion

#region StopTimerAndGoToEndQuizCommand

public ICommand StopTimerAndGoToEndQuizCommand { get; }

private void OnStopTimerAndGoToEndQuizCommandExecuted(object p)

{

EndQuiz();

}

#endregion

#region TimerCommand

private void TimerTick(object sender, EventArgs e)

{

timeRemaining = timeRemaining.Subtract(TimeSpan.FromSeconds(1));

if (timeRemaining <= TimeSpan.Zero)

{

EndQuiz();

}

else

{

OnPropertyChanged(nameof(TimeRemainingText));

}

}

private void StartTimer()

{

timeRemaining = sharedDataStore.RemainingTime;

Timer.Start();

}

private void EndQuiz()

{

Timer.Stop();

sharedDataStore.RemainingTime = TimeSpan.FromMinutes(sharedDataStore.QuizToPass.Timer) - timeRemaining;

endQuizRenavigator.Renavigate();

}

private void SetupTimer()

{

if (sharedDataStore.DispatcherTimer == null)

{

sharedDataStore.DispatcherTimer = new DispatcherTimer

{

Interval = TimeSpan.FromSeconds(1)

};

}

Timer = sharedDataStore.DispatcherTimer;

Timer.Tick += TimerTick;

if (sharedDataStore.RemainingTime > TimeSpan.Zero)

{

StartTimer();

}

}

#endregion

}

**ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Результаты работы программы**

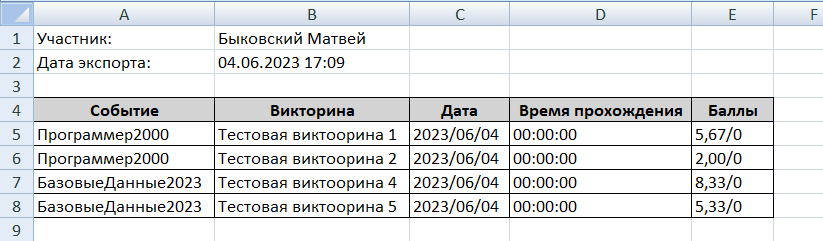
****

Рисунок Б.1

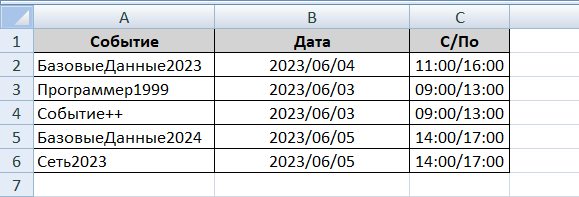


Рисунок Б.2

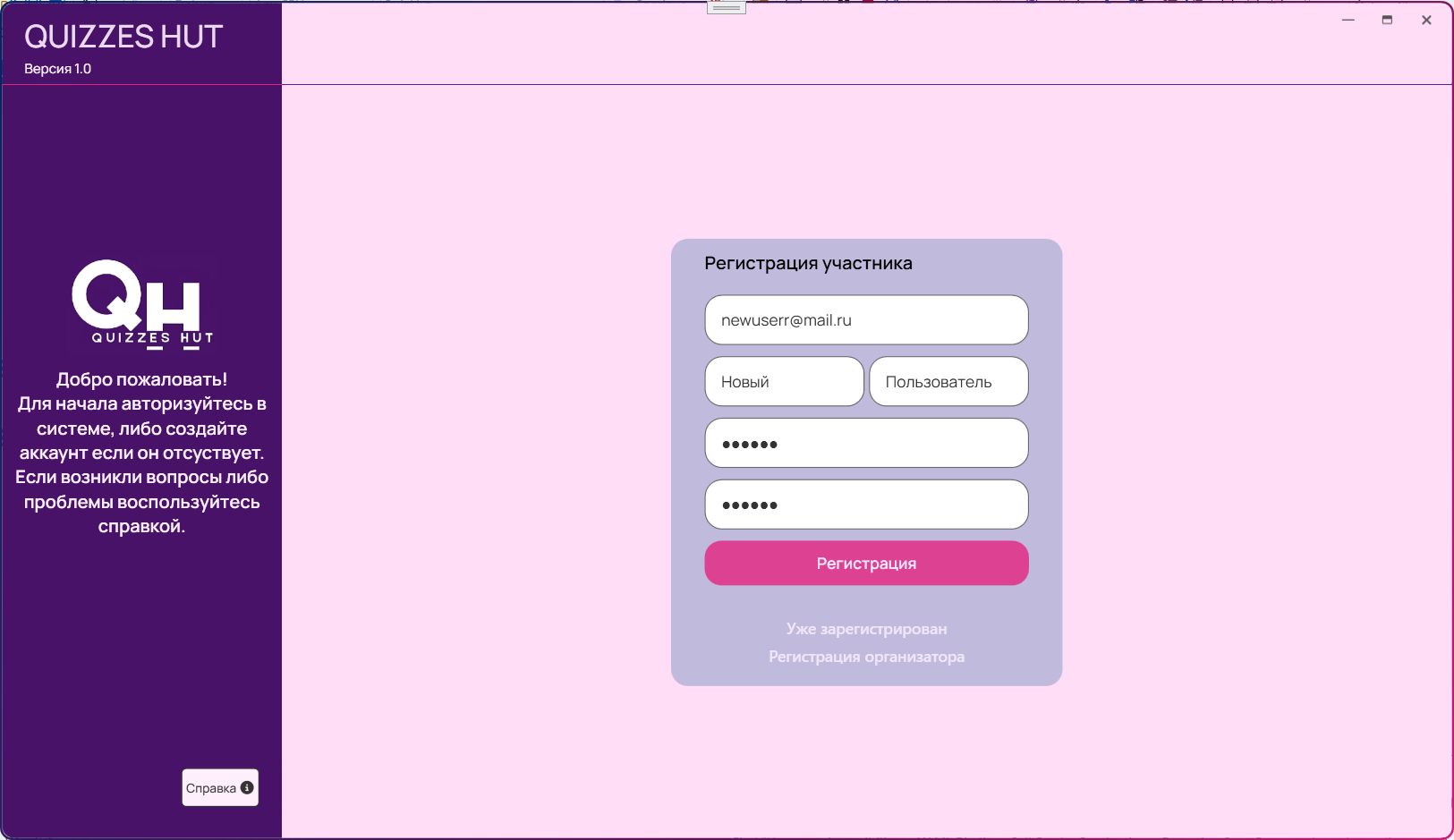


Рисунок Б.3

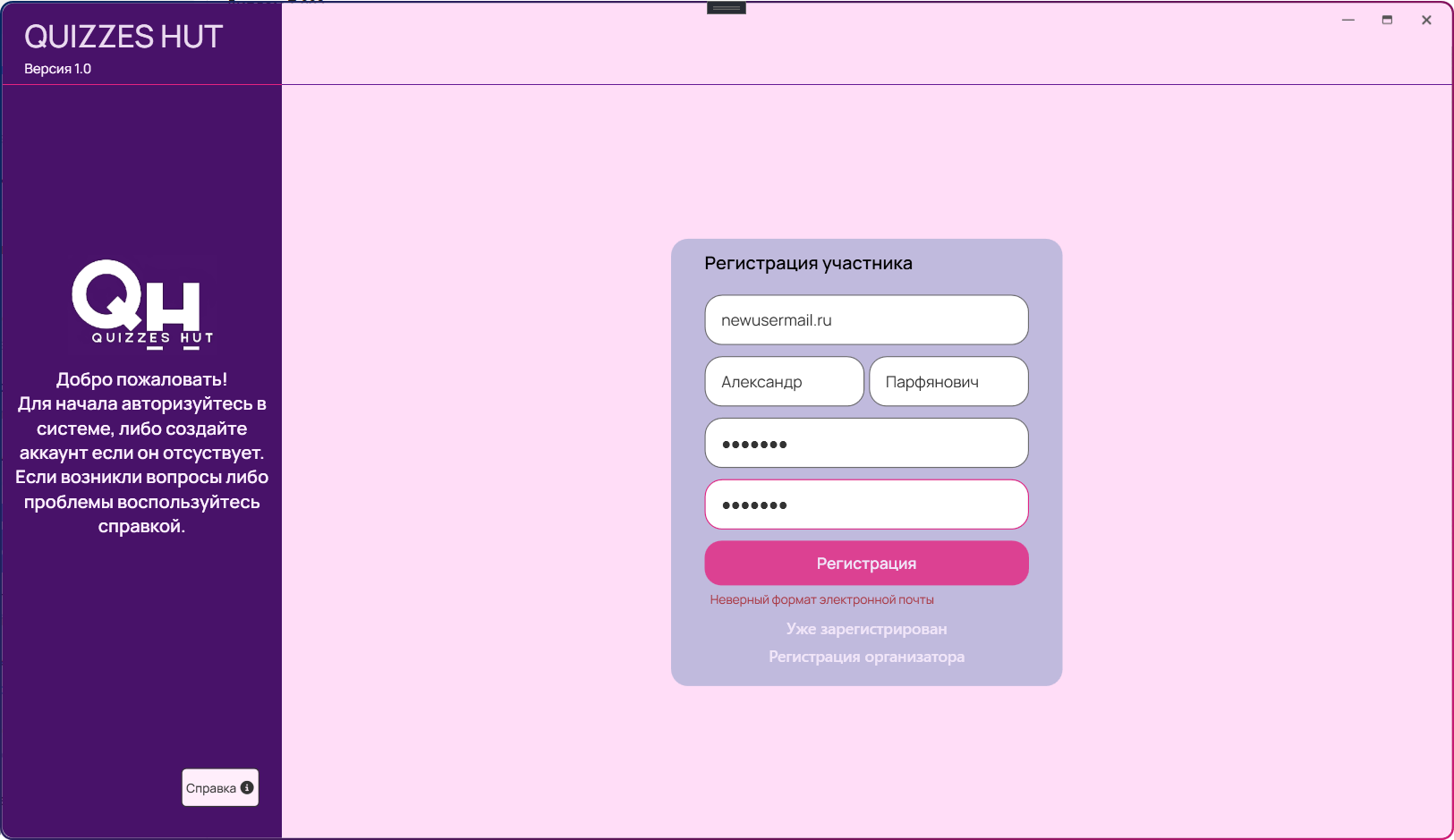


Рисунок Б.4

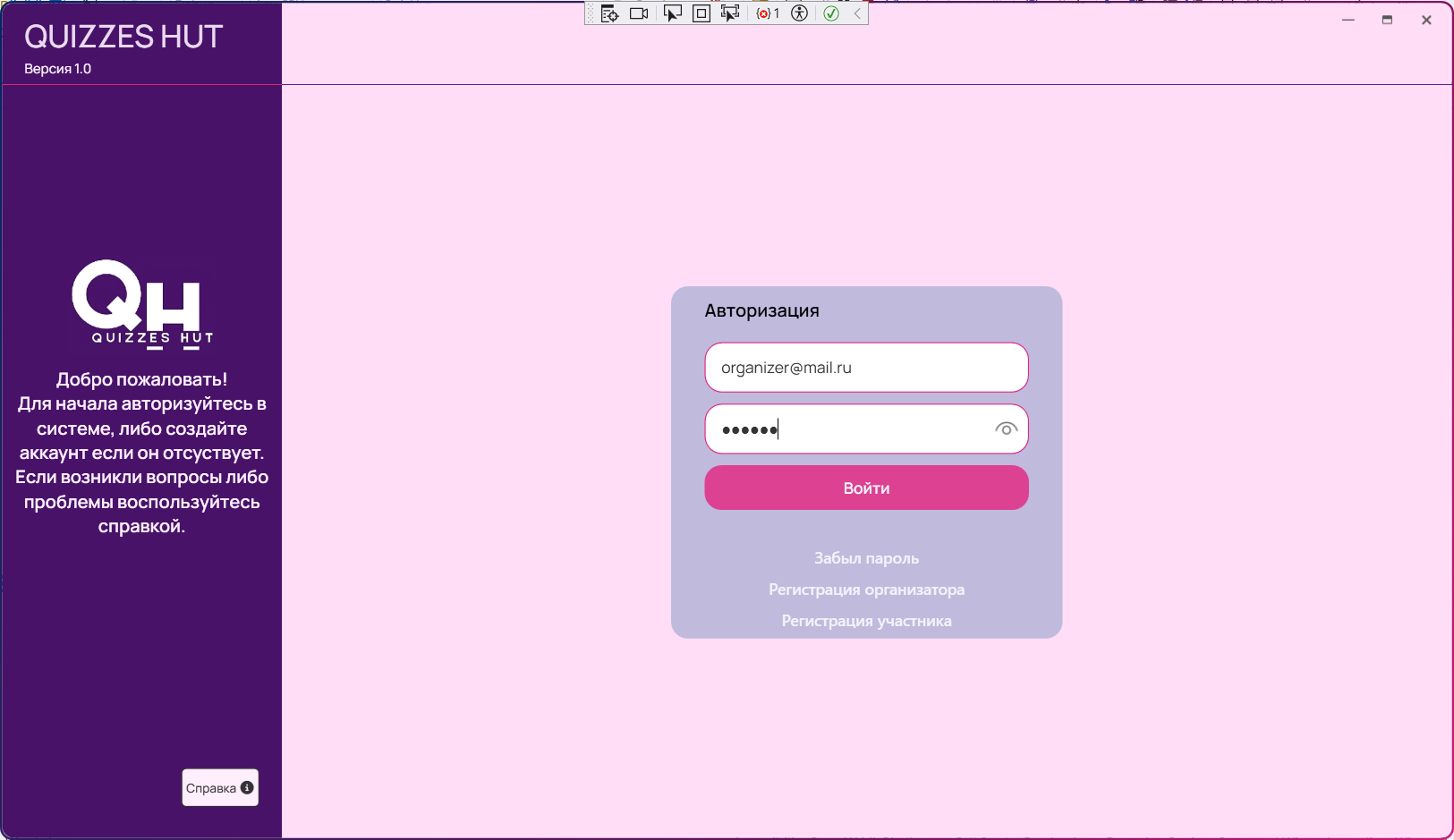


Рисунок Б.5

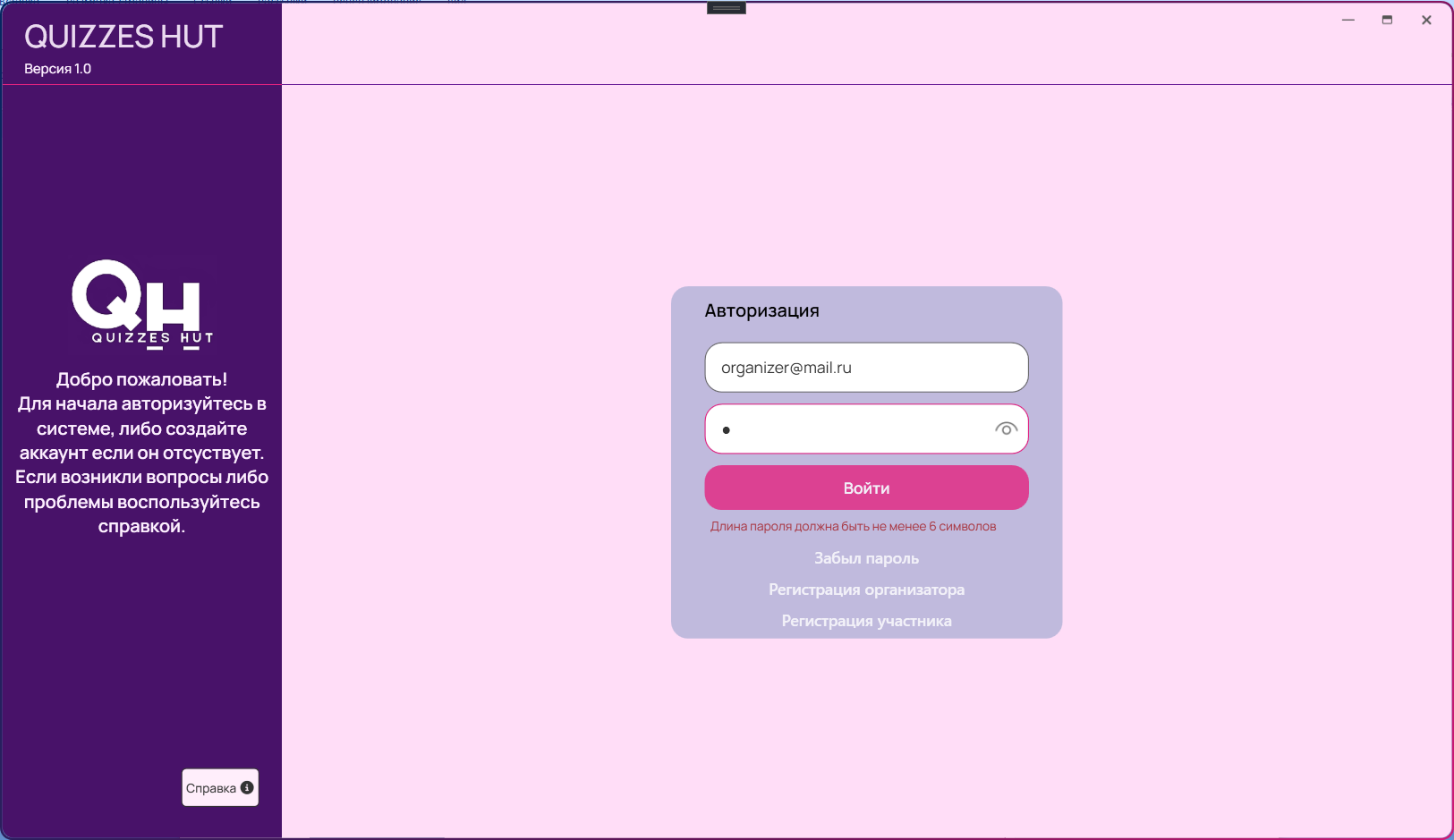


Рисунок Б.6

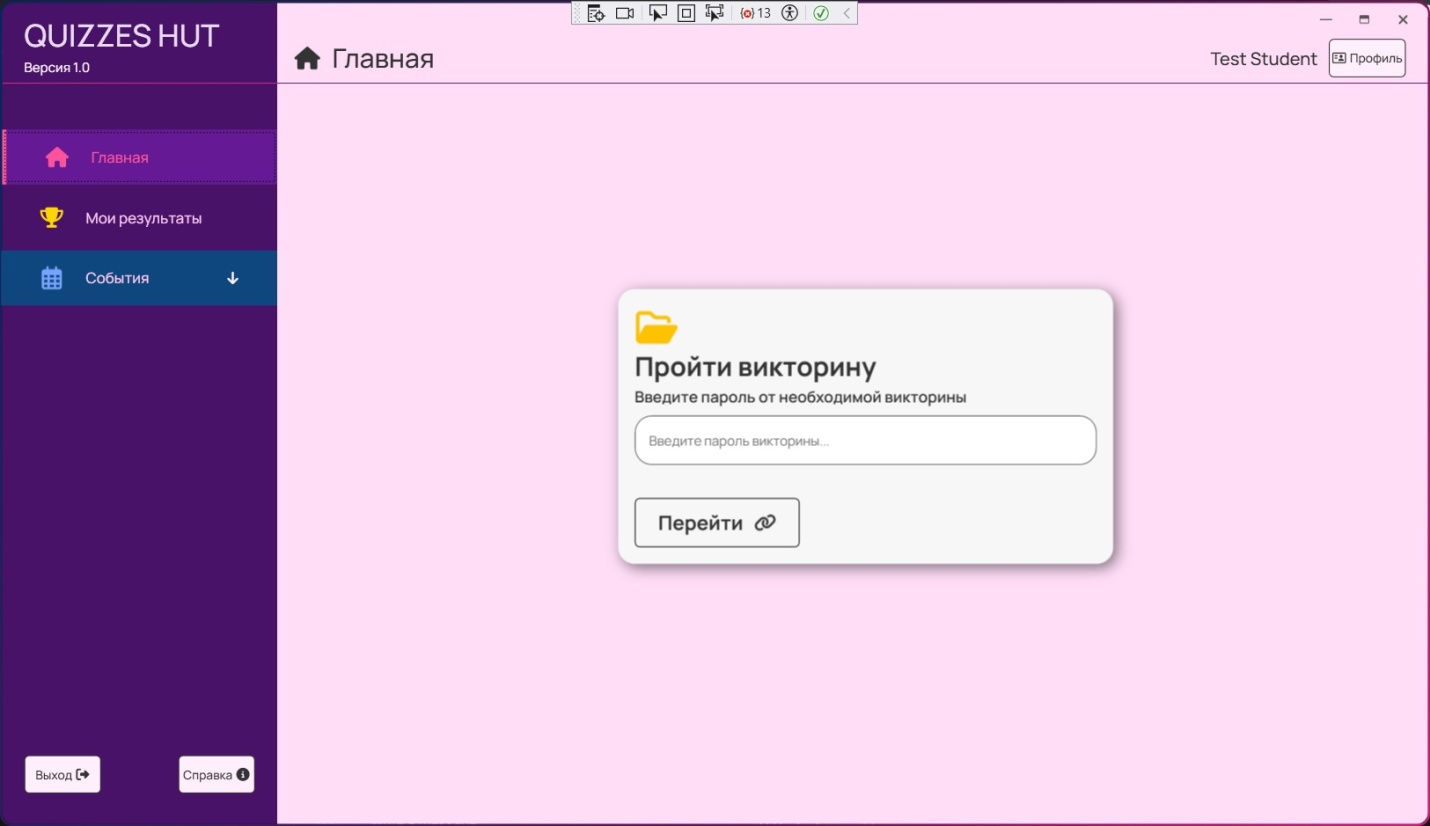


Рисунок Б.7

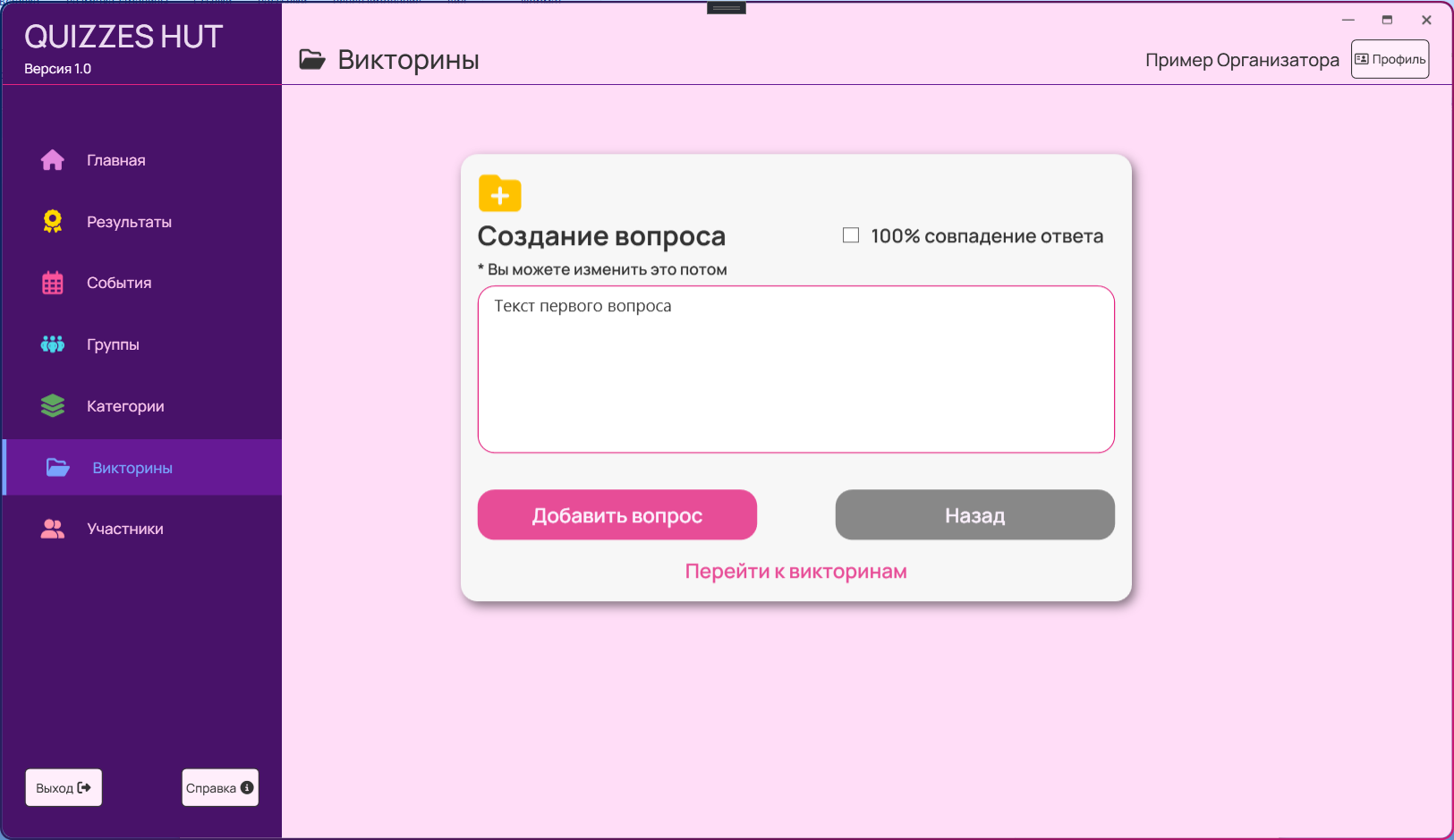


Рисунок Б.8

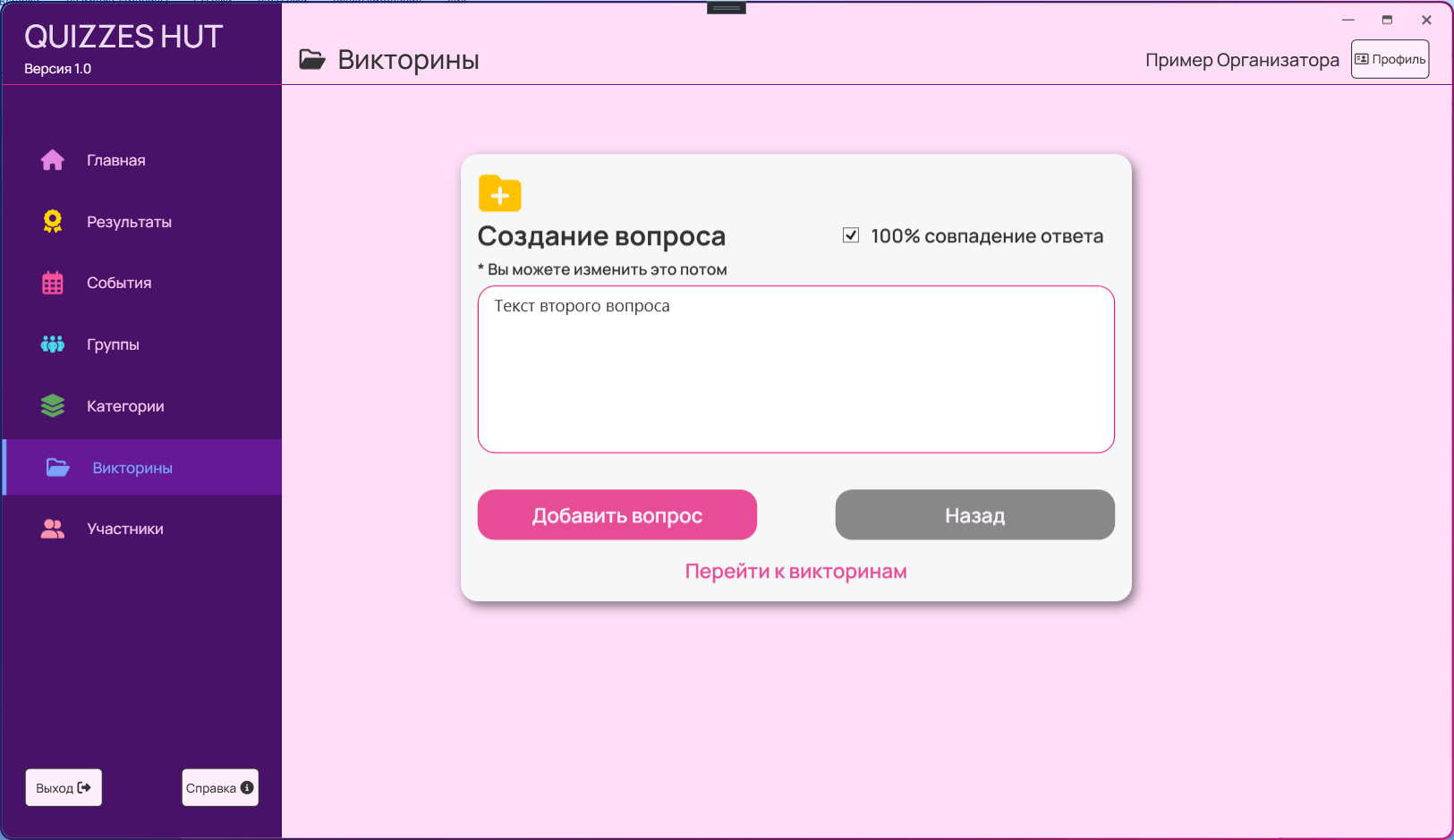


Рисунок Б.9

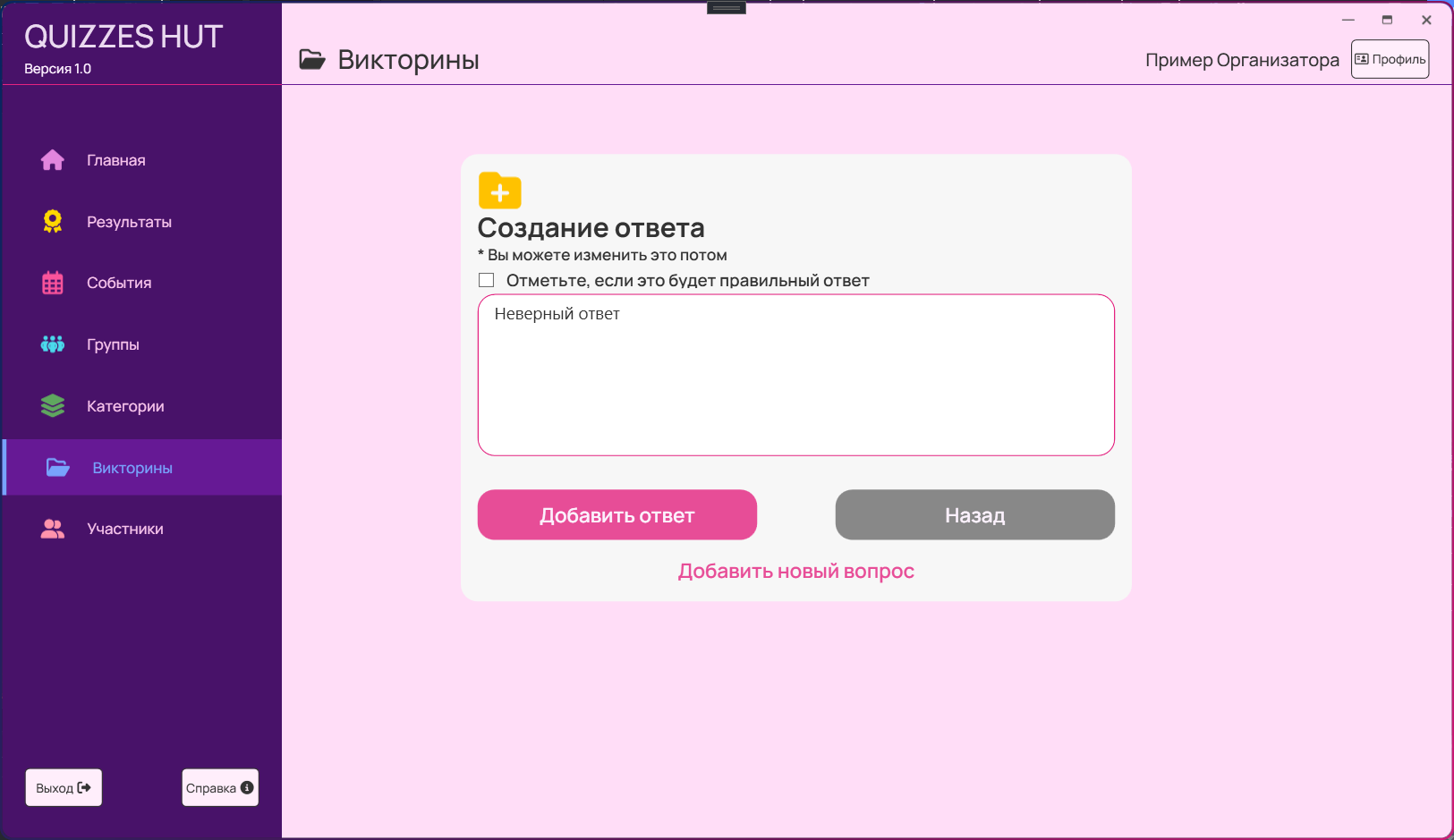


Рисунок Б.10

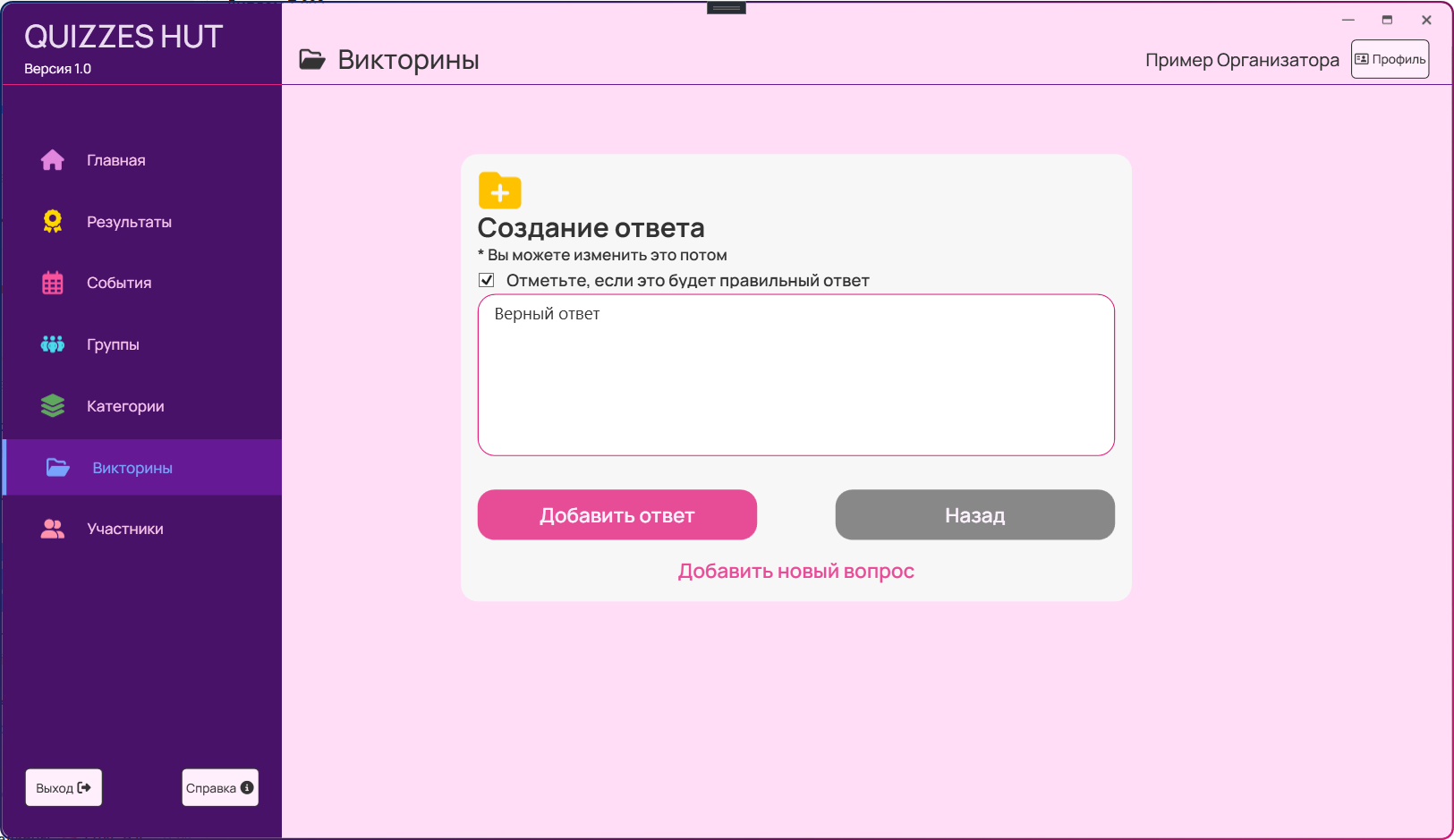


Рисунок Б.11

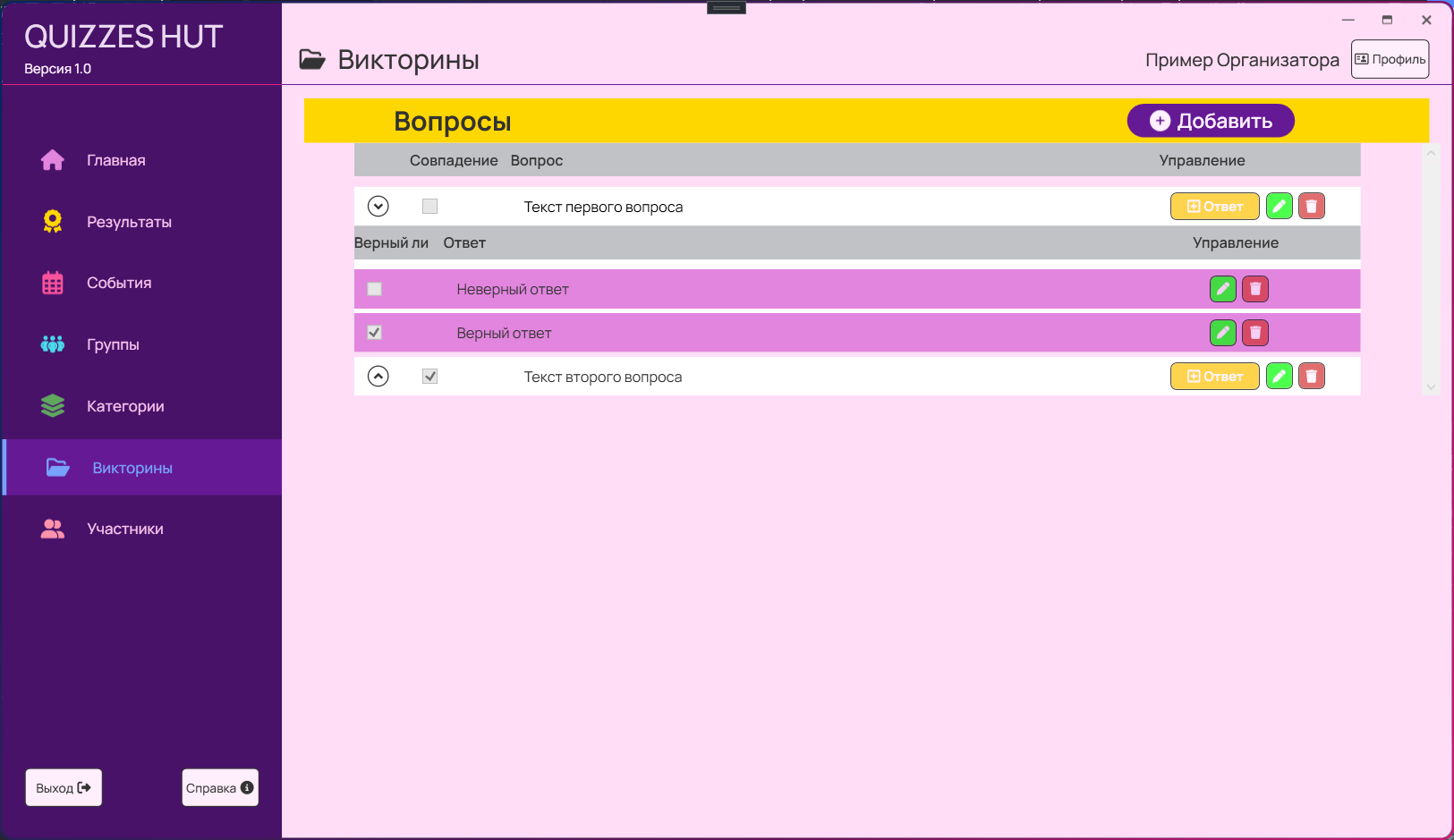


Рисунок Б.12

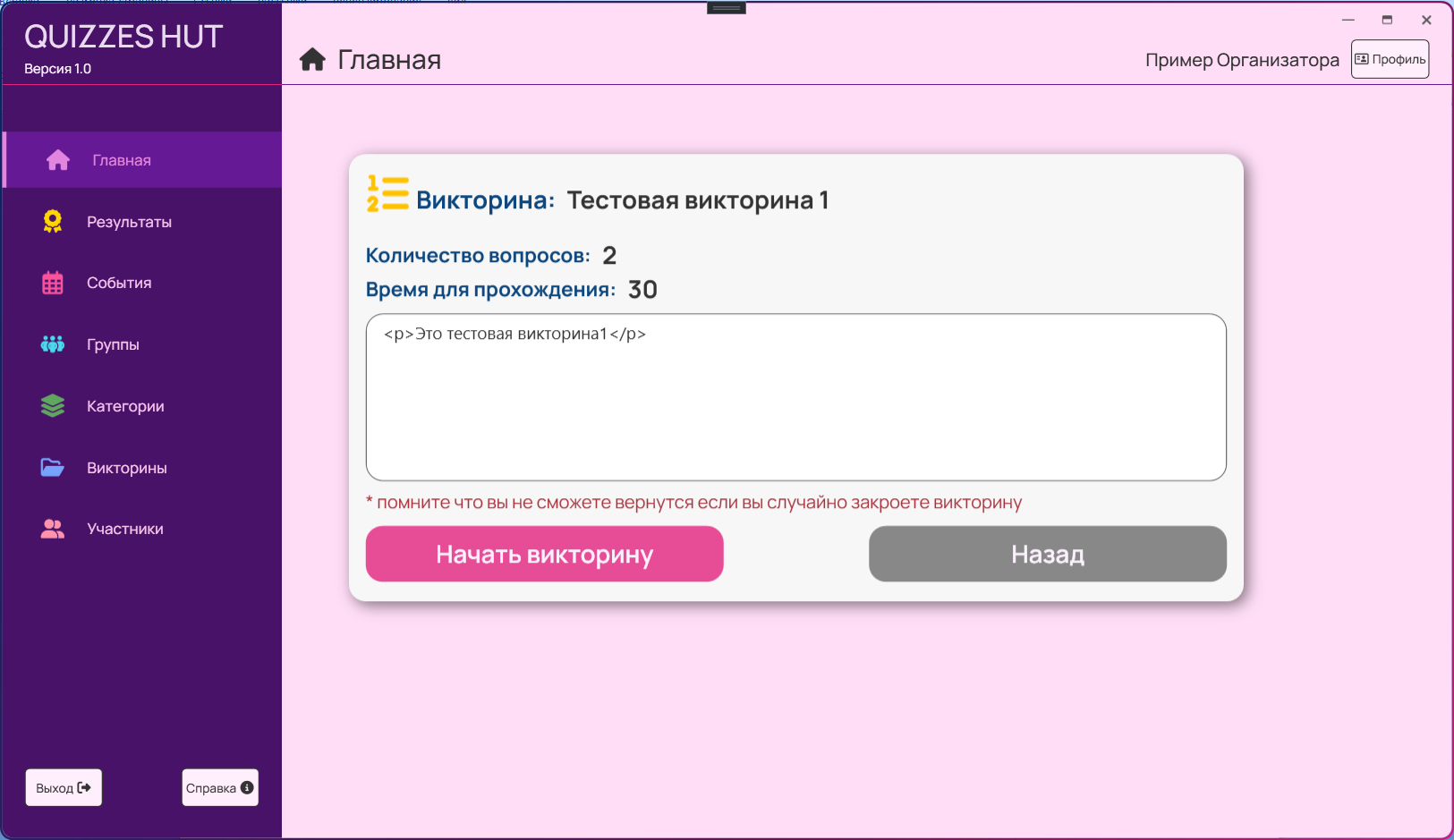


Рисунок Б.13

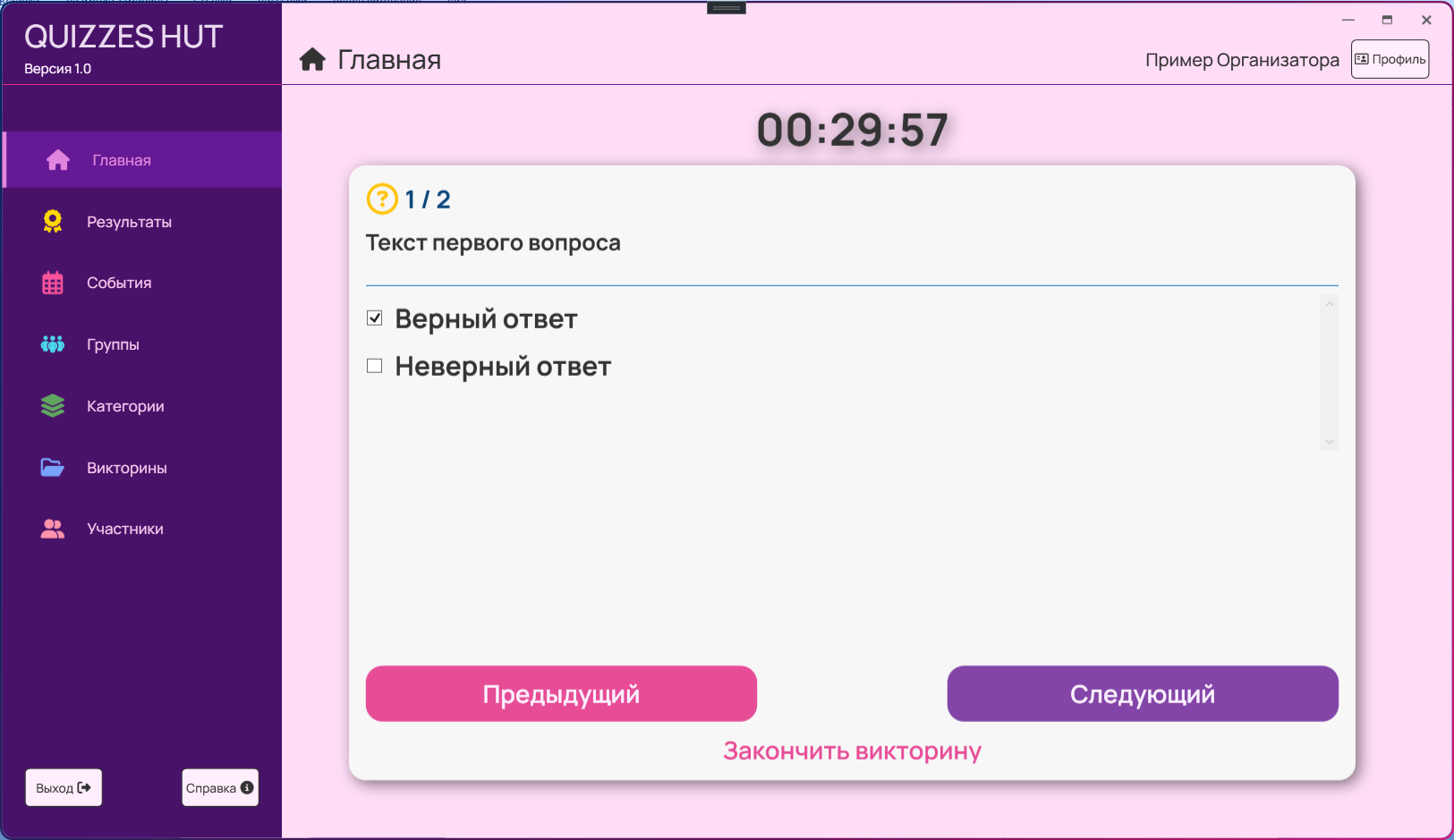


Рисунок Б.14

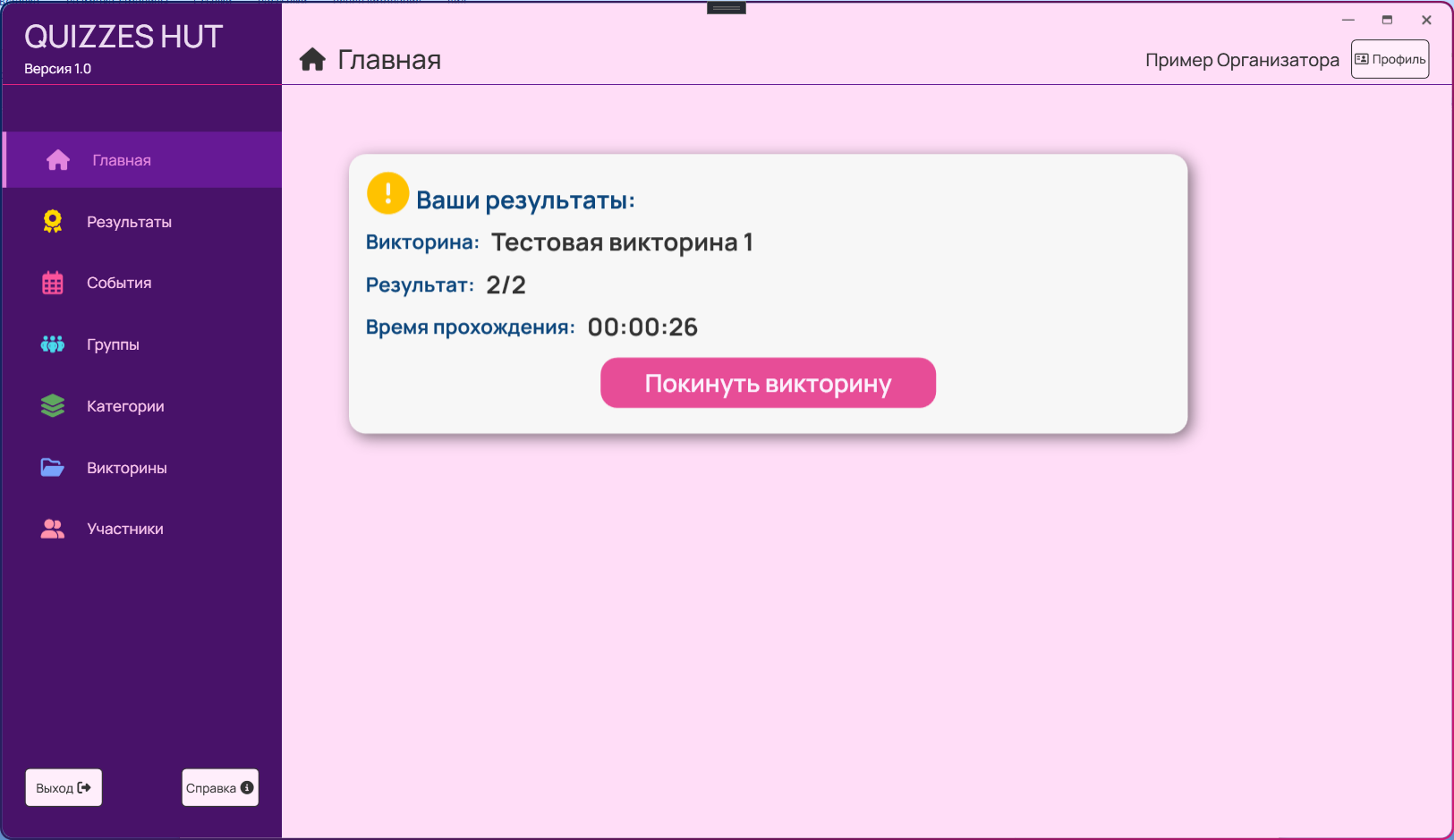


Рисунок Б.15



ДП Т.992103.401 ГЧ

ДП Т.992103.401 ГЧ

Т. контр.

Н. контр.

Утверд.

КБП

Листов 3 3Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.Инв.№дубл.

№докум

Дата

Дата

Изм.

Изм.

Лист

ист

Взам.инв.№Взам.инв.№

Подп. и дата

*СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ ЗНАНИЙ УЧАСТНИКОВ* *ОЛИМПИАДЫ*

*Диаграмма вариантов использования*

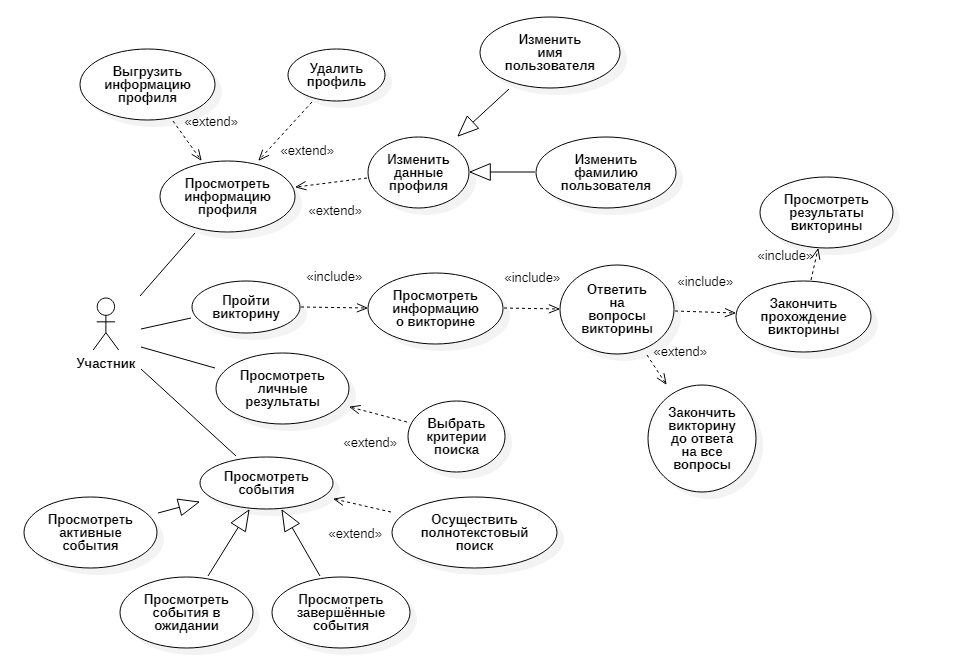
МассаМасса

Лит.

Масштаб

Лист 1

Реценз.



*Багласова Т.Г.*

*Багласова Т.Г.*

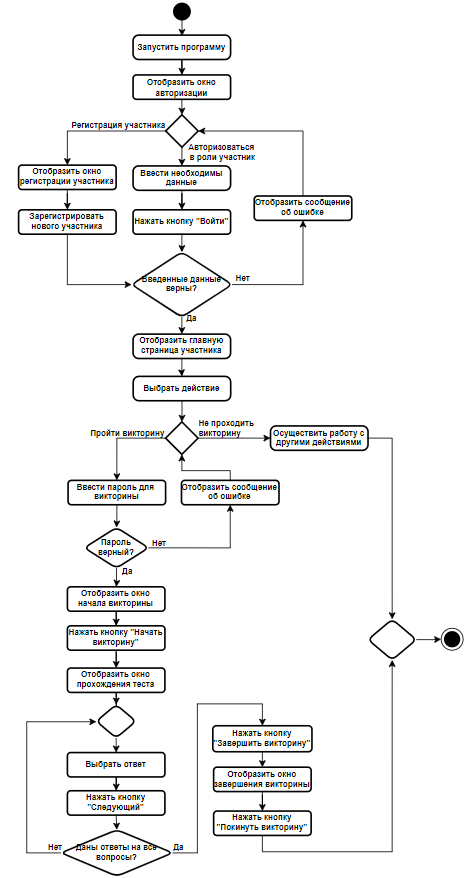
*Подпись*

*Басалыга Л.В.*

*Сватко И.Б.*

*Парфянович А.С.*

*Багласова Т.Г.*



*Подпись*

*Багласова Т.Г.*

*Багласова Т.Г.*

*Басалыга Л.В.*

*Сватко И.Б.*

*Парфянович А.С.*

ДП Т.992103.401 ГЧ

Т. контр.

Н. контр.

Утверд.

КБП

*Листов 3* *Листов* *3*

Провер.

Разраб.

Подп. и дата

Инв.№дубл.

№докум

Дата

Изм.

Лист

ист

Взам.инв.№

Подп. и дата

*СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ ЗНАНИЙ УЧАСТНИКОВ* *ОЛИМПИАДЫ*

*Диаграмма ДЕЯТЕЛЬНОСТИ для прохождения*

Масса

Лит.

Масштаб

Лист 2

Реценз.



ДП Т.992103.401 ГЧ

ДП Т.992103.401 ГЧ

Т. контр.

Н. контр.

Утверд.

КБП

Листов 3 3Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.Инв.№дубл.

№докум

Дата

Дата

Изм.

Изм.

Лист

ист

Взам.инв.№Взам.инв.№

Подп. и дата

*СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ ЗНАНИЙ УЧАСТНИКОВ* *ОЛИМПИАДЫ*

*Диаграмма*

«*Сущность-связь*»

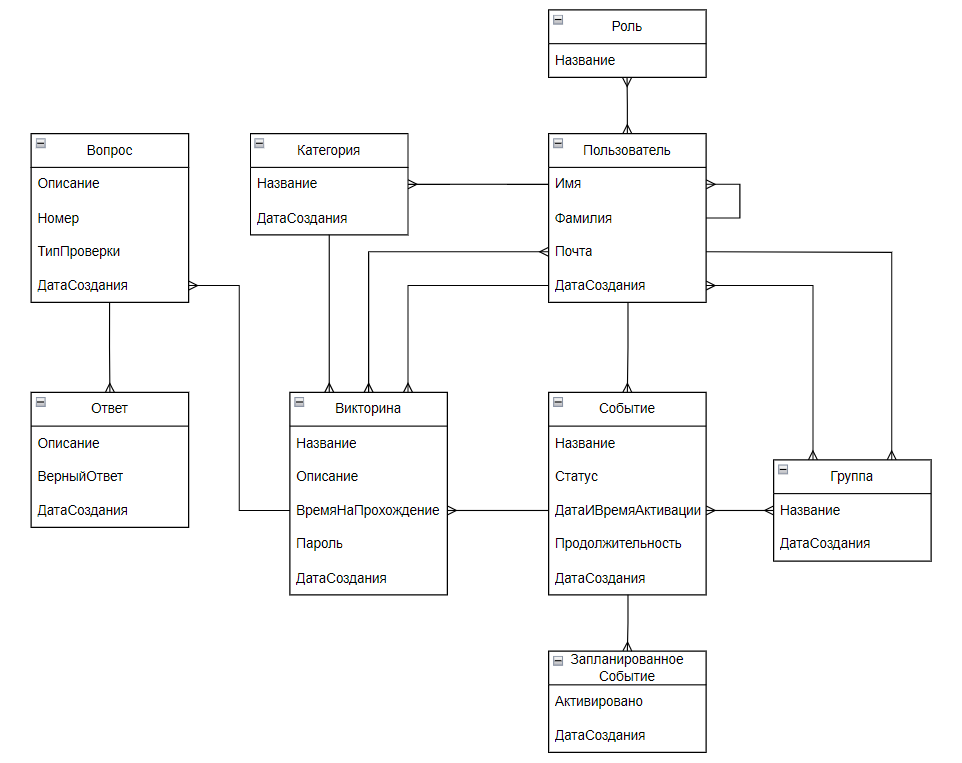
МассаМасса

Лит.

Масштаб

Лист 3

Реценз.



*Парфянович А.С.*

*Басалыга Л.В.*

*Сватко И.Б.*

*Подпись*

*Багласова Т.Г.*

*Багласова Т.Г.*